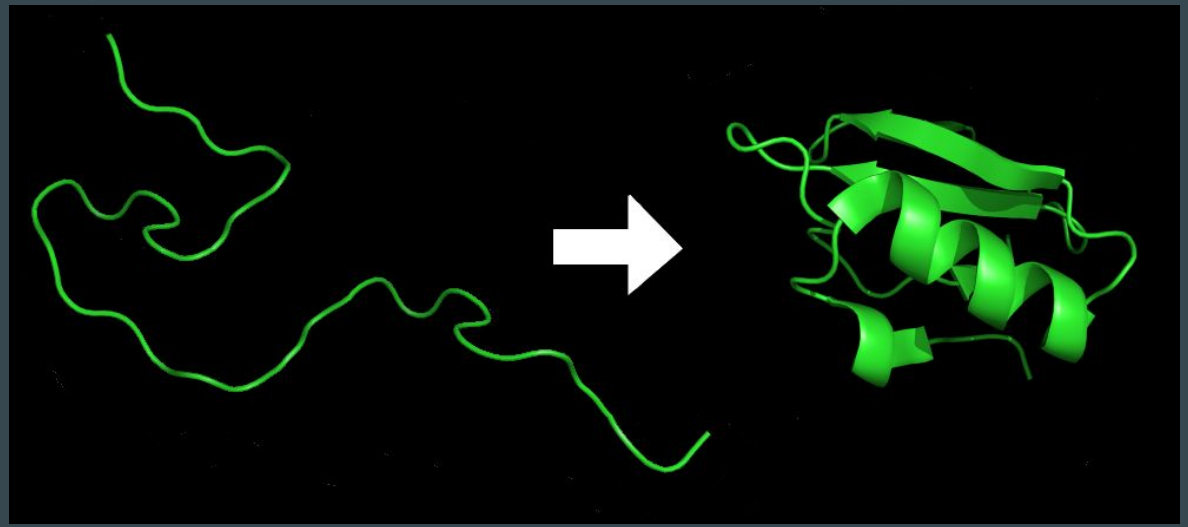
Folding

Grupo 8 - Eduardo V. Lima

Contextualização teórica

- Simplificação do processo de estabilização de uma proteína por dobras
- Síntese por ribossomo

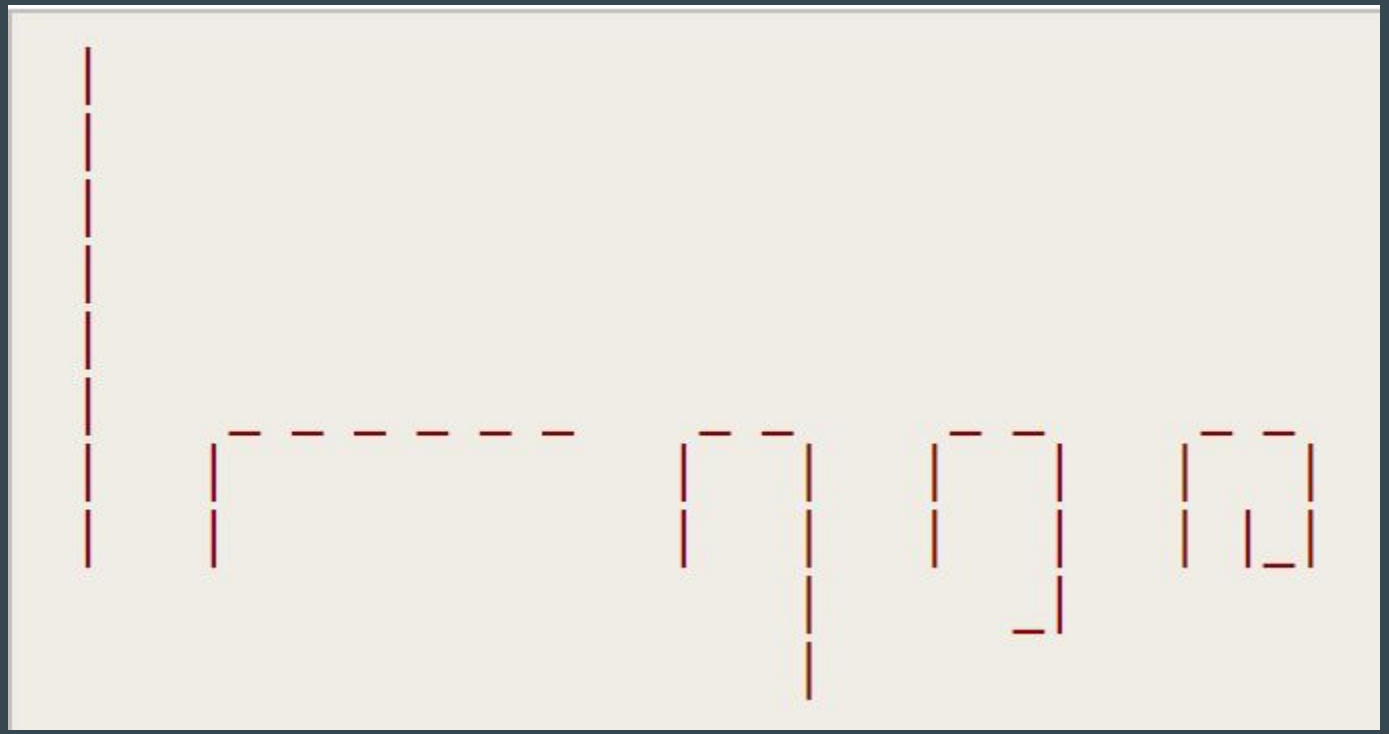


Contextualização do problema

- A partir de uma proteína reta e não transformada, encontrar um plano de dobras que resulte na transformação desejada
- Assume-se que a proteína é uma linha composta por N elementos equidistantes em um plano cartesiano
- A ação possível é rotacionar um elemento em 90 graus no sentido horário ou anti-horário
- Os elementos não podem ocupar o mesmo espaço no plano

Exemplo - Objetivo

Exemplo - Plano de ações



Domínio

```
(define (domain folding)
(:requirements :adl :action-costs)

(:types
    node - object
    coord - object
    direction - object
    rotation - object
)

(:constants
    left right up down - direction
    clockwise counterclockwise - rotation
)
```

Domínio

```
(:predicates
    ;; ?dfrom rotated by ?r ends up as ?d2to
    (NEXT-DIRECTION ?dfrom - direction ?r - rotation ?d2to - direction)
    ;; ?cnext = ?c + 1
    (COORD-INC ?c ?cnext - coord)
    ;; Last node of the string
    (END-NODE ?n - node)
    ;; ?n2 follows right after ?n1 in the string
    (CONNECTED ?n1 ?n2 - node)

    ;; Position of the node in the grid
    (at ?n - node ?x ?y - coord)
    ;; Heading of the outgoing edge of this node
    (heading ?n - node ?dir - direction)
    ;; The coordinates are not occupied by any node
    (free ?x ?y - coord)
    ;; Flag indicating we are in the process of rotating the string
    (rotating)
    ;; Cursor storing that ?nstart was rotated by ?r and the next node to
    ;; process is ?n
    (node-first-pass-next ?nstart - node ?r - rotation ?n - node)
    ;; Cursor for the second pass
    (node-second-pass-next ?n - node)
)
```

Domínio

```
(:functions
    (total-cost) - number
    (rotate-cost) - number
    (update-cost) - number
)


;; Rotates the string after this node and start the first pass computing
;; absolute directions and coordinates of other nodes
(:action rotate
    :parameters (?n - node ?r - rotation ?fromdir ?todir - direction)
    :precondition
        (and
            (not (rotating))
            (NEXT-DIRECTION ?fromdir ?r ?todir)
            (heading ?n ?fromdir)
        )
    :effect
        (and
            (not (heading ?n ?fromdir))
            (heading ?n ?todir)
            (rotating)
            (node-first-pass-next ?n ?r ?n)
            (increase (total-cost) (rotate-cost))
        )
)
```

Domínio

```
(:action rotate
    :parameters (?n - node ?r - rotation ?fromdir ?todir - direction)
    :precondition
        (and
            (not (rotating))
            (NEXT-DIRECTION ?fromdir ?r ?todir)
            (heading ?n ?fromdir)
        )
    :effect
        (and
            (not (heading ?n ?fromdir))
            (heading ?n ?todir)
            (rotating)
            (node-first-pass-next ?n ?r ?n)
            (increase (total-cost) (rotate-cost))
        )
)
```

Domínio

```
;; The first pass fixes the direction of nodes and removes all (at ...) facts
(:action rotate-first-pass
    :parameters (?nstart - node ?r - rotation
                 ?n1 - node
                 ?n2 - node ?n2x ?n2y - coord ?n2dir ?n2setdir - direction)
    :precondition
        (and
            (CONNECTED ?n1 ?n2)
            (NEXT-DIRECTION ?n2dir ?r ?n2setdir)
            (node-first-pass-next ?nstart ?r ?n1)
            (at ?n2 ?n2x ?n2y)
            (heading ?n2 ?n2dir)
        )
    :effect
        (and
            (not (node-first-pass-next ?nstart ?r ?n1))
            (node-first-pass-next ?nstart ?r ?n2)
            (not (at ?n2 ?n2x ?n2y))
            (free ?n2x ?n2y)
            (not (heading ?n2 ?n2dir))
            (heading ?n2 ?n2setdir)
            (increase (total-cost) (update-cost))
        )
)
```

Domínio

```
(:action rotate-first-pass-end
    :parameters (?nstart - node ?r - rotation
                 ?n1 - node
                 ?n2 - node ?n2x ?n2y - coord)
    :precondition
        (and
            (END-NODE ?n2)
            (CONNECTED ?n1 ?n2)
            (node-first-pass-next ?nstart ?r ?n1)
            (at ?n2 ?n2x ?n2y)
        )
    :effect
        (and
            (not (at ?n2 ?n2x ?n2y))
            (free ?n2x ?n2y)
            (not (node-first-pass-next ?nstart ?r ?n1))
            (node-second-pass-next ?nstart)
            (increase (total-cost) (update-cost))
        )
)
```

```
(:action rotate-second-pass
    :parameters (?n1 - node ?n1x ?n1y - coord ?n1dir - direction
                 ?n2 - node ?n2x ?n2y - coord)
    :precondition
        (and
            (CONNECTED ?n1 ?n2)
            (node-second-pass-next ?n1)
            (at ?n1 ?n1x ?n1y)
            (heading ?n1 ?n1dir)
            (free ?n2x ?n2y)

            (or
                (and (= ?n1dir up)
                     (= ?n1x ?n2x)
                     (COORD-INC ?n1y ?n2y)
                )

                (and (= ?n1dir down)
                     (= ?n1x ?n2x)
                     (COORD-INC ?n2y ?n1y)
                )

                (and (= ?n1dir left)
                     (= ?n1y ?n2y)
                     (COORD-INC ?n2x ?n1x)
                )

                (and (= ?n1dir right)
                     (= ?n1y ?n2y)
                     (COORD-INC ?n1x ?n2x)
                )
            )
        )
    :effect
        (and
            (not (node-second-pass-next ?n1))
            (node-second-pass-next ?n2)
            (not (free ?n2x ?n2y))
            (at ?n2 ?n2x ?n2y)
            (increase (total-cost) (update-cost))
        )
)
```

Domínio

```
(:action rotate-second-pass-end
    :parameters (?n - node)
    :precondition
        (and
            (END-NODE ?n)
            (node-second-pass-next ?n)
        )
    :effect
        (and
            (not (node-second-pass-next ?n))
            (not (rotating))
            (increase (total-cost) (update-cost))
        )
)

)
```

Exemplo

```
;; .-.
;; | |
;; . .-.
;; |   |
;; x   .-.
(define (problem reverse-folding-asp-8-5-431814)
(:domain reverse-folding)

(:objects
    n1 n2 n3 n4 n5 n6 n7 n8 - node
    c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 - coord
)
(:init
    (NEXT-DIRECTION up clockwise right)
    (NEXT-DIRECTION up counterclockwise left)
    (NEXT-DIRECTION down clockwise left)
    (NEXT-DIRECTION down counterclockwise right)
    (NEXT-DIRECTION left clockwise up)
    (NEXT-DIRECTION left counterclockwise down)
    (NEXT-DIRECTION right clockwise down)
    (NEXT-DIRECTION right counterclockwise up)

    (COORD-INC c1 c2)
    (COORD-INC c2 c3)
    (COORD-INC c3 c4)
    (COORD-INC c4 c5)
    (COORD-INC c5 c6)
    (COORD-INC c6 c7)
    (COORD-INC c7 c8)
    (COORD-INC c8 c9)
    (COORD-INC c9 c10)
    (COORD-INC c10 c11)
    (COORD-INC c11 c12)
    (COORD-INC c12 c13)
    (COORD-INC c13 c14)
    (COORD-INC c14 c15)

    (CONNECTED n1 n2)
    (CONNECTED n2 n3)
    (CONNECTED n3 n4)
    (CONNECTED n4 n5)
    (CONNECTED n5 n6)
    (CONNECTED n6 n7)
    (CONNECTED n7 n8)
    (END-NODE n8)

    (at n1 c8 c8)
    (at n2 c8 c9)
    (at n3 c8 c10)
    (at n4 c8 c11)
    (at n5 c8 c12)
    (at n6 c8 c13)
    (at n7 c8 c14)
    (at n8 c8 c15)
    (heading n1 up)
    (heading n2 up)
    (heading n3 up)
    (heading n4 up)
    (heading n5 up)
    (heading n6 up)
    (heading n7 up)
    (free c1 c1)
    (free c1 c2)
    (free c1 c3)
    (free c1 c4)
    (free c1 c5)
    (free c1 c6)
    (free c1 c7)
    (free c1 c8)
    (free c1 c9)
    (free c1 c10)
    (free c1 c11)
    (free c1 c12)
    (free c1 c13)
    (free c1 c14)
    (free c1 c15)
    (free c2 c1)
    (free c2 c2)
    (free c2 c3)
    (free c2 c4)
    (free c2 c5)
    (free c2 c6)
    (free c2 c7)
    (free c2 c8)
    (free c2 c9)
    (free c2 c10)
    (free c2 c11)
    (free c2 c12)
    (free c2 c13)
    (free c2 c14)
    (free c2 c15)
    (free c3 c1)
    (free c3 c2)
    (free c3 c3)
    (free c3 c4)
    (free c3 c5)
    (free c3 c6)
    (free c3 c7)
    (free c3 c8)
    (free c3 c9)
    (free c3 c10)
    (free c3 c11)
    (free c3 c12)
    (free c3 c13)
    (free c3 c14)
    (free c3 c15)
    (free c4 c1)
    (free c4 c2)
    (free c4 c3)
    (free c4 c4)
    (free c4 c5)
    (free c4 c6)
    (free c4 c7)
    (free c4 c8)
    (free c4 c9)
    (free c4 c10)
    (free c4 c11)
    (free c4 c12)
    (free c4 c13)
    (free c4 c14)
    (free c4 c15)
    (free c5 c1)
    (free c5 c2)
    (free c5 c3)
    (free c5 c4)
    (free c5 c5)
    (free c5 c6)
    (free c5 c7)
    (free c5 c8)
    (free c5 c9)
    (free c5 c10)
    (free c5 c11)
    (free c5 c12)
    (free c5 c13)
    (free c5 c14)
    (free c5 c15)
    (free c6 c1)
    (free c6 c2)
    (free c6 c3)
    (free c6 c4)
    (free c6 c5)
    (free c6 c6)
    (free c6 c7)
    (free c6 c8)
    (free c6 c9)
    (free c6 c10)
    (free c6 c11)
    (free c6 c12)
    (free c6 c13)
    (free c6 c14)
    (free c6 c15)
    (free c7 c1)
    (free c7 c2)
    (free c7 c3)
    (free c7 c4)
```

Exemplo

```
    (= (total-cost) 0)
    (= (rotate-cost) 1)
    (= (update-cost) 0)
)
(:goal
    (and
        (at n1 c8 c8)
        (at n2 c8 c9)
        (at n3 c8 c10)
        (at n4 c9 c10)
        (at n5 c9 c9)
        (at n6 c10 c9)
        (at n7 c10 c8)
        (at n8 c11 c8)
        (not (rotating))
    )
)
(:metric minimize (total-cost))
)
```

Exemplo

```
;; Cost: 5
;; Length: 40
(rotate n3 clockwise up right) ;; cost: 1
(rotate-first-pass n3 clockwise n3 n4 c8 c11 up right) ;; cost: 0
(rotate-first-pass n3 clockwise n4 n5 c8 c12 up right) ;; cost: 0
(rotate-first-pass n3 clockwise n5 n6 c8 c13 up right) ;; cost: 0
(rotate-first-pass n3 clockwise n6 n7 c8 c14 up right) ;; cost: 0
(rotate-first-pass-end n3 clockwise n7 n8 c8 c15) ;; cost: 0
(rotate-second-pass n3 c8 c10 right n4 c9 c10) ;; cost: 0
(rotate-second-pass n4 c9 c10 right n5 c10 c10) ;; cost: 0
(rotate-second-pass n5 c10 c10 right n6 c11 c10) ;; cost: 0
(rotate-second-pass n6 c11 c10 right n7 c12 c10) ;; cost: 0
(rotate-second-pass n7 c12 c10 right n8 c13 c10) ;; cost: 0
(rotate-second-pass-end n8) ;; cost: 0
(rotate n4 clockwise right down) ;; cost: 1
(rotate-first-pass n4 clockwise n4 n5 c10 c10 right down) ;; cost: 0
(rotate-first-pass n4 clockwise n5 n6 c11 c10 right down) ;; cost: 0
(rotate-first-pass n4 clockwise n6 n7 c12 c10 right down) ;; cost: 0
(rotate-first-pass-end n4 clockwise n7 n8 c13 c10) ;; cost: 0
(rotate-second-pass n4 c9 c10 down n5 c9 c9) ;; cost: 0
(rotate-second-pass n5 c9 c9 down n6 c9 c8) ;; cost: 0
(rotate-second-pass n6 c9 c8 down n7 c9 c7) ;; cost: 0
(rotate-second-pass n7 c9 c7 down n8 c9 c6) ;; cost: 0
(rotate-second-pass-end n8) ;; cost: 0
```

Exemplo

```
(rotate n5 counterclockwise down right) ;; cost: 1
(rotate-first-pass n5 counterclockwise n5 n6 c9 c8 down right) ;; cost: 0
(rotate-first-pass n5 counterclockwise n6 n7 c9 c7 down right) ;; cost: 0
(rotate-first-pass-end n5 counterclockwise n7 n8 c9 c6) ;; cost: 0
(rotate-second-pass n5 c9 c9 right n6 c10 c9) ;; cost: 0
(rotate-second-pass n6 c10 c9 right n7 c11 c9) ;; cost: 0
(rotate-second-pass n7 c11 c9 right n8 c12 c9) ;; cost: 0
(rotate-second-pass-end n8) ;; cost: 0
(rotate n6 clockwise right down) ;; cost: 1
(rotate-first-pass n6 clockwise n6 n7 c11 c9 right down) ;; cost: 0
(rotate-first-pass-end n6 clockwise n7 n8 c12 c9) ;; cost: 0
(rotate-second-pass n6 c10 c9 down n7 c10 c8) ;; cost: 0
(rotate-second-pass n7 c10 c8 down n8 c10 c7) ;; cost: 0
(rotate-second-pass-end n8) ;; cost: 0
(rotate n7 counterclockwise down right) ;; cost: 1
(rotate-first-pass-end n7 counterclockwise n7 n8 c10 c7) ;; cost: 0
(rotate-second-pass n7 c10 c8 right n8 c11 c8) ;; cost: 0
(rotate-second-pass-end n8) ;; cost: 0
```

Resultados

Optimal Track

- Goal: Find an optimal plan
- Metric: number of plans solved

Resultados

Coverage	folding	labyrinth	quantum	recharg.	ricochet.	rubiks	slither.	SUM
ragnarok	8	**8**	13	**14**	**17**	10	**7**	77
scorpion-2023	8	5	**14**	**14**	**17**	10	6	74
odin	8	5	13	**14**	**17**	10	6	73
dofri	8	5	13	13	**17**	10	4	70
cegarplusplus	**9**	5	13	**14**	**17**	0	**7**	65
hapori-stonesoup-opt	7	2	13	**14**	9	**11**	6	62
fdss-2023-opt	7	3	13	13	12	9	4	61
hapori-mip2-opt	7	1	13	**14**	9	10	6	60
hapori-ibacop2-opt	6	1	13	12	15	7	4	58
hapori-greedy-opt	5	1	13	11	9	10	**7**	56
baseline-blind	7	1	7	12	11	8	4	50
decstar-opt	6	1	12	11	8	8	4	50
hapori-delfi-opt	5	2	12	12	8	0	2	41
complementary	5	1	12	13	3	0	3	37
decabstar	2	1	12	10	7	0	5	37
symk	3	1	9	13	4	0	**7**	37
fts-ms-opt	1	1	12	13	2	0	**7**	36
baseline-lmcut	2	1	12	8	5	0	6	34
hapori-epslr-opt	2	1	9	6	4	10	2	34
SymBD-2023-opt	2	1	9	13	1	0	6	32
dom-opt	2	1	12	6	4	0	6	31
hapori-epsdt-opt	1	0	9	6	4	7	4	31
dalai-opt	2	1	11	7	4	0	4	29
fts-sbd-opt	1	0	4	13	0	0	4	22

Resultados

Satisficing Track

- Goal: Find a plan with high quality
- Metric: $C^\star/C$
  - same as in 2008 and 2018 but different from 2011, 2014
  - reference plans by many different means

Resultados

Sat score	folding	labyrinth	quantum	recharg.	ricochet.	rubiks	slither.	SUM
maidu-sat	6.80	0.00	19.60	13.94	11.36	**14.16**	**6.00**	71.86
levitron-sat	8.66	0.00	19.60	13.94	11.44	**14.16**	4.00	71.79
fdss-2023-sat	8.95	0.00	19.49	13.80	8.51	14.13	**6.00**	70.88
baseline-lama	**9.70**	1.00	17.86	13.19	9.83	12.18	5.00	68.76
hapori-ibacop2-sat	8.69	3.91	17.07	13.46	7.37	11.22	**6.00**	67.72
disco-sat	8.58	0.00	17.78	**13.95**	9.14	11.77	5.00	66.24
cerberus-sat	6.39	0.00	18.03	10.43	12.73	10.93	**6.00**	64.52
spock	5.83	0.00	**19.67**	13.94	5.01	13.90	**6.00**	64.36
decstar-sat	5.40	0.00	17.33	13.29	8.41	12.42	**6.00**	62.85
tftm-argmax-sat	6.73	0.00	16.74	9.55	12.73	9.98	**6.00**	61.73
tftm-co1-sat	6.67	0.00	15.84	10.43	12.65	10.10	**6.00**	61.68
hapori-mip2-sat	5.64	3.91	18.86	13.28	5.04	10.00	4.00	60.74
dalai-sat	4.84	4.00	17.29	13.00	8.82	5.00	3.00	55.95
ApxNoveltyAnytime	5.00	0.00	18.31	8.00	**15.32**	5.00	4.00	55.63
hapori-stonesoup-sat	7.69	3.91	18.44	12.32	4.17	4.00	4.00	54.52
NoveltyFBAnytime	1.00	0.00	18.31	8.00	13.80	5.00	4.00	50.12
opcount4sat-sat	2.00	3.76	13.32	10.11	6.15	0.00	4.00	39.34
hapori-epsdt-sat	7.40	**14.14**	0.00	5.38	0.62	6.03	0.00	33.57
hapori-delfi-sat	3.81	0.00	12.52	8.85	2.35	4.00	1.00	32.54
FSM	3.03	0.00	6.60	8.22	3.00	6.00	3.00	29.84
powerlifted-sat	9.69	0.00	16.71	0.00	0.00	0.00	2.00	28.40
hapori-epslr-sat	1.00	0.00	0.00	3.45	0.00	11.00	5.00	20.45
hapori-greedy-sat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

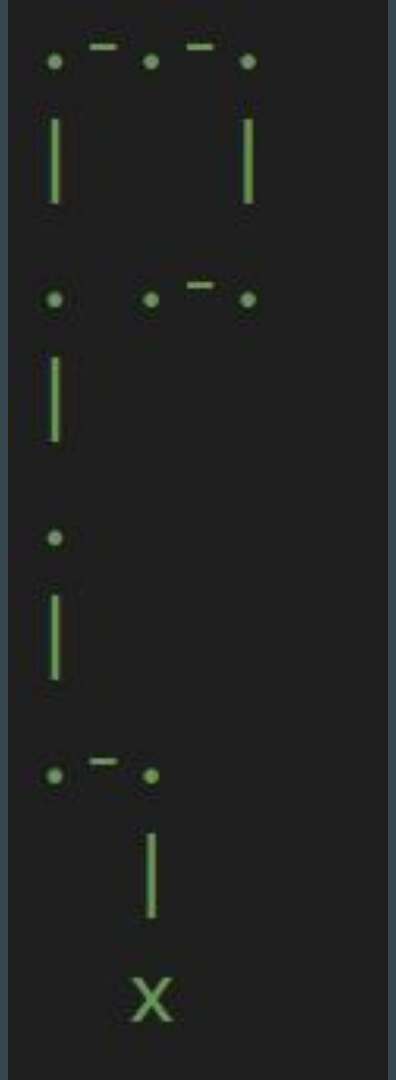
Resultados

Agile Track

- Goal: Find a plan quickly
- Metric: $1 - \log(t)/\log(300)$, or 1 if solved in first second
  - same as 2018
- Instance selection:
  - Same instances as in satisficing track

Resultados

Agile score	folding	labyrinth	quantum.	recharg.	ricochet.	rubiks	slither.	SUM
baseline-lama-first	3.35	0.00	16.86	3.79	0.94	**13.16**	2.19	40.28
decstar-agl	2.62	0.00	15.26	5.01	2.72	12.95	1.70	40.25
fdss-2023-agl	2.96	1.07	15.31	4.47	0.56	11.07	2.39	37.82
ApxNoveltyTarski	1.91	**2.66**	19.53	3.92	3.41	3.13	1.80	36.35
disco-agl	**3.39**	0.00	16.82	6.47	1.08	6.30	1.91	35.96
maidu-agl	2.52	0.28	15.57	4.41	1.57	9.64	1.93	35.93
levitron-agl	2.49	0.34	15.64	4.11	1.00	9.48	2.55	35.61
ApxNovelty	1.43	0.00	**19.64**	3.03	3.41	4.00	1.81	33.32
cerberus-agl	2.74	0.00	13.47	2.38	1.83	7.79	2.64	30.84
dalai-agl	3.05	0.34	13.64	**6.57**	**3.54**	2.67	1.01	30.81
hapori-stonesoup-agl	0.87	0.74	16.70	3.89	0.81	3.86	1.62	28.49
hapori-mip2-agl	0.37	0.77	13.60	2.47	0.82	7.12	**2.67**	27.82
tftm-argmax-agl	2.73	0.00	12.15	2.45	1.74	6.01	2.64	27.73
tftm-co1-agl	2.56	0.00	12.26	2.37	1.89	6.01	2.61	27.70
NoveltyFB	0.14	0.00	18.88	0.83	1.18	4.00	0.48	25.50
fts-ff-agl	0.00	0.00	9.61	5.19	0.55	0.00	1.97	17.31
hapori-epsdt-agl	1.84	2.65	0.00	2.82	0.51	5.20	0.00	13.03
hapori-epslr-agl	0.01	0.00	0.00	2.48	0.00	7.46	0.98	10.94
hapori-delfi-agl	0.68	0.00	5.06	2.40	1.53	0.64	0.52	10.83
opcount4sat-agl	0.00	0.04	3.04	6.28	0.00	0.00	0.74	10.10
FSM	0.00	0.00	0.82	0.95	0.24	3.17	0.09	5.26
powerlifted-agl	0.73	0.00	0.00	0.00	0.00	0.00	0.07	0.80
hapori-greedy-agl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00



Resultados

```
edvl@EDVL-Desktop:~/domain-folding$ ../fast-downward.sif --alias lama-first p-bias-spiral-30-01.pddl
INFO     planner time limit: None
INFO     planner memory limit: None
```

```
Solution found.
Peak memory: 28572 KB
Remove intermediate file output.sas
search exit code: 0

INFO     Planner time: 2.38s
```

```
; cost = 5 (general cost)
```

Resultados

```
edvl@EDVL-Desktop:~/domain-folding$ ../fast-downward.sif --alias seq-opt-lmcut p-bias-spiral-10-05.pddl
INFO     planner time limit: None
INFO     planner memory limit: None
```

```
Solution found.
Peak memory: 31324 KB
Remove intermediate file output.sas
search exit code: 0

INFO     Planner time: 32.73s
```

```
; cost = 5 (general cost)
```

Resultados

```
edvl@EDVL-Desktop:~/domain-folding$ ../fast-downward.sif --alias seq-sat-fd-autotune-2 p-bias-spiral-10-05.pddl
INFO     planner time limit: None
INFO     planner memory limit: None
```

```
Solution found.
Peak memory: 76416 KB
Remove intermediate file output.sas
search exit code: 0

INFO     Planner time: 466.55s
```

```
; cost = 15 (general cost)
```

```
; cost = 5 (general cost)
```

Obrigado!

...