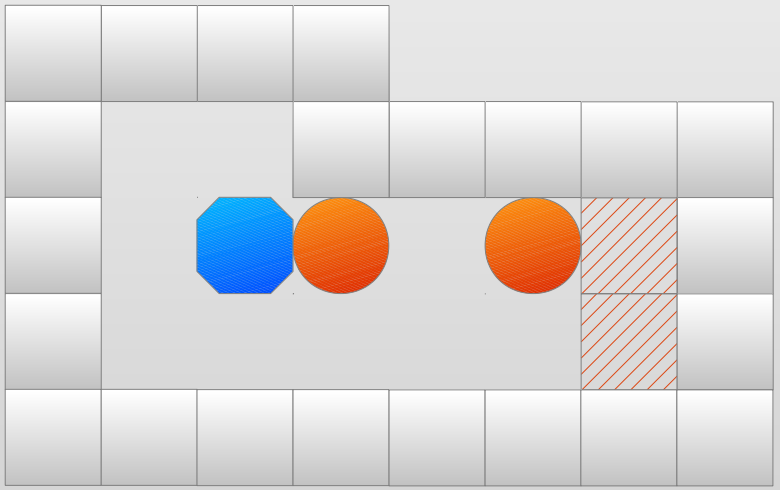
Search & Optimization

Sokoban: Enhancing general single-agent search methods using domain knowledge

Andreas Junghanns, Jonathan Schaeffer

Sokoban?

- Elements:
  - Player(blue)
  - Stones(orange)
  - Goals(shaded)
- Target:
  - Place stones on goals
  - Only push stones



Search-space properties

Property	Specifics	24-Puzzle	Rubik's Cube	Sokoban
Branching factor	Average Range	2.37 1-3	13.35 12-15	12 0-136
Solution length	Average Range	100+ 1-unknown	18 1-20	260 96-674
Search-space size	Upper bound	10^{25}	10^{19}	10^{98}
Calculation of lower bound	Full Incremental	O(n) O(1)	O(n) O(1)	$O(n^3)$ $O(n^2)$
Underlying graph		Undirected	Undirected	Directed

Why is it so difficult?

- Long solution lengths (97-674)
- Large branching factors (0-136)
- Mostly sequential solutions
- Hard to calculate good lower bound of solution length
- Deadlock states
- Application-independent planners have a lot of problems with this domain

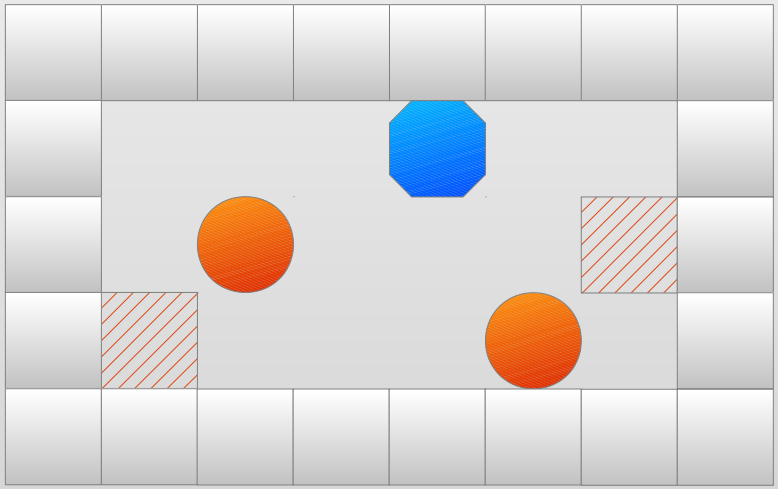
Setup / What was used

- IDA*
- Limit of 20 million nodes
- Runtime of 1 to 3h
- Machine with 195 MHz
- Work done over a period of 3 years
- 90 Sokoban problems
- Optimizations were incrementally added

Lower bound (naive)

static, domain, 0 problems solved

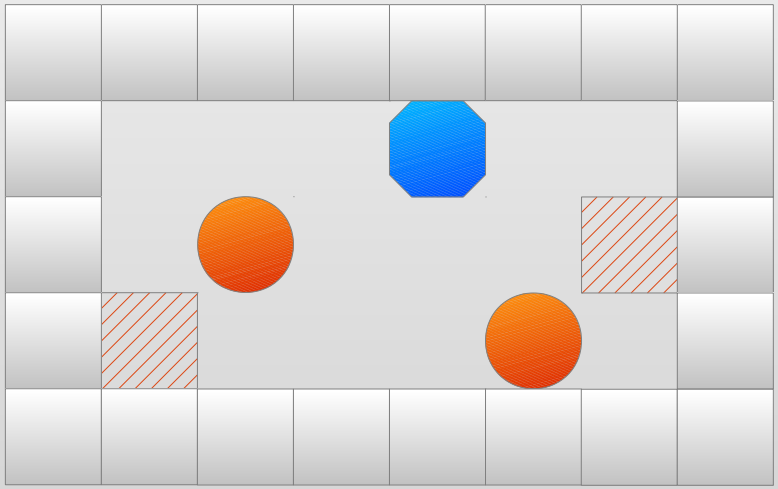
- Sum of distances of each stone to closest goal
- Much too low in general
- Example:
  - Lower bound: 4
  - Solution length: 8



Minimum matching lower bound

static, domain, *0 problems solved*

- Assigns each stone to a goal
- Sum of individual solution lengths of each stone to its assigned goal
- $O(N^3)$ for N stones
- Example:
  - MM lower bound: 8



Transposition table

dynamic, subtree, 5 problems solved

- Duplicate elimination
  - Skip already visited nodes before expansion
- Table with 2^{18} entries
- Presentation of Christian Mächler

Move ordering

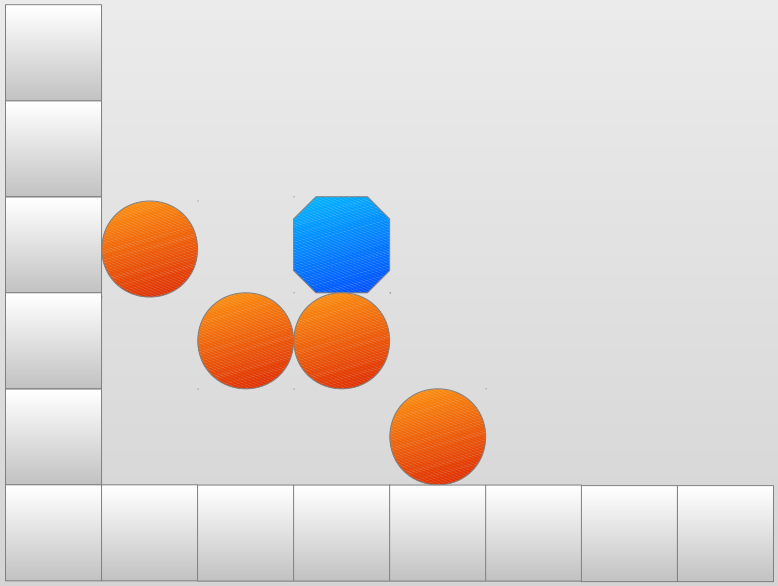
static, subtree, 4 problems solved

- Ordering of actions
- Most promising actions first
  - Inertia: use same stone first
  - Lower h-value
- Find solution earlier in iteration
  - Only in last iteration when using IDA*
  - Last iteration far bigger than the ones before

Deadlock table

dynamic, domain, 5 problems solved

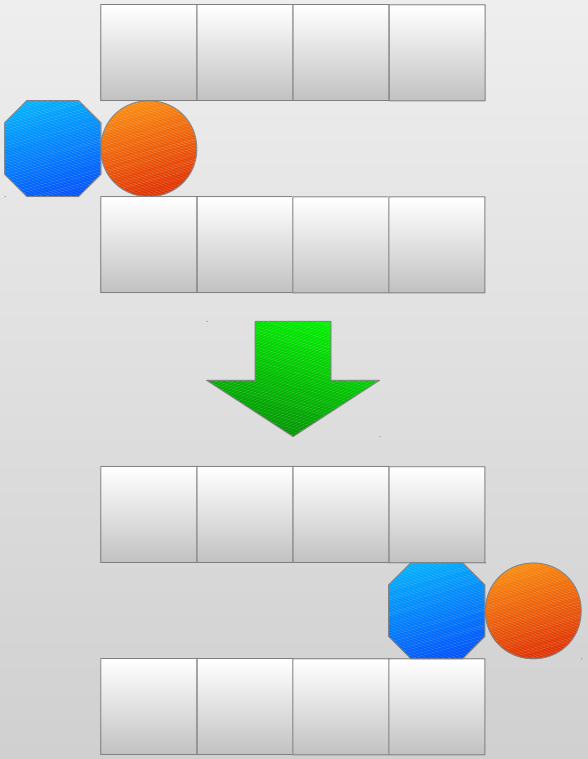
- 4x5 Region
- Store all possible placements of walls, stones and the player
- Store if a deadlock is present
- Check against table during search



Tunnel macros

static, instance, 6 problems solved

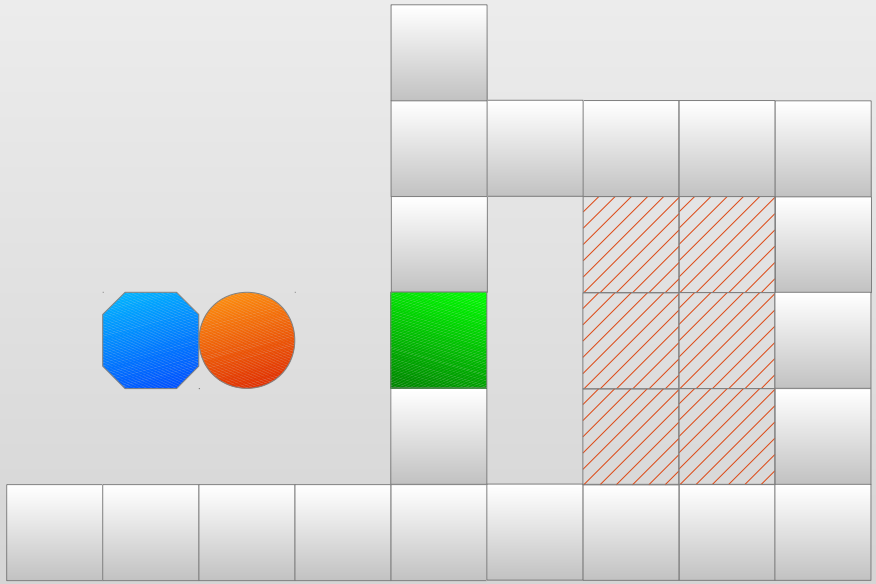
- Group related actions together
- No interleaving with other actions
- *Warning: Check for correctness and completeness*



Goal macros

dynamic, instance, 17 problems solved

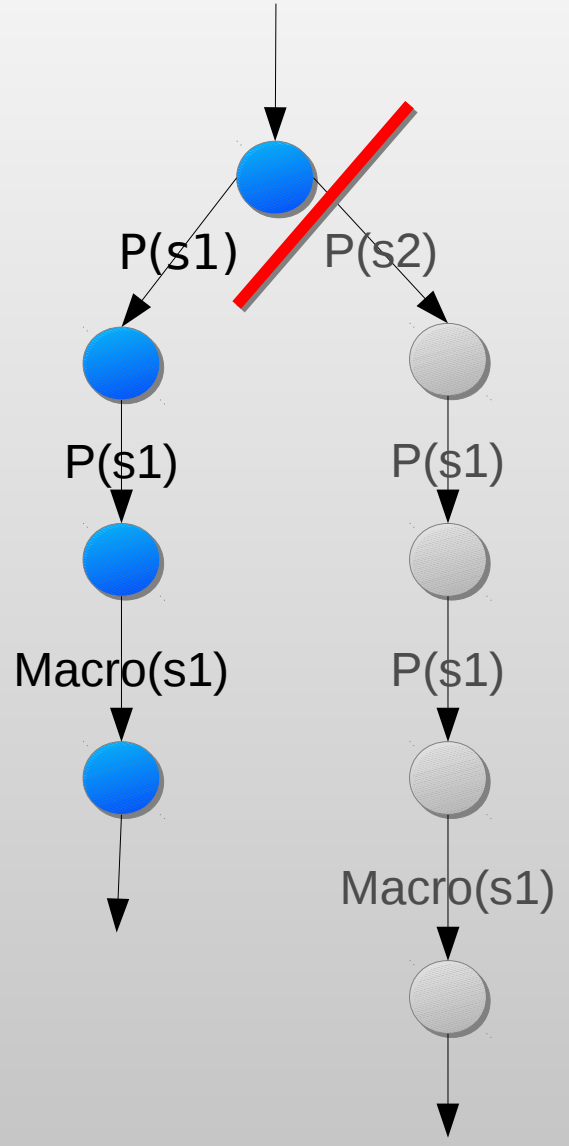
- Goals often grouped together
- Decompose problem:
  - Push stone to entrance (green)
  - Pack stones in goal areas
- Prevents interleaving with other actions



Goal cuts

static, subtree, 24 problems solved

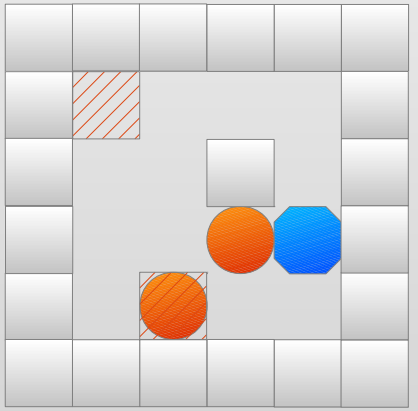
- Prevents interleaving with other actions
- Paths with goal macros at end



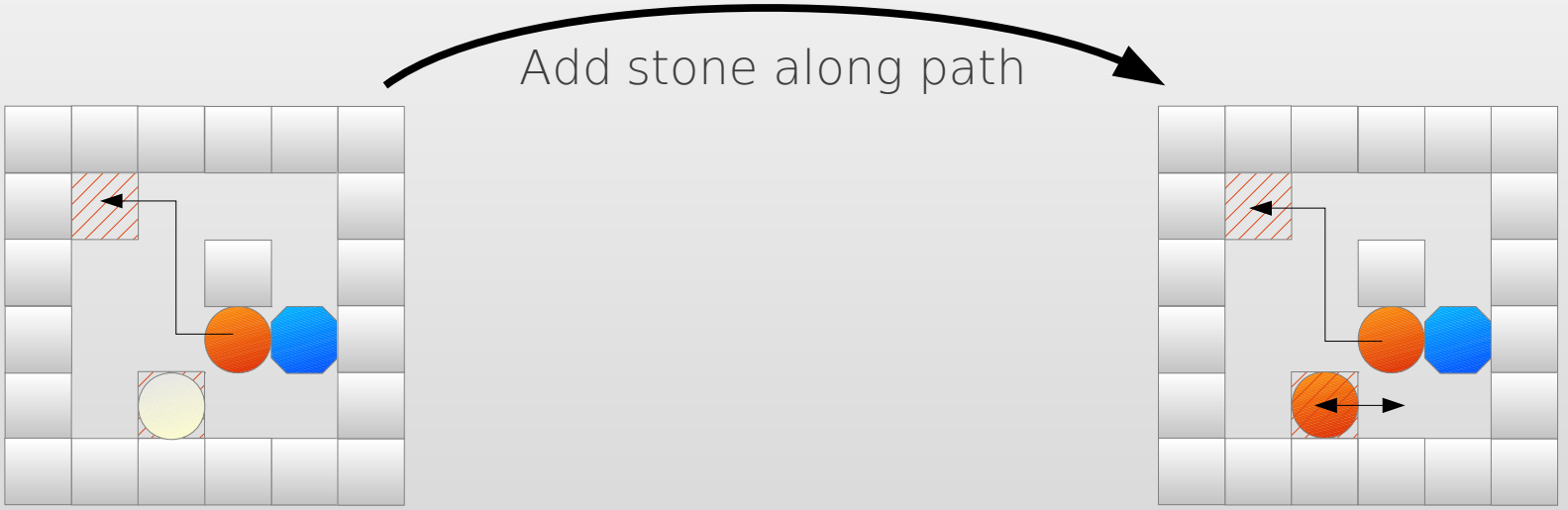
Pattern search

dynamic, subtree, 48 problems solved

- Idea:
  - Search for solution with current stone(s)
  - Add stones along the path
  - Repeat until
    - No solution = deadlock
    - No stones added = done
- Control function to detect promising cases
- Increase



Pattern search example



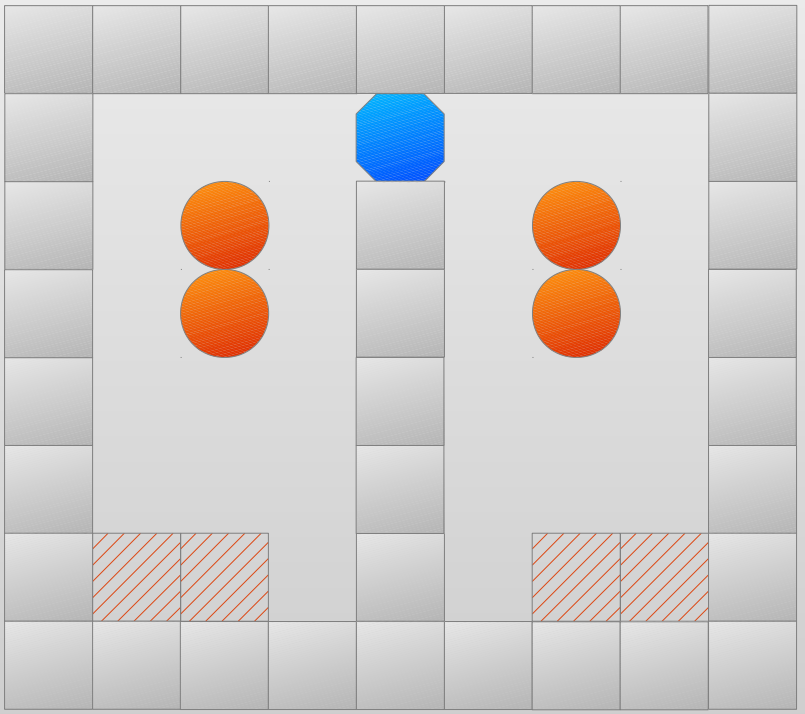
Search solution
(single stone)

Search solution
(length:6)

Relevance cuts

static, instance, 50 problems solved

- Prevent interleaving actions if not relevant to each other
- Measurement of relevance:
  - Alternatives
  - Goal-Skew
  - Connection
  - Tunnel



Overestimation

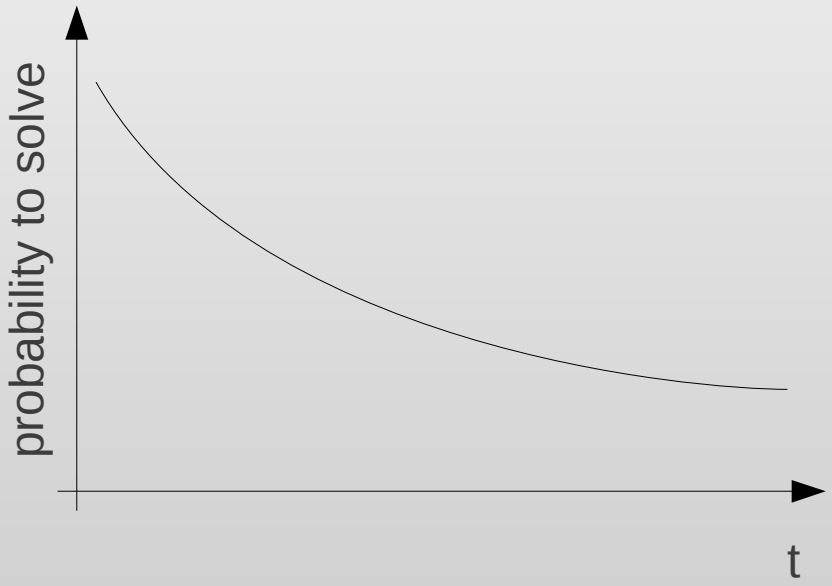
dynamic, subtree, 54 problems solved

- Better fit of h to h*
- Loose admissibility of heuristic for more accurate heuristic value
  - Solution no longer guaranteed to be optimal
- Use all penalties from pattern search (sum of max per stone)

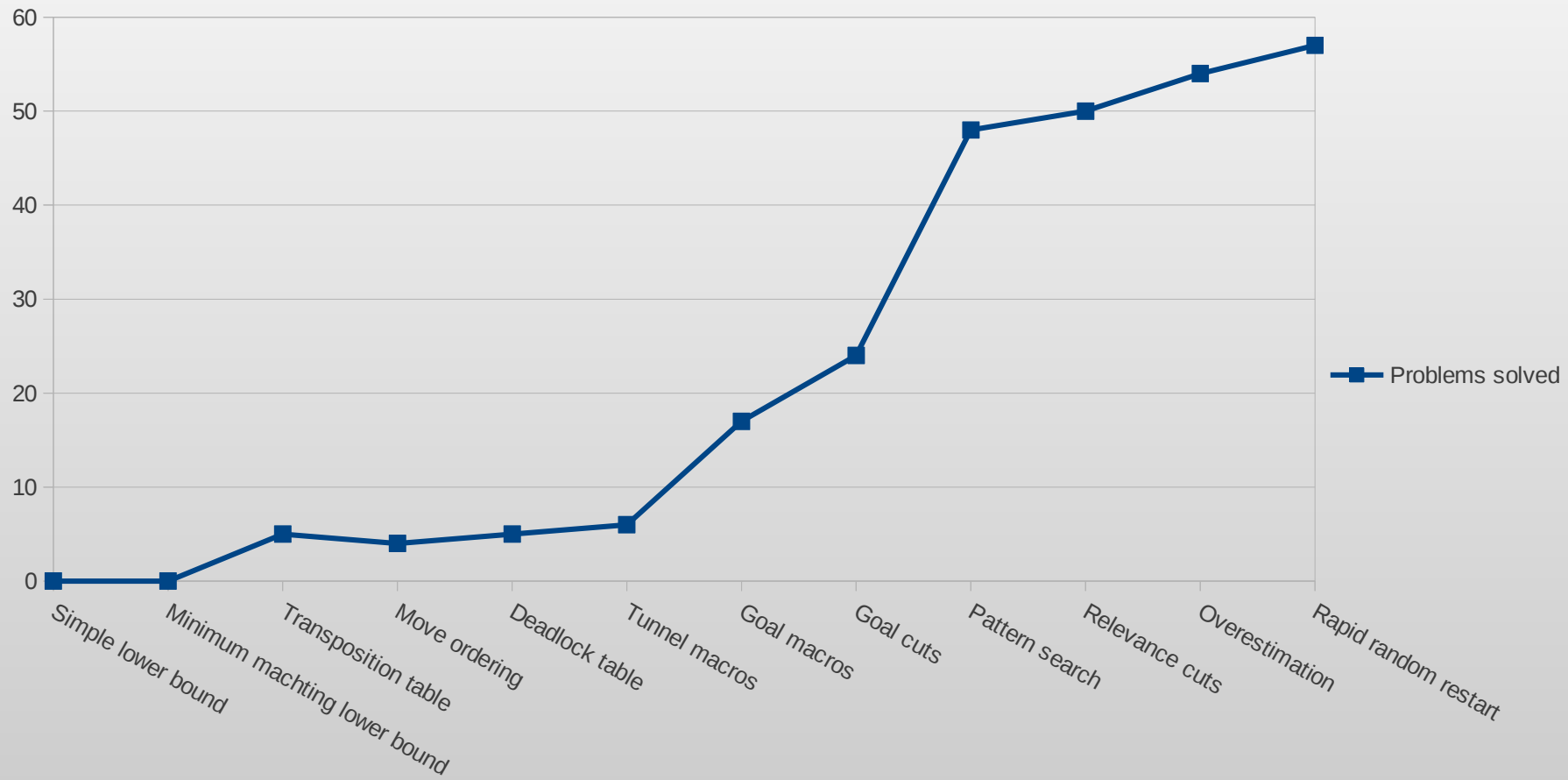
Rapid random restart

57 problems solved

- For problems with *heavy tails* (some instances very hard with bad parameters)
- Multiple searches with different parameter values



Results


60
50
40
30
20
10
0
Problems solved
Simple lower bound
Minimum machting lower bound
Transposition table
Move ordering
Deadlock table
Tunnel macros
Goal macros
Goal cuts
Pattern search
Relevance cuts
Overestimation
Rapid random restart

Thank you for your attention

any questions?