Evaluation Strategies for Planning as Satisfiability

Jussi Rintanen

Albert-Ludwigs-Universität Freiburg, Germany

August 26, ECAI'04

Introduction

- We consider evaluation strategies for satisfiability planning: find a (not necessarily shortest) plan. Trade-off: quality vs. cost to produce.
- Application domain: any approach to planning in which basic step is finding a plan of a given length, like planning as satisfiability, by CSP, by MILP, Graphplan, ...
- Significance: speed-ups of 0, 1, 2, 3, 4, ... orders of magnitude in comparison to the standard sequential evaluation strategy (as used in Graphplan, BLACKBOX, ...)

Strengths of satisfiability planning (SATP)

Satisfiability planning (Kautz & Selman, 1992/96) is an efficient approach for solving inherently difficult planning problems:

- optimal solutions to otherwise easy problems (Most of the standard planning benchmarks are solvable non-optimally by simple poly-time algorithms!!!)
- hard problems in the phase transition region [Rintanen, KR’04]
- combinatorially difficult planning problems

SATP vs. heuristic state-space planning

Heuristic state-space search [Bonet & Geffner 2000] has been considered stronger than SATP on many non-optimal planning problems, but

- apples vs. oranges: SATP planners give optimality guarantees but planners like HSP do not, and
- nobody has used SATP planners for non-optimal planning.

Open question

How efficient SATP actually is when optimality is not required?

SATP for non-optimal planning

Goal Non-optimal planning: relax all optimality requirements, any plan will do!

Consequence SATP becomes extremely good on standard big-and-easy benchmarks.

Disclaimer Problems that are very easy and very big likely remain to be solved by more specialized planning techniques:
After all, SAT solvers are general-purpose problem solvers and cannot be as efficient as more specialized techniques on all types of problems.

The standard sequential evaluation algorithm

Formula ϕ_j represents the question *Is there a plan of length j?*

PROCEDURE AlgorithmS()
i := 0;
REPEAT
    test satisfiability of ϕ_i;
    IF ϕ_i is satisfiable *THEN* terminate;
    i := $i + 1$;
UNTIL 1=0;

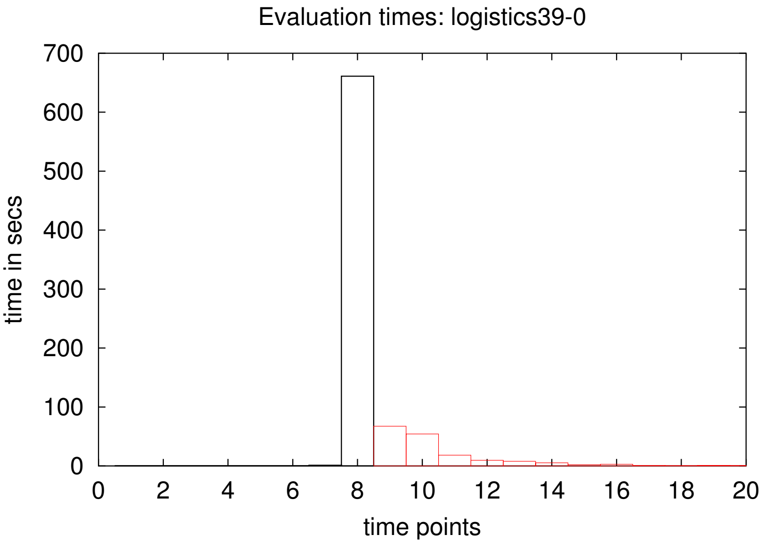
Problem

This algorithm proves that the plan has optimal length!!!

Experimentation

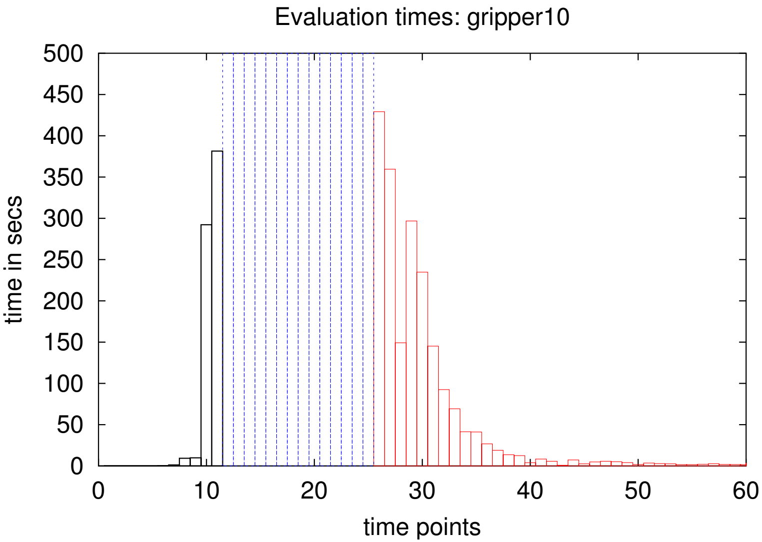
- How do runtime profiles of different benchmarks look like?
  1. benchmarks from planning competitions 1998, 2000, 2002
  2. samples from the set of all instances [Rintanen KR'04]
- Tests were run with Siege SAT solver version 4 (by Lawrence Ryan of University of Washington and Synopsys).
  This is one of the best SAT solvers for planning problems.

Examples

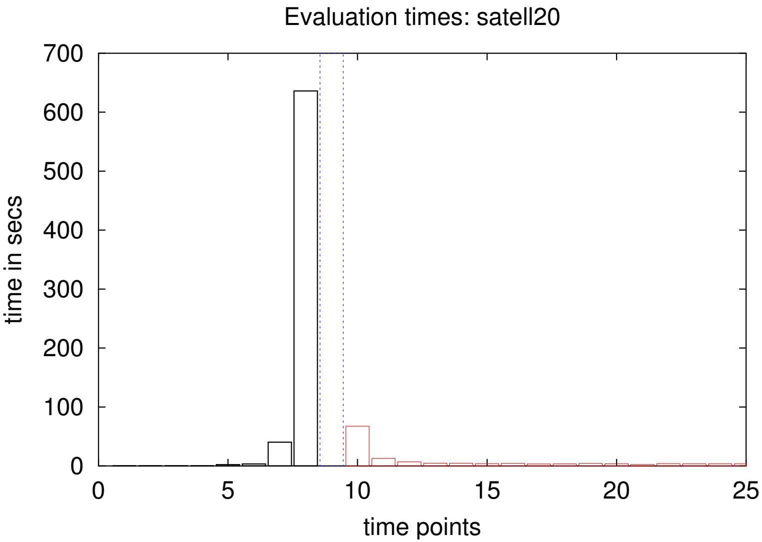


Evaluation times: logistics39-0
time in secs
time points
0
100
200
300
400
500
600
700
0
2
4
6
8
10
12
14
16
18
20

Examples

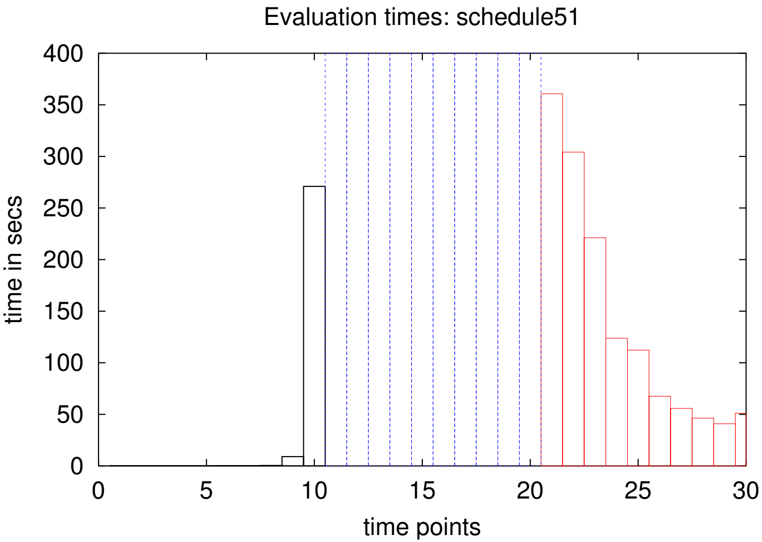

Evaluation times: gripper10
time in secs
500
450
400
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60
time points

Examples

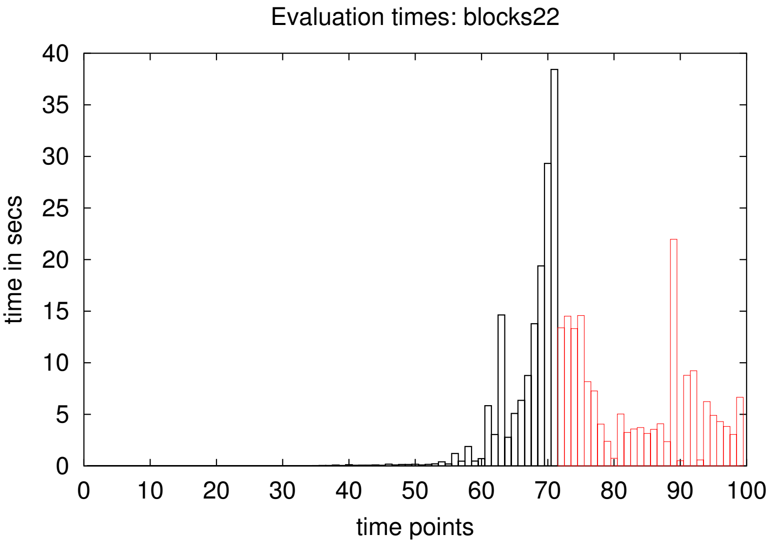


Evaluation times: satell20
time in secs
0
100
200
300
400
500
600
700
0
5
10
15
20
25
time points

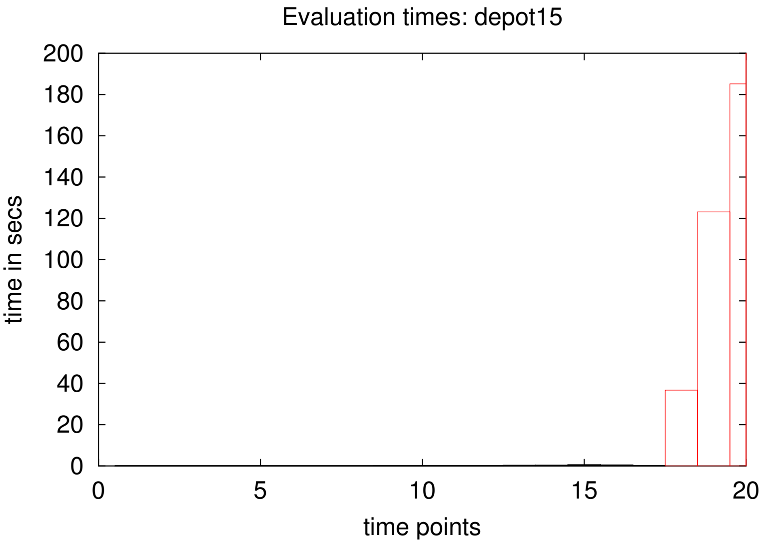
Examples


Evaluation times: schedule51
time in secs
0
50
100
150
200
250
300
350
400
time points
0
5
10
15
20
25
30

Examples


Evaluation times: blocks22
time in secs
time points
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50
60
70
80
90
100

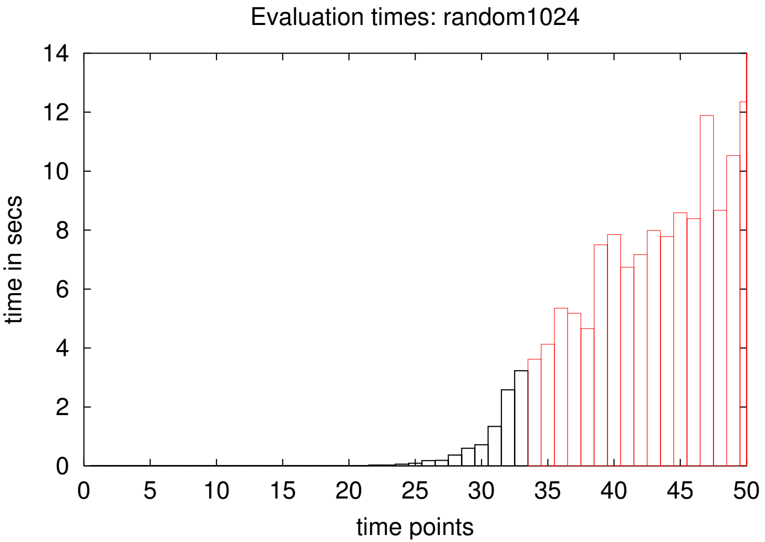
Examples


Evaluation times: depot15
time in secs
200
180
160
140
120
100
80
60
40
20
0
0
5
10
15
20
time points

Difficult problems with 20 state variables

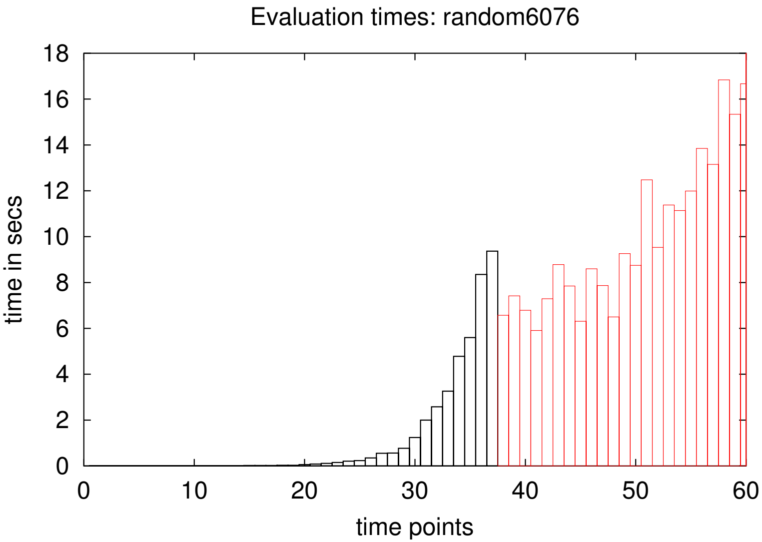
- Sampled from the space of all problems instances with 20 state variables, 40 or 42 STRIPS operators each having 3 precondition literals and 2 effect literals.
- This is in the phase transition region [Rintanen, KR'04].
- We show here some of the most difficult instances.
- Easier instances are solved (by satisfiability planners) in milliseconds.

Examples

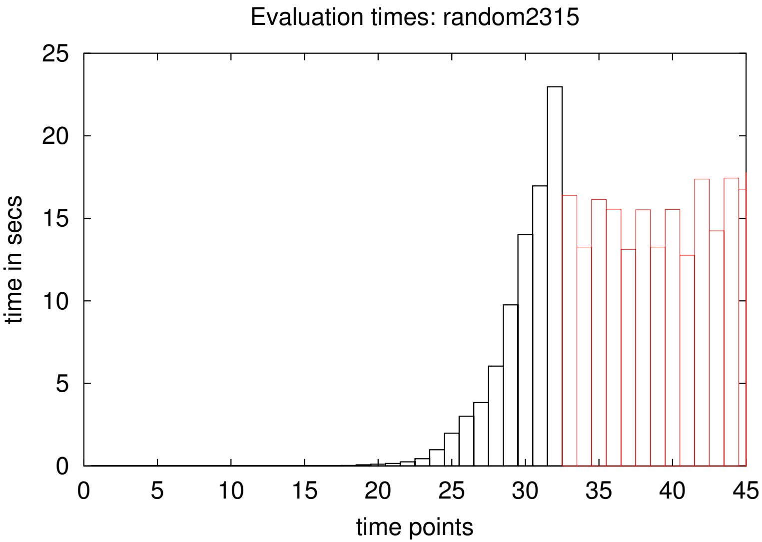


Evaluation times: random1024
time in secs
time points
0
2
4
6
8
10
12
14
0
5
10
15
20
25
30
35
40
45
50

Examples

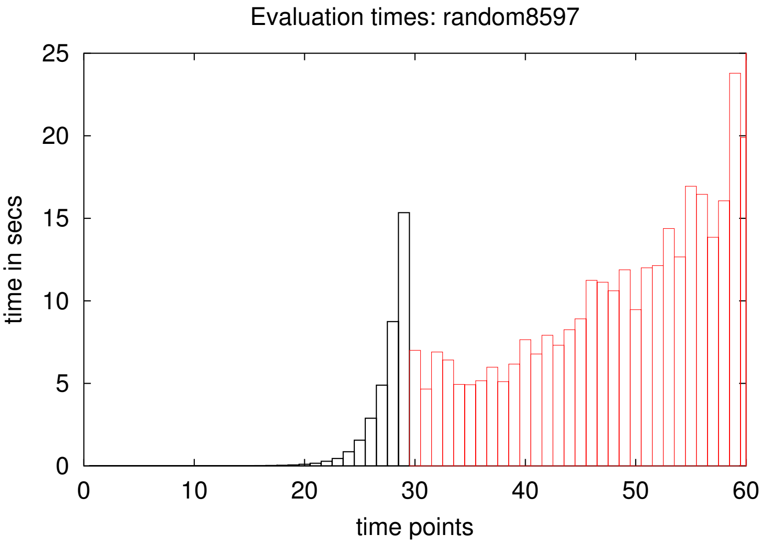

Evaluation times: random6076
time in secs
time points
0
2
4
6
8
10
12
14
16
18
0
10
20
30
40
50
60

Examples



Evaluation times: random2315
time in secs
0
5
10
15
20
25
time points
0
5
10
15
20
25
30
35
40
45

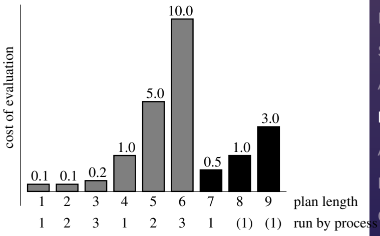
Examples



Introduction
SAT planning
Algorithm S
Experimentation
Algorithms
Experiments
Conclusion

The important insight

- Characteristic shape:
- Most of the difficulty is in the last unsatisfiable formulae.
- Devise evaluation strategies that get to evaluate the easier satisfiable formulae early!!



Algorithm A

- n processes: evaluate n plan lengths simultaneously (starting from lengths 0 to $n-1$)
- When a process finishes one length, in continues with the first unallocated one.
- Special case $n = 1$ is Algorithm S.

Algorithm B

- Evaluate all plan lengths simultaneously at different rates.
- If rate of length n is r, evaluate length $n+1$ at rate γr. γ is a constant $0 < \gamma < 1$.
- The CPU times allocated to the formulae form a geometric sequence

$$t\gamma^0, t\gamma^1, t\gamma^2, \ldots$$

with a finite sum

$$\frac{t}{1-\gamma}.$$

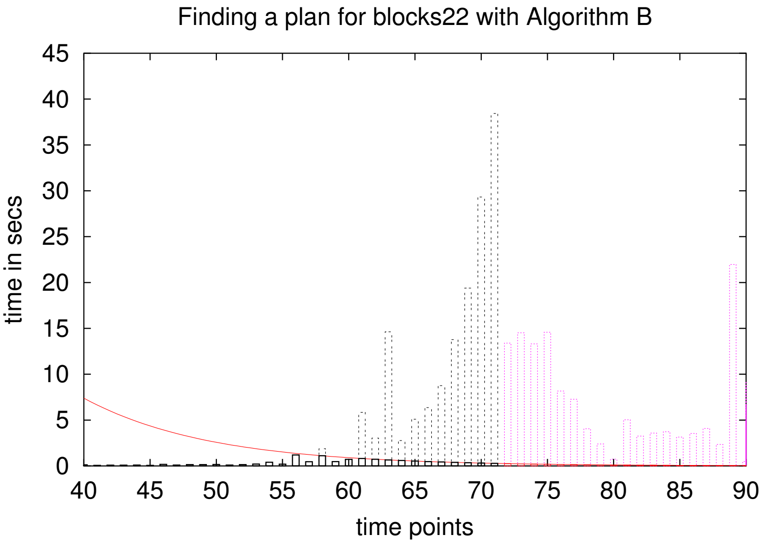
Properties of Algorithm B

- The first unfinished formula gets $1 - \gamma$ of the CPU. With $\gamma = 0.9$ this is $\frac{1}{10}$, with $\gamma = 0.5$ it is $\frac{1}{2}$.
- Speed-up is between $1 - \gamma$ and ∞.

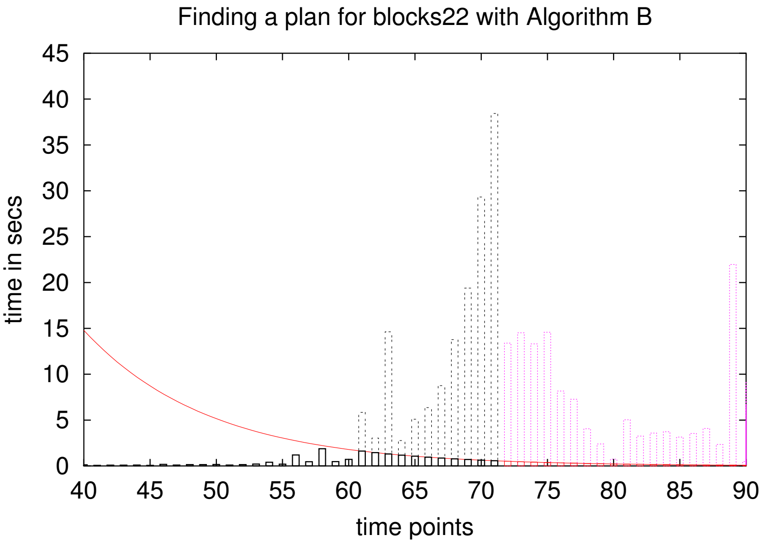
$$\text{Speed-up} = \frac{\text{runtime with Algorithm S}}{\text{runtime with Algorithm B}}$$

Worst-case slow-down only a constant factor!
Speed-up can be arbitrarily high!!

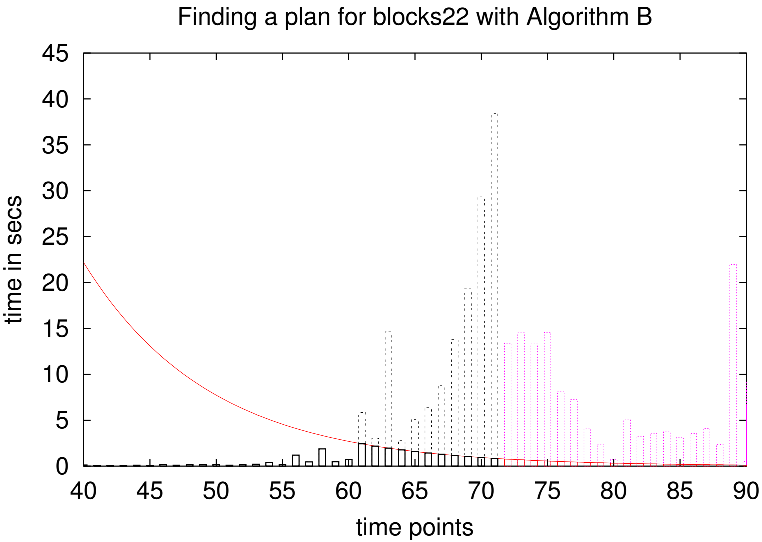
Algorithm B with $\gamma = 0.9$



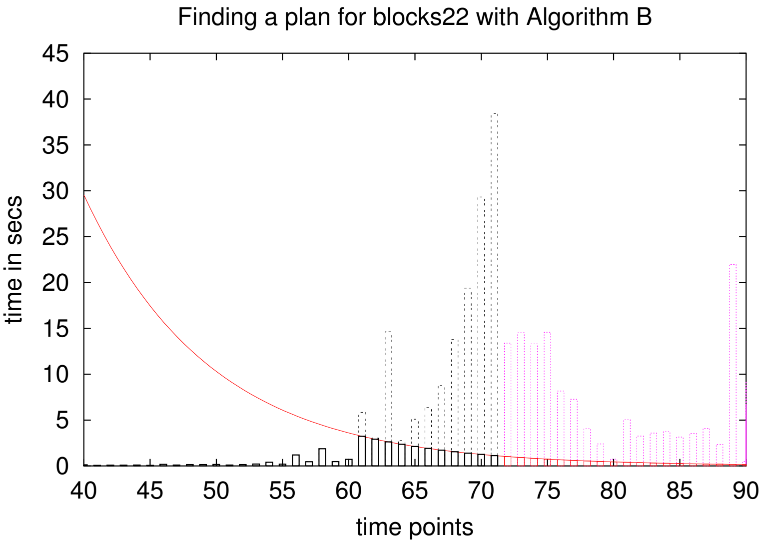
Algorithm B with $\gamma = 0.9$



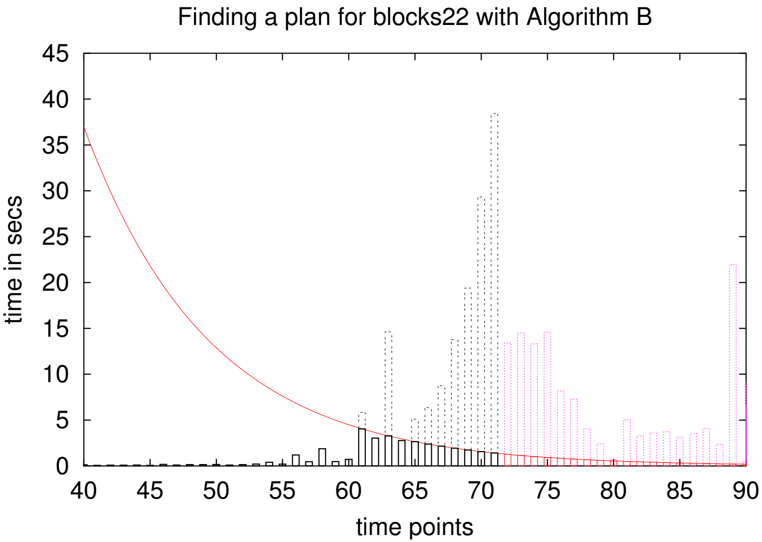
Algorithm B with $\gamma = 0.9$



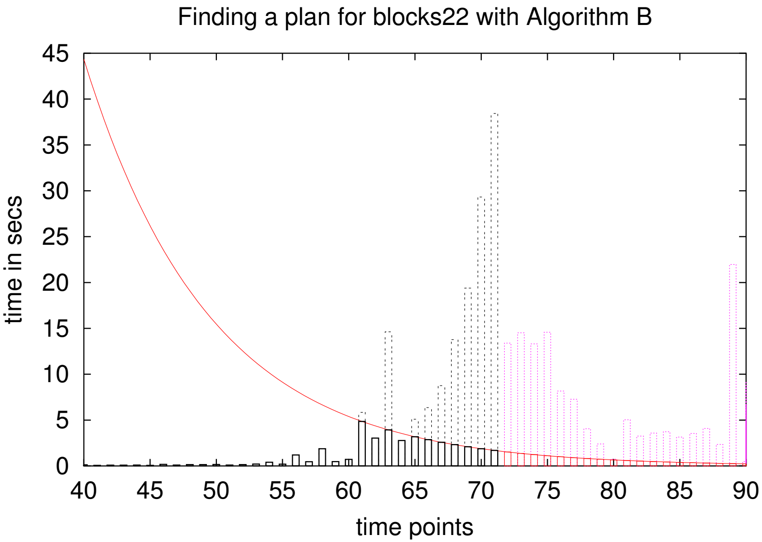
Algorithm B with $\gamma = 0.9$



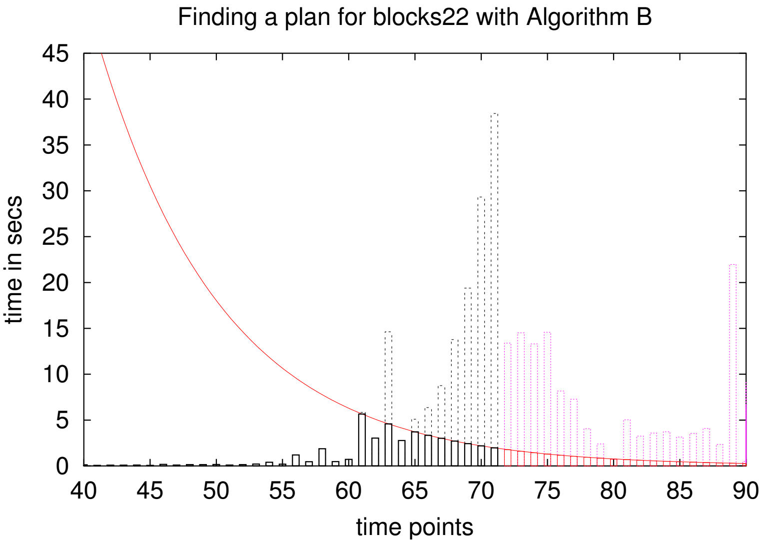
Algorithm B with $\gamma = 0.9$



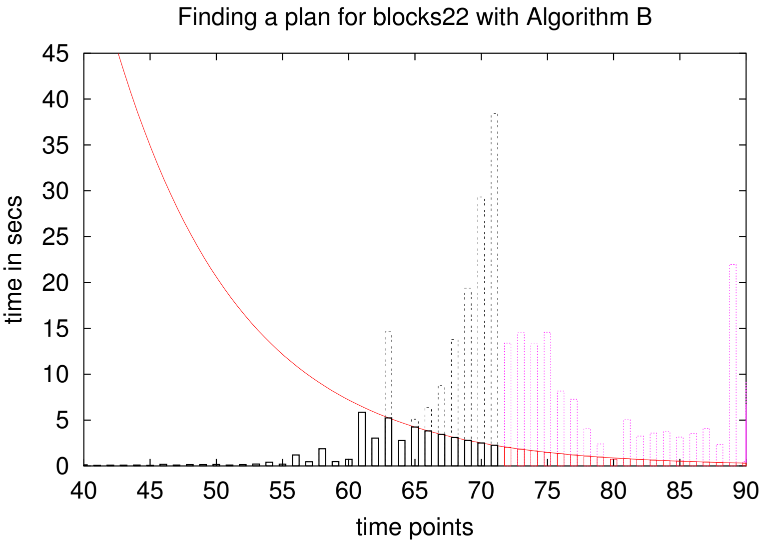
Algorithm B with $\gamma = 0.9$



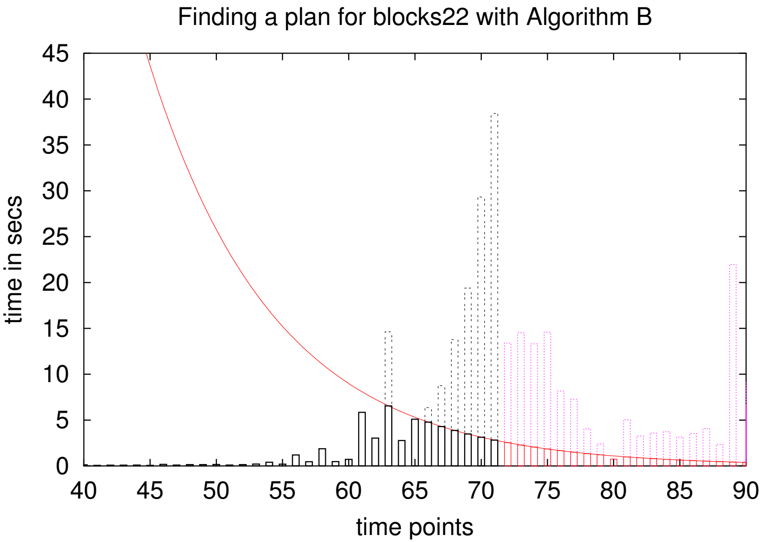
Algorithm B with $\gamma = 0.9$



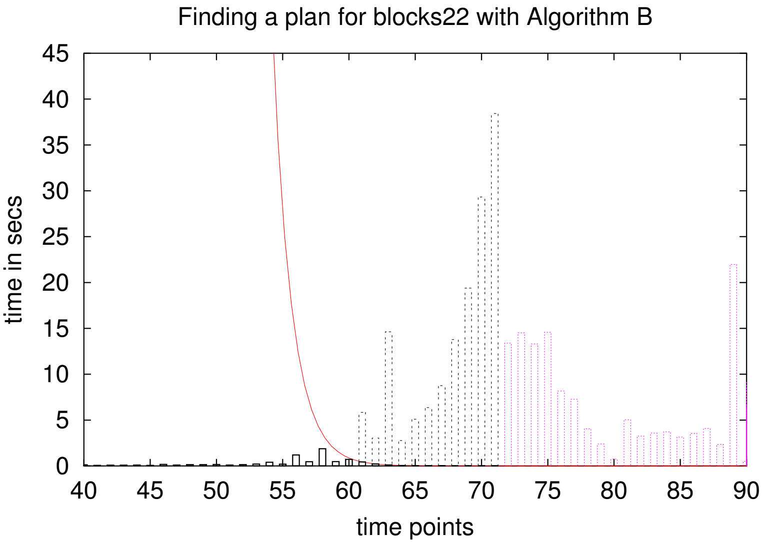
Algorithm B with $\gamma = 0.9$



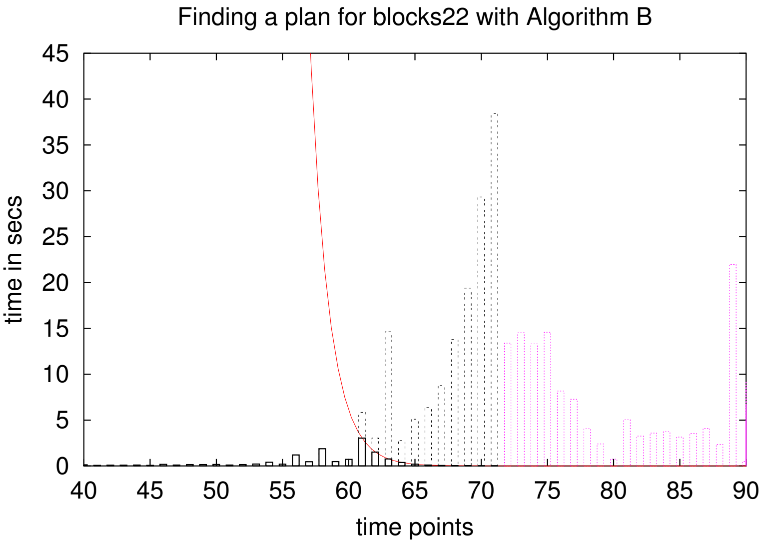
Algorithm B with $\gamma = 0.9$



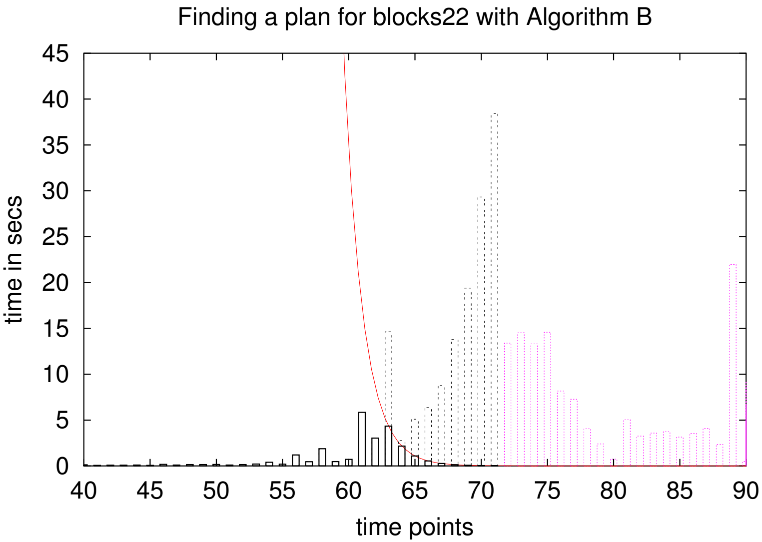
Algorithm B with $\gamma = 0.5$



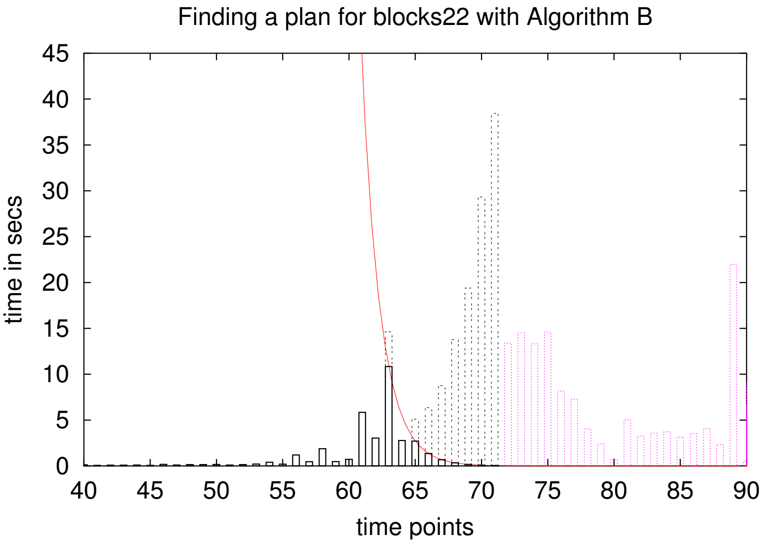
Algorithm B with $\gamma = 0.5$



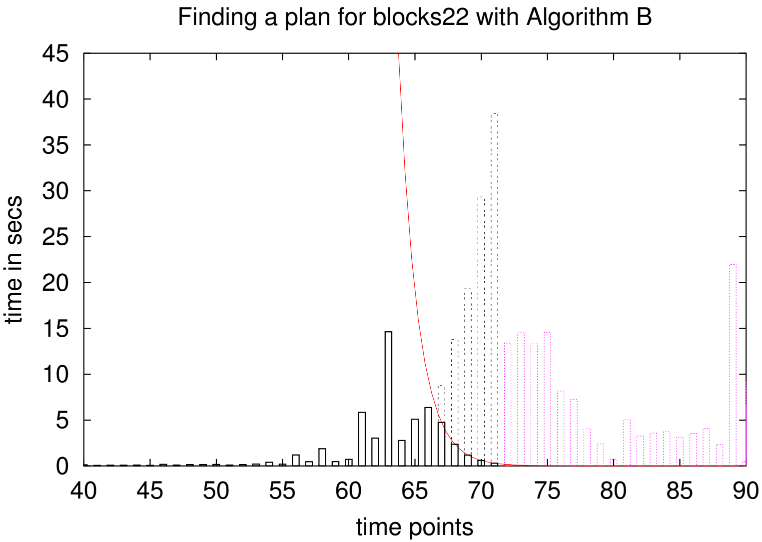
Algorithm B with $\gamma = 0.5$



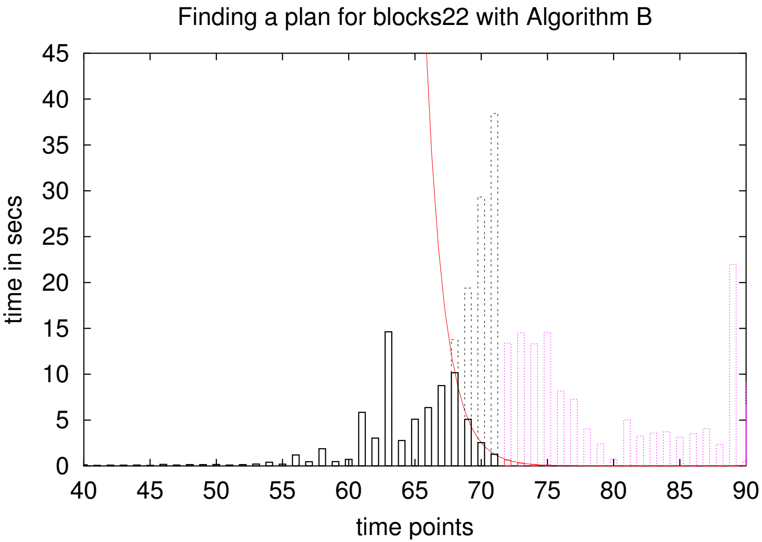
Algorithm B with $\gamma = 0.5$



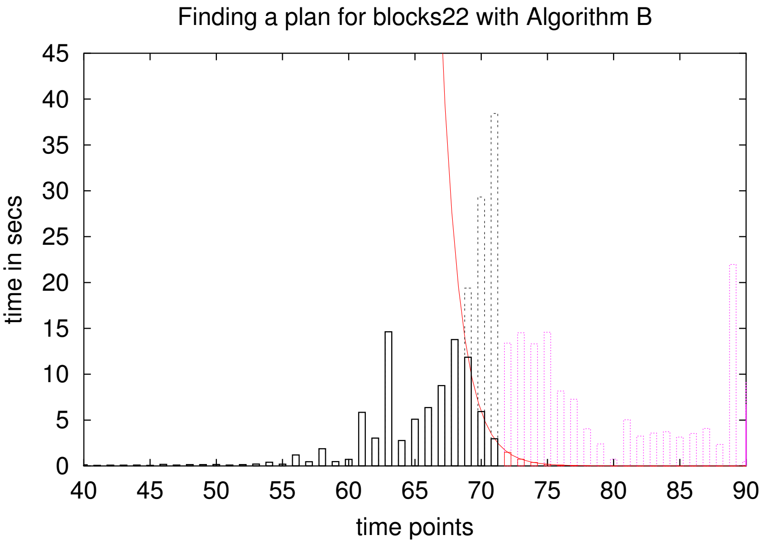
Algorithm B with $\gamma = 0.5$



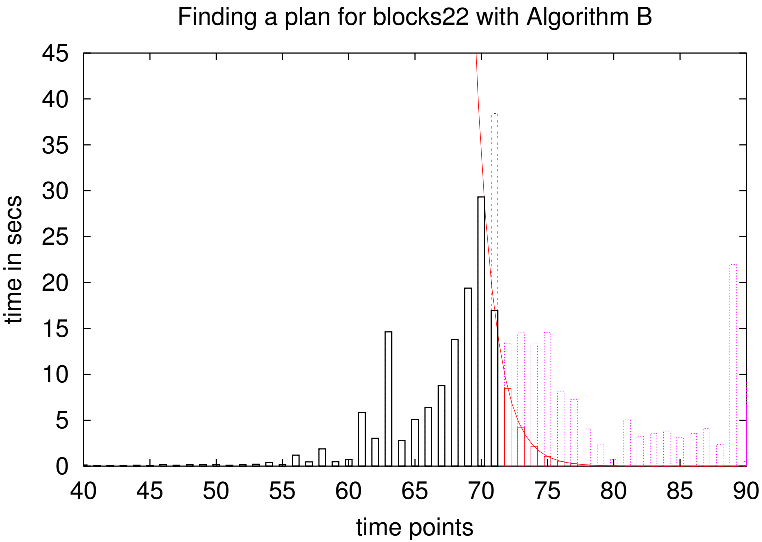
Algorithm B with $\gamma = 0.5$



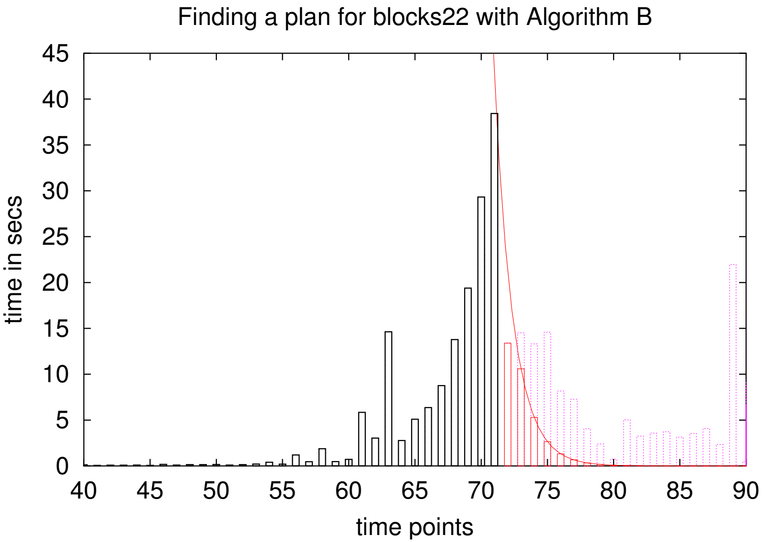
Algorithm B with $\gamma = 0.5$



Algorithm B with $\gamma = 0.5$



Algorithm B with $\gamma = 0.5$



instance	Algorithm A with n 1	2	4	8	16
logistics-39-0	-	-	54.2	8.7	5.4
logistics-39-1	-	564.9	84.2	15.6	5.3
logistics-40-0	1279.0	732.8	86.7	10.6	5.1
logistics-40-1	-	-	59.9	42.7	8.3
logistics-41-0	-	-	375.0	4.6	8.6
logistics-41-1	-	-	138.3	18.8	7.7

instance	Alg. S	Algorithm B with γ 0.500	0.750	0.875	0.938
logistics-39-0	-	136.4	17.2	9.5	10.1
logistics-39-1	-	86.2	11.6	7.8	8.9
logistics-40-0	1279.0	83.8	11.5	7.5	8.7
logistics-40-1	-	206.3	29.5	15.6	15.7
logistics-41-0	-	70.9	13.9	11.1	13.7
logistics-41-1	-	219.2	26.0	14.2	14.5

Efficiency on standard benchmarks

instance	Alg. S	Algorithm B with γ 0.500	0.750	0.875	0.938
blocks-22-0	150.1	163.0	99.9	53.4	40.9
blocks-24-0	2355.8	1822.8	390.1	171.2	95.0
blocks-26-0	-	4100.6	1919.6	547.1	243.0
blocks-28-0	-	2041.3	545.6	229.4	155.7
blocks-30-0	-	22777.6	3573.0	1462.2	900.2
blocks-32-0	-	$> 27h$	$> 27h$	7590.5	2637.2
blocks-34-0	219.4	231.0	238.5	246.3	236.4

Note

We can improve most of the runtimes on these slides to fractions by considering only e.g. plan lengths $0, 10, 20, 30, \ldots$.

Efficiency on standard benchmarks

instance	Alg. S	Algorithm B with γ 0.500	0.750	0.875	0.938
gripper-3	0.5	0.5	0.2	0.2	0.3
gripper-4	14.2	3.6	1.4	0.5	0.4
gripper-5	710.1	10.4	1.8	0.6	0.4
gripper-6	-	28.6	4.7	2.3	2.3
gripper-7	-	1600.4	82.6	10.8	3.8
gripper-8	-	9786.4	393.0	42.1	17.5
gripper-9	-	$> 27h$	2999.7	117.9	26.6
gripper-10	-	$> 27h$	12027.4	183.3	34.7
gripper-11	-	$> 27h$	3712.5	55.1	9.4
gripper-12	-	$> 27h$	43813.2	198.9	19.4
gripper-13	-	$> 27h$	$> 27h$	761.4	119.6
gripper-14	-	$> 27h$	$> 27h$	20949.6	892.3
gripper-15	-	$> 27h$	$> 27h$	3412.9	160.3

Efficiency on standard benchmarks

instance	Alg. S	Algorithm B with γ 0.500	0.750	0.875	0.938
sched-47-1	-	7153.6	370.5	113.2	92.5
sched-47-2	-	1512.2	100.0	51.2	54.8
sched-48-0	-	380.3	107.9	105.3	80.4
sched-48-1	-	252.0	50.9	25.9	27.7
sched-48-2	-	238.7	40.5	28.9	32.9
sched-49-0	-	29178.4	802.6	103.0	59.7
sched-49-1	-	22.2	13.9	17.1	26.6
sched-49-2	152.0	95.7	45.5	33.7	39.7
sched-50-0	140.1	27.8	14.5	13.5	14.8
sched-50-1	-	$> 27h$	4813.1	664.0	358.7
sched-50-2	-	104.3	35.1	27.5	32.4
sched-51-0	-	$> 27h$	2768.4	389.3	212.9
sched-51-1	-	30011.7	1033.0	209.6	144.5
sched-51-2	-	$> 27h$	4236.0	825.8	605.7

Efficiency on standard benchmarks

instance	Alg. S	Algorithm B with γ 0.500	0.750	0.875	0.938
driver-4-4-8	0.3	0.4	0.6	0.9	1.6
driver-5-5-10	805.4	754.0	304.0	284.4	376.4
driver-5-5-15	83.1	111.1	136.5	170.3	272.9
driver-5-5-20	667.1	103.8	92.7	134.1	230.3
driver-5-5-25	-	$> 27h$	24641.5	10817.7	10851.0
driver-8-6-25	-	$> 27h$	$> 27h$	17485.9	5429.7

Efficiency on standard benchmarks

instance	Alg. S	Algorithm B with γ 0.500	0.750	0.875	0.938
depot-09-5451	12.5	21.4	39.1	74.7	145.8
depot-10-7654	0.1	0.1	0.1	0.2	0.2
depot-11-8765	0.4	0.6	0.7	1.1	1.8
depot-12-9876	148.1	3.2	2.9	3.9	6.0
depot-13-5646	0.1	0.1	0.1	0.2	0.2
depot-14-7654	0.2	0.3	0.5	0.8	1.4
depot-15-4534	63.8	124.6	246.1	489.1	975.1
depot-16-4398	0.1	0.1	0.1	0.2	0.2
depot-17-6587	0.1	0.1	0.1	0.1	0.2
depot-18-1916	2.6	1.4	1.7	2.4	4.0
depot-19-6178	0.2	0.2	0.3	0.5	0.7
depot-20-7615	51.2	6.8	4.5	5.4	8.1
depot-21-8715	0.3	0.5	0.9	1.7	3.0
depot-22-1817	174.9	347.3	692.1	1381.8	2761.2

Conclusions

- Our work makes the trade-off between plan quality and planning difficulty in satisfiability planning explicit.
- Possibility of arbitrarily high performance gains is obtained by accepting the possibility of a small constant-factor slow-down and the loss of guarantees for plan optimality.
- A planner based on the new evaluation algorithms and new efficient encodings [Rintanen, Heljanko & Niemelä 2005] outperforms Kautz & Selman's BLACKBOX by ..,3,4,5,6,... orders of magnitude on many problems.