Fast Downward Stone Soup: A Baseline for Building Planner Portfolios

Paper by:

Helmert, Roger, and Karpas

Presentation by:

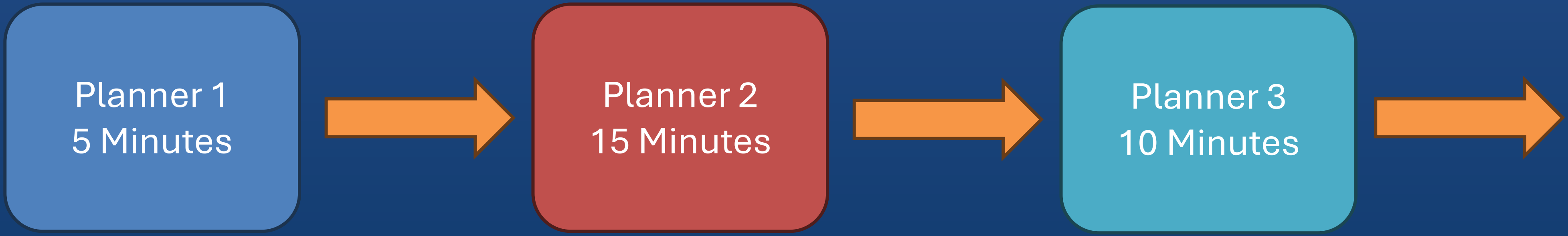
Michael Hentz

Observations

- After Fast Downward, two things made clear:
    - No current single algorithm or heuristic dominates all others
        - Different Heuristics work better on different problems
    - If a planner does not solve quickly, problem will probably not be solved
        - In the competition, each problem is 30 to 60 minutes
- Can we take the best of each?

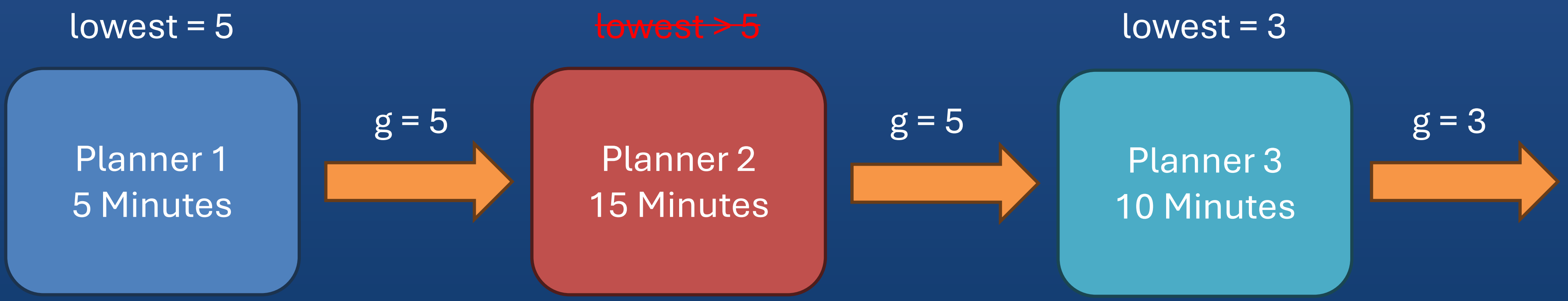
Sequential Planning Portfolios

- List of different planners in specific order
  - Different Planners and Different Heuristics
- Run each planner at a time with given slice of overall time
- Fast Downward Stone Soup
  - Naming: The Folk Tale of the Stone Soup
  - Sum of planners beats each individual planner



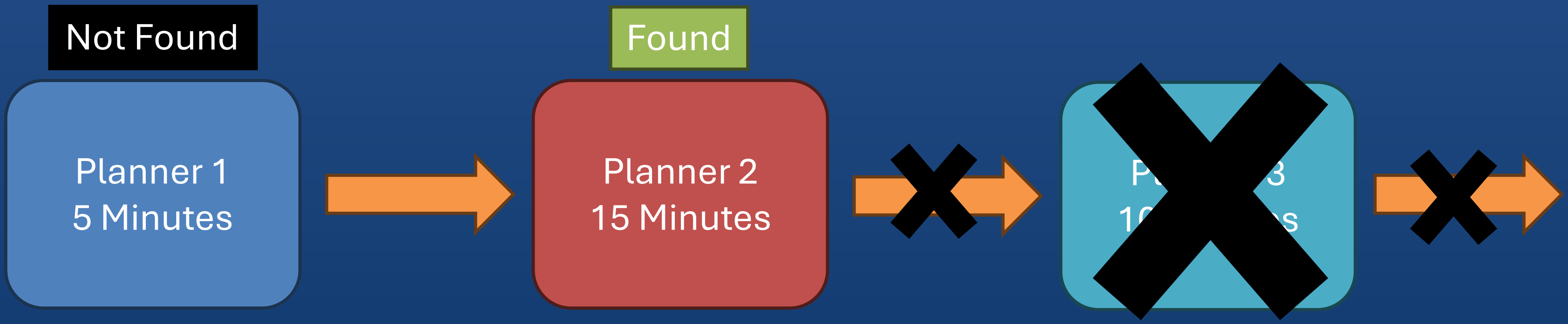
Optimal Portfolio Planner

- Find the best solution
  - Lowest Cost and Shortest Plan
- Planners communicate the current best in the sequence
  - Next planners can prune plans currently larger than the current best



Satisficing Portfolio Planner

- Find any solution
- Portfolio ends once any planner finds solution



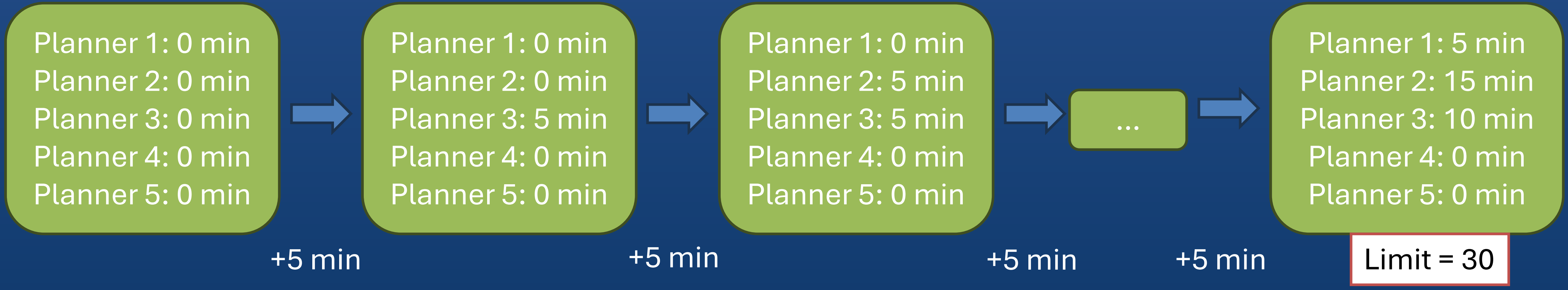
Describing Planner Portfolio

- Set of planners in portfolio is subset of possible planners in portfolio
- Portfolio represented as map of algorithms to allotted time
  - Time of 0 means algorithm not in portfolio
- Want to get “Holy Grail” – Each problem can be solved by at least one algorithm in portfolio

Planner 1: 5 min
Planner 2: 15 min
Planner 3: 10 min
Planner 4: 0 min
Planner 5: 0 min
…

Building Planner Portfolio

- Use Hill Climbing algorithm to find best portfolio
  - Start with portfolio of all algorithms set to 0
  - Successor portfolios are same as current but with allotted time added to one algorithm
  - Best successor chosen as new current portfolio
  - Repeat until sum of allotted time is more than overall time limit



Planner 1: 0 min
Planner 2: 0 min
Planner 3: 0 min
Planner 4: 0 min
Planner 5: 0 min

+5 min

Planner 1: 0 min
Planner 2: 0 min
Planner 3: 5 min
Planner 4: 0 min
Planner 5: 0 min

+5 min

Planner 1: 0 min
Planner 2: 5 min
Planner 3: 5 min
Planner 4: 0 min
Planner 5: 0 min

+5 min

…

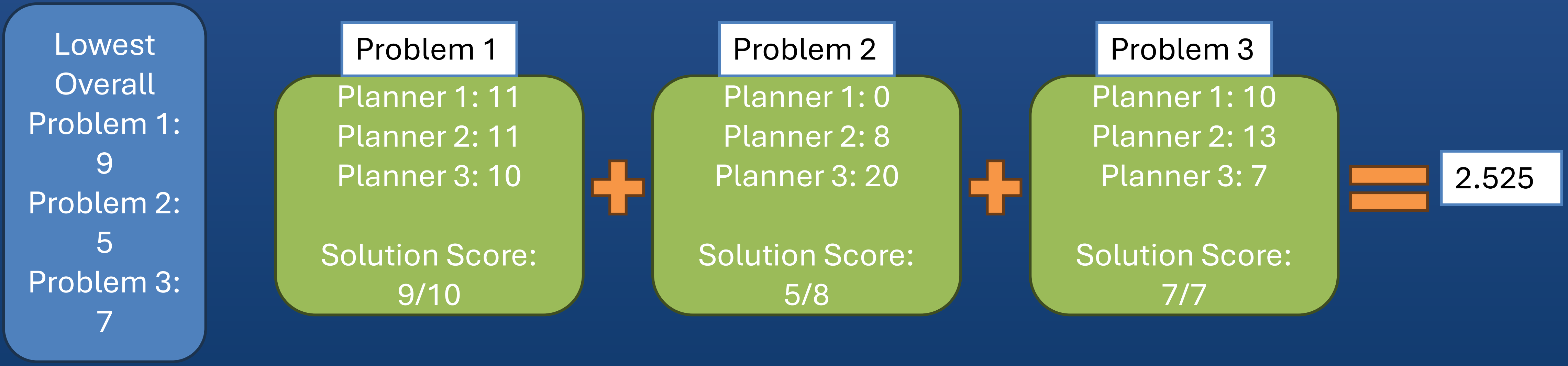
+5 min

Planner 1: 5 min
Planner 2: 15 min
Planner 3: 10 min
Planner 4: 0 min
Planner 5: 0 min

Limit = 30

Judging Planner Portfolio

- Portfolios can be judged based on how well they solve test problems
- Sum of all solution costs of all test problems for portfolio
  - Solution cost of a test problem for a portfolio is the lowest cost of any possible algorithm solving divided by lowest cost of portfolio solving



Lowest Overall
Problem 1: 9
Problem 2: 5
Problem 3: 7

Problem 1
Planner 1: 11
Planner 2: 11
Planner 3: 10

Solution Score: 9/10

+

Problem 2
Planner 1: 0
Planner 2: 8
Planner 3: 20

Solution Score: 5/8

+

Problem 3
Planner 1: 10
Planner 2: 13
Planner 3: 7

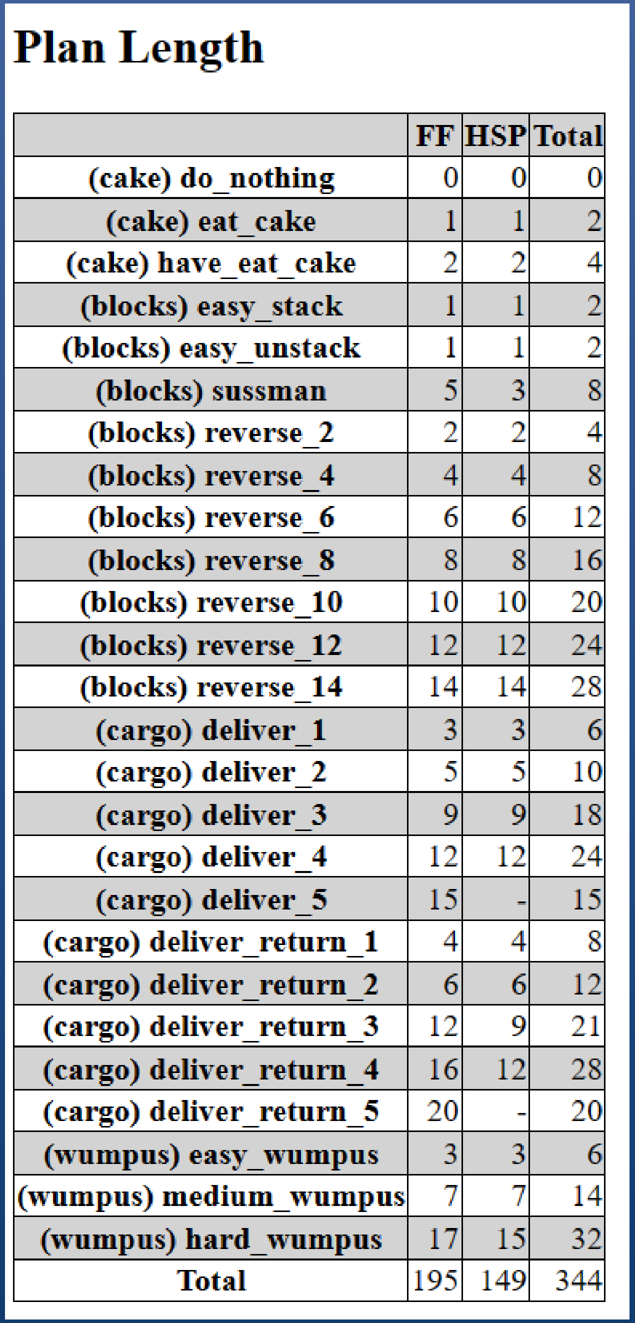
Solution Score: 7/7

=

2.525

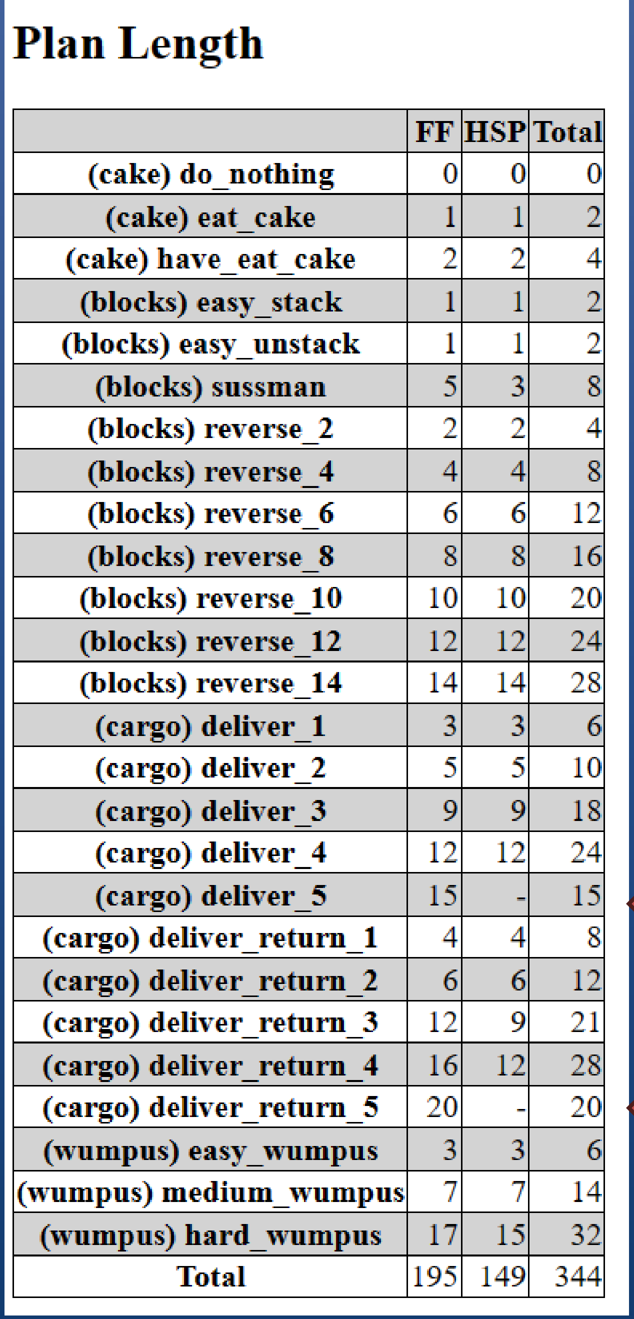
Demonstration

Optimal Track comparing HSP, Fast Forward, and Planner Portfolio of HSP and Fast Forward



Plan Length

	FF	HSP	Total
(cake) do_nothing	0	0	0
(cake) eat_cake	1	1	2
(cake) have_eat_cake	2	2	4
(blocks) easy_stack	1	1	2
(blocks) easy_unstack	1	1	2
(blocks) sussman	5	3	8
(blocks) reverse_2	2	2	4
(blocks) reverse_4	4	4	8
(blocks) reverse_6	6	6	12
(blocks) reverse_8	8	8	16
(blocks) reverse_10	10	10	20
(blocks) reverse_12	12	12	24
(blocks) reverse_14	14	14	28
(cargo) deliver_1	3	3	6
(cargo) deliver_2	5	5	10
(cargo) deliver_3	9	9	18
(cargo) deliver_4	12	12	24
(cargo) deliver_5	15	-	15
(cargo) deliver_return_1	4	4	8
(cargo) deliver_return_2	6	6	12
(cargo) deliver_return_3	12	9	21
(cargo) deliver_return_4	16	12	28
(cargo) deliver_return_5	20	-	20
(wumpus) easy_wumpus	3	3	6
(wumpus) medium_wumpus	7	7	14
(wumpus) hard_wumpus	17	15	32
Total	195	149	344



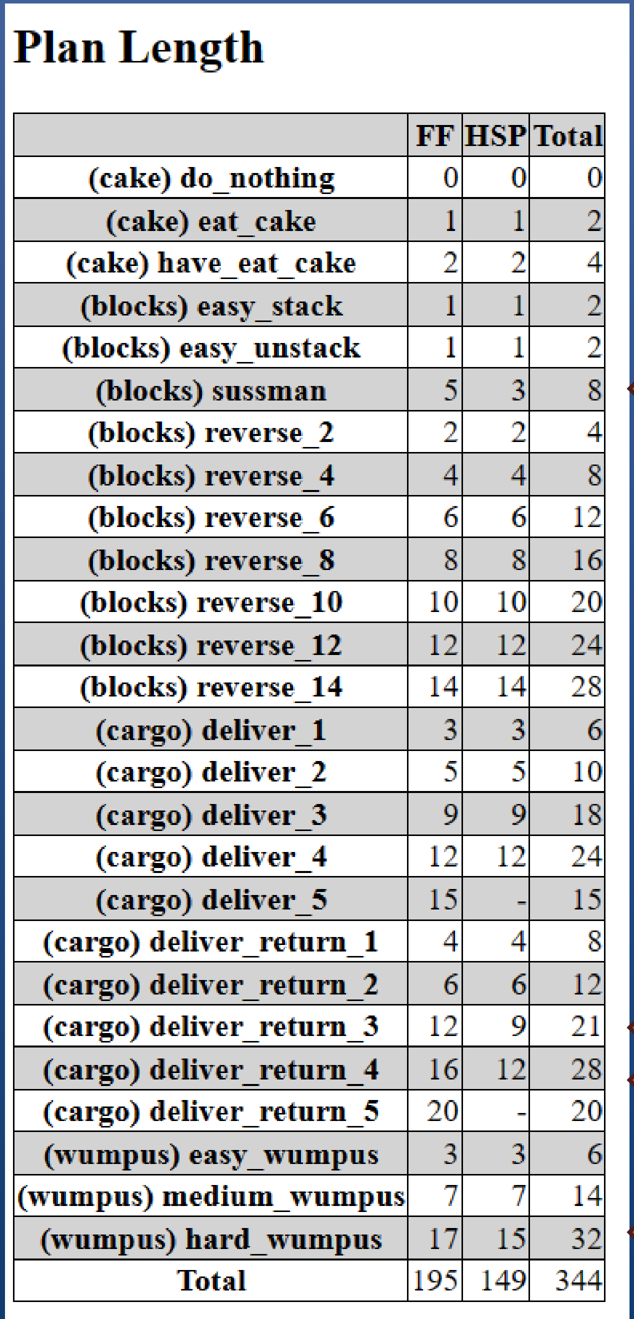
Plan Length

	FF	HSP	Total
(cake) do_nothing	0	0	0
(cake) eat_cake	1	1	2
(cake) have_eat_cake	2	2	4
(blocks) easy_stack	1	1	2
(blocks) easy_unstack	1	1	2
(blocks) sussman	5	3	8
(blocks) reverse_2	2	2	4
(blocks) reverse_4	4	4	8
(blocks) reverse_6	6	6	12
(blocks) reverse_8	8	8	16
(blocks) reverse_10	10	10	20
(blocks) reverse_12	12	12	24
(blocks) reverse_14	14	14	28
(cargo) deliver_1	3	3	6
(cargo) deliver_2	5	5	10
(cargo) deliver_3	9	9	18
(cargo) deliver_4	12	12	24
(cargo) deliver_5	15	-	15
(cargo) deliver_return_1	4	4	8
(cargo) deliver_return_2	6	6	12
(cargo) deliver_return_3	12	9	21
(cargo) deliver_return_4	16	12	28
(cargo) deliver_return_5	20	-	20
(wumpus) easy_wumpus	3	3	6
(wumpus) medium_wumpus	7	7	14
(wumpus) hard_wumpus	17	15	32
Total	195	149	344

HSP

Solves

24



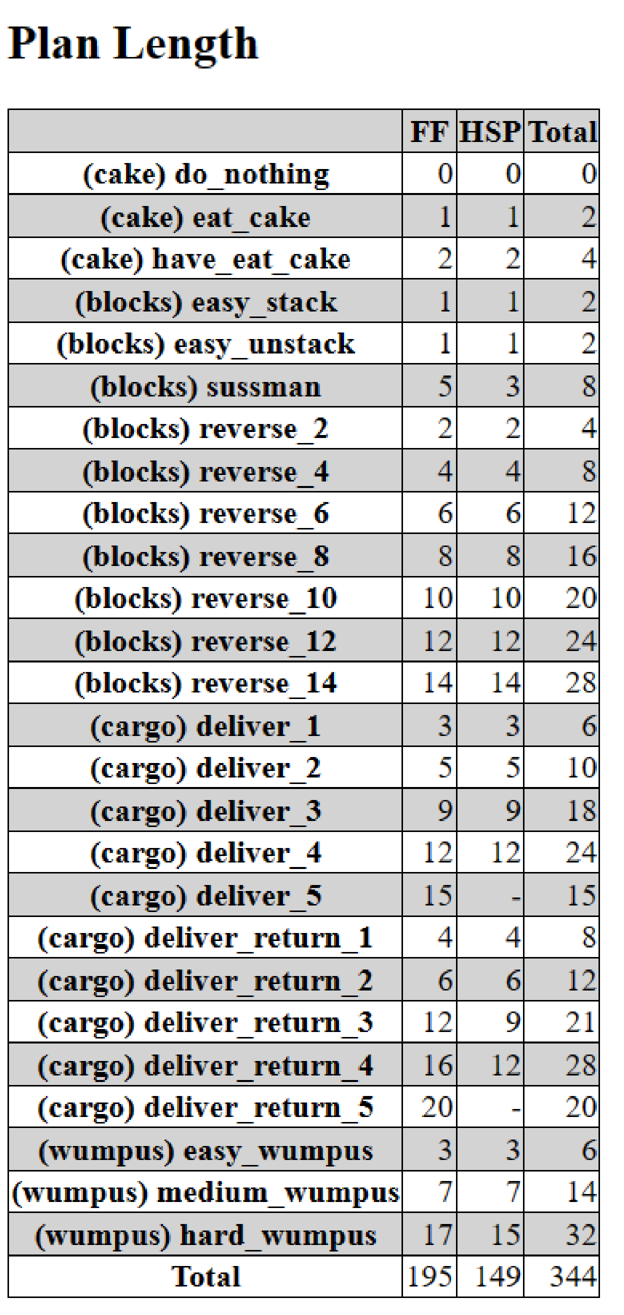
Plan Length

	FF	HSP	Total
(cake) do_nothing	0	0	0
(cake) eat_cake	1	1	2
(cake) have_eat_cake	2	2	4
(blocks) easy_stack	1	1	2
(blocks) easy_unstack	1	1	2
(blocks) sussman	5	3	8
(blocks) reverse_2	2	2	4
(blocks) reverse_4	4	4	8
(blocks) reverse_6	6	6	12
(blocks) reverse_8	8	8	16
(blocks) reverse_10	10	10	20
(blocks) reverse_12	12	12	24
(blocks) reverse_14	14	14	28
(cargo) deliver_1	3	3	6
(cargo) deliver_2	5	5	10
(cargo) deliver_3	9	9	18
(cargo) deliver_4	12	12	24
(cargo) deliver_5	15	-	15
(cargo) deliver_return_1	4	4	8
(cargo) deliver_return_2	6	6	12
(cargo) deliver_return_3	12	9	21
(cargo) deliver_return_4	16	12	28
(cargo) deliver_return_5	20	-	20
(wumpus) easy_wumpus	3	3	6
(wumpus) medium_wumpus	7	7	14
(wumpus) hard_wumpus	17	15	32
Total	195	149	344

FF

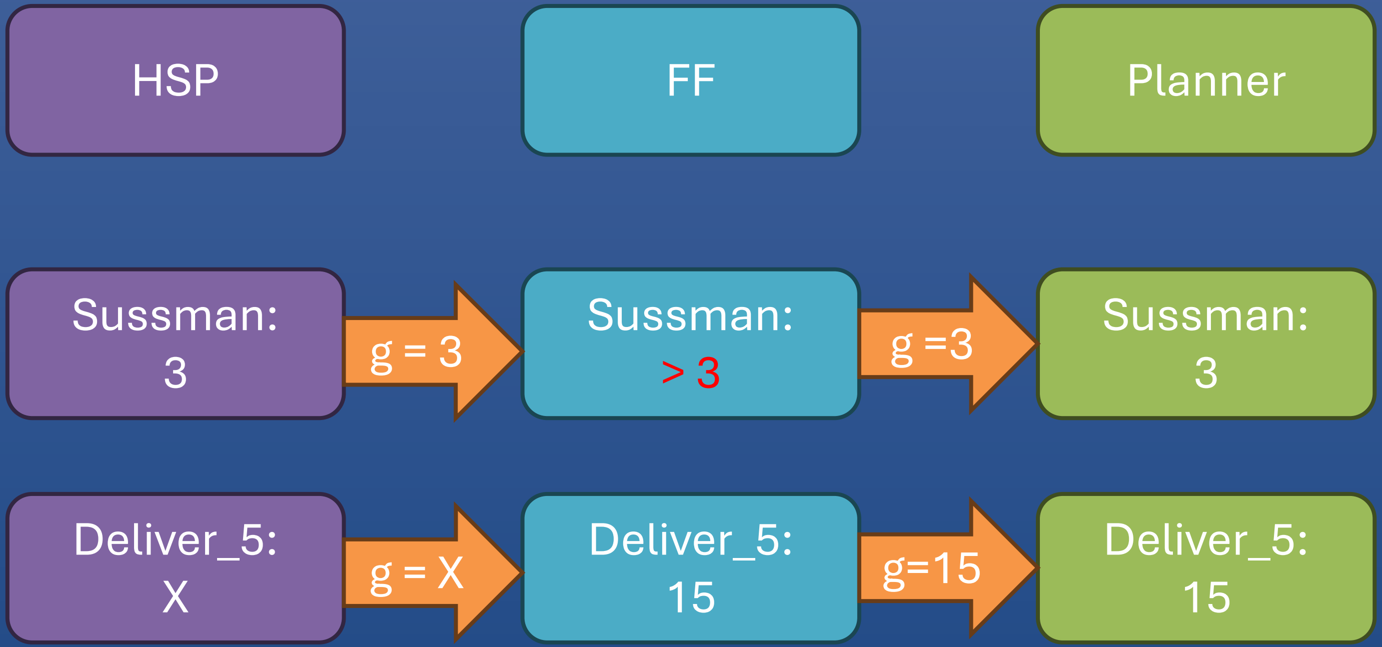
Solves

20



Plan Length

	FF	HSP	Total
(cake) do_nothing	0	0	0
(cake) eat_cake	1	1	2
(cake) have_eat_cake	2	2	4
(blocks) easy_stack	1	1	2
(blocks) easy_unstack	1	1	2
(blocks) sussman	5	3	8
(blocks) reverse_2	2	2	4
(blocks) reverse_4	4	4	8
(blocks) reverse_6	6	6	12
(blocks) reverse_8	8	8	16
(blocks) reverse_10	10	10	20
(blocks) reverse_12	12	12	24
(blocks) reverse_14	14	14	28
(cargo) deliver_1	3	3	6
(cargo) deliver_2	5	5	10
(cargo) deliver_3	9	9	18
(cargo) deliver_4	12	12	24
(cargo) deliver_5	15	-	15
(cargo) deliver_return_1	4	4	8
(cargo) deliver_return_2	6	6	12
(cargo) deliver_return_3	12	9	21
(cargo) deliver_return_4	16	12	28
(cargo) deliver_return_5	20	-	20
(wumpus) easy_wumpus	3	3	6
(wumpus) medium_wumpus	7	7	14
(wumpus) hard_wumpus	17	15	32
Total	195	149	344



HSP

FF

Planner

Sussman: 3

g = 3

Sussman: > 3

g =3

Sussman: 3

Deliver_5: X

g = X

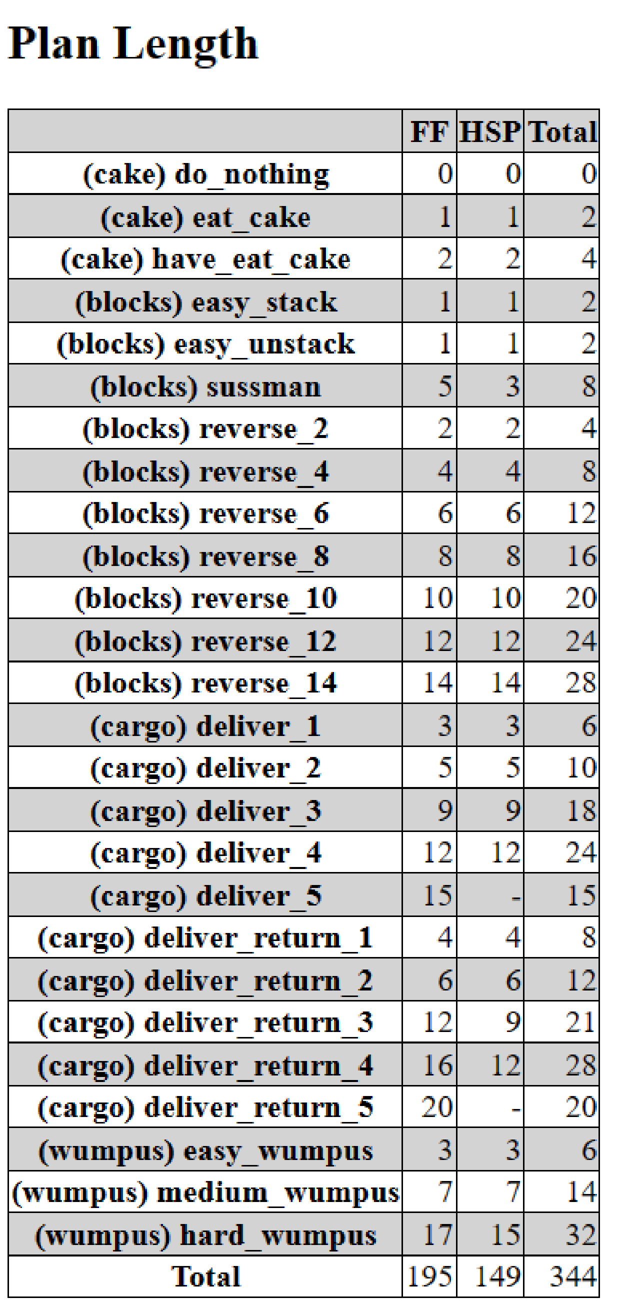
Deliver_5: 15

g=15

Deliver_5: 15

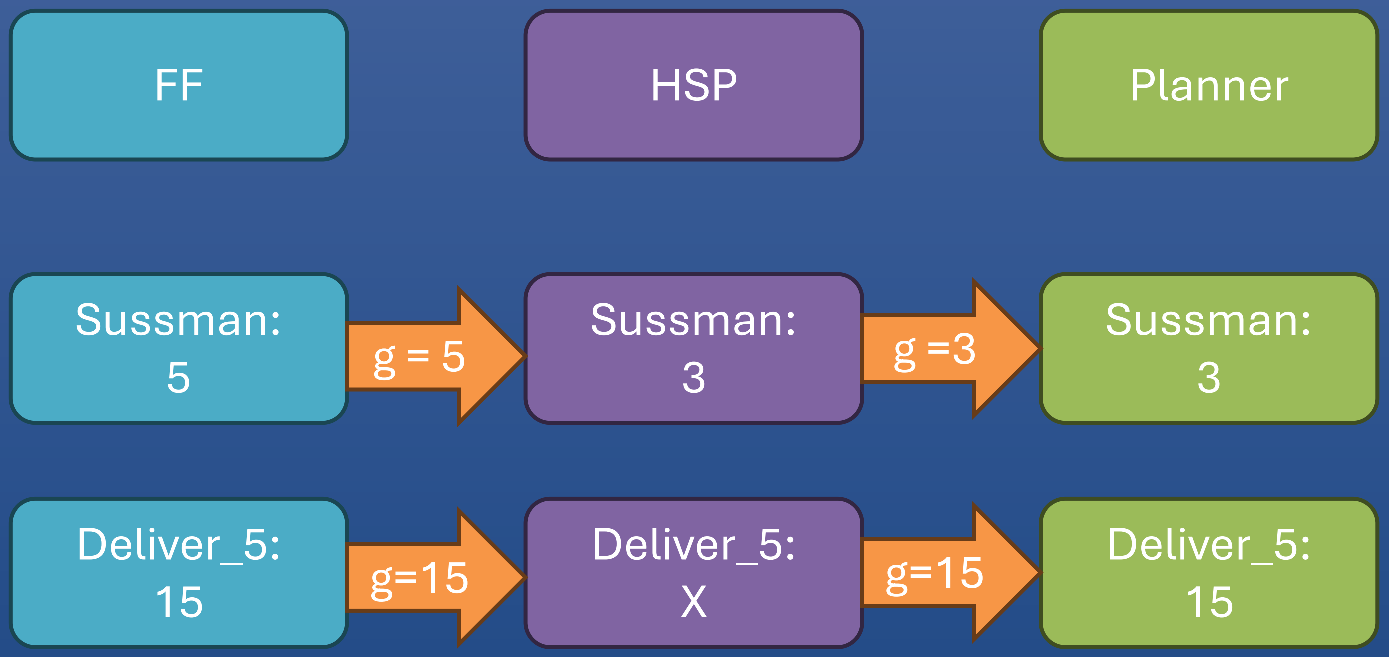
Planner Portfolio of HSP and Fast Forward solves all 26

(During planning, Fast Forward of Sussman would be pruned of all plans that exceed g)



Plan Length

	FF	HSP	Total
(cake) do_nothing	0	0	0
(cake) eat_cake	1	1	2
(cake) have_eat_cake	2	2	4
(blocks) easy_stack	1	1	2
(blocks) easy_unstack	1	1	2
(blocks) sussman	5	3	8
(blocks) reverse_2	2	2	4
(blocks) reverse_4	4	4	8
(blocks) reverse_6	6	6	12
(blocks) reverse_8	8	8	16
(blocks) reverse_10	10	10	20
(blocks) reverse_12	12	12	24
(blocks) reverse_14	14	14	28
(cargo) deliver_1	3	3	6
(cargo) deliver_2	5	5	10
(cargo) deliver_3	9	9	18
(cargo) deliver_4	12	12	24
(cargo) deliver_5	15	-	15
(cargo) deliver_return_1	4	4	8
(cargo) deliver_return_2	6	6	12
(cargo) deliver_return_3	12	9	21
(cargo) deliver_return_4	16	12	28
(cargo) deliver_return_5	20	-	20
(wumpus) easy_wumpus	3	3	6
(wumpus) medium_wumpus	7	7	14
(wumpus) hard_wumpus	17	15	32
Total	195	149	344



Planner Portfolio of Fast Forward and HSP solves all 26

Optimizing Planner Portfolio

- With Fast Downward, data transferring from planner to planner takes time away from allotted time of each planner
- Possible excess time utilization
- Ordering of planners in portfolio
- Need test problems
- Possibly more transfer of data

Sequential Planning Portfolios

- After Fast Downward:
  - No current single algorithm or heuristic dominates all others
  - If a planner does not solve quickly, problem will probably not be solved
- List each algorithm with a slice of the limited time and run each planner sequentially with given slice of overall time
- The sum of the algorithms beats each individual algorithm
- Satisficing and Optimizing
- Much more optimizing is possible