A Portfolio Solver for Answer Set Programming:

Preliminary Report

M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub,
M. Schneider, and S. Ziller

University of Potsdam

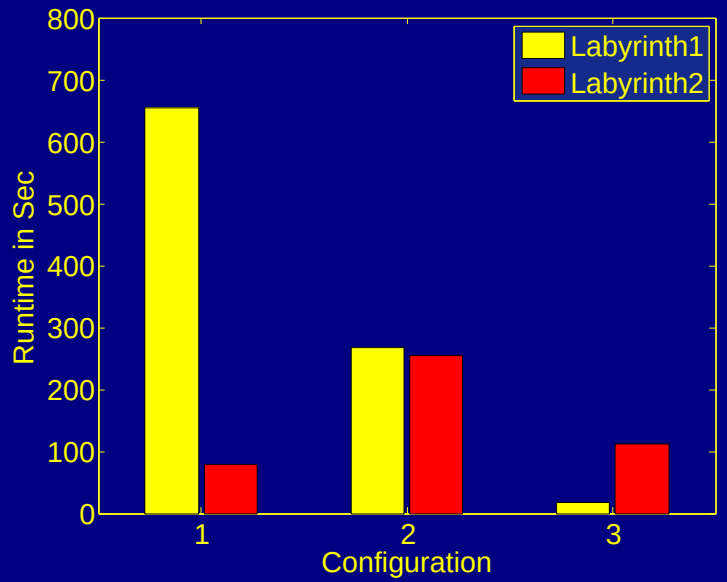
Outline

Outline

Motivation: Per-Instance Configuration

Problem (1): Problems are sensible to different solver configurations.
Question: How to decide automatically what is a good configuration for a given problem?

Problem (2): There is no best configuration for all kind of problems.
Solution: Per-instance configuration



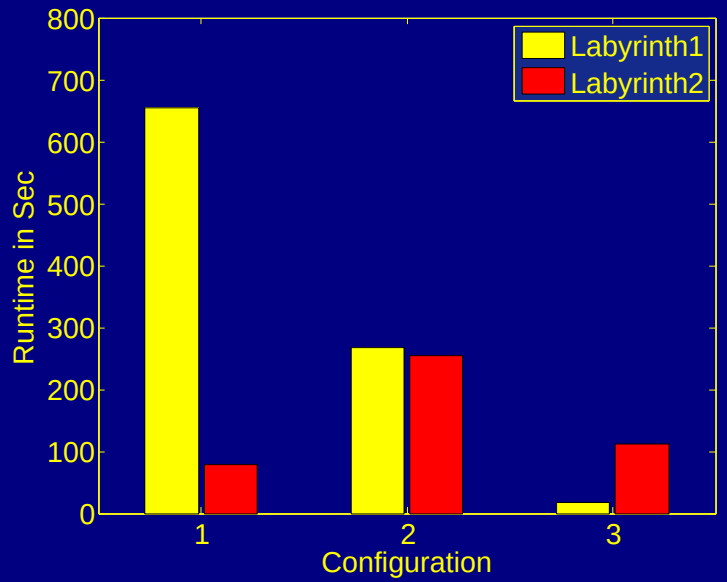
Motivation: Per-Instance Configuration

Problem (1): Problems are sensible to different solver configurations.

Question: How to decide automatically what is a good configuration for a given problem?

Problem (2): There is no best configuration for all kind of problems.

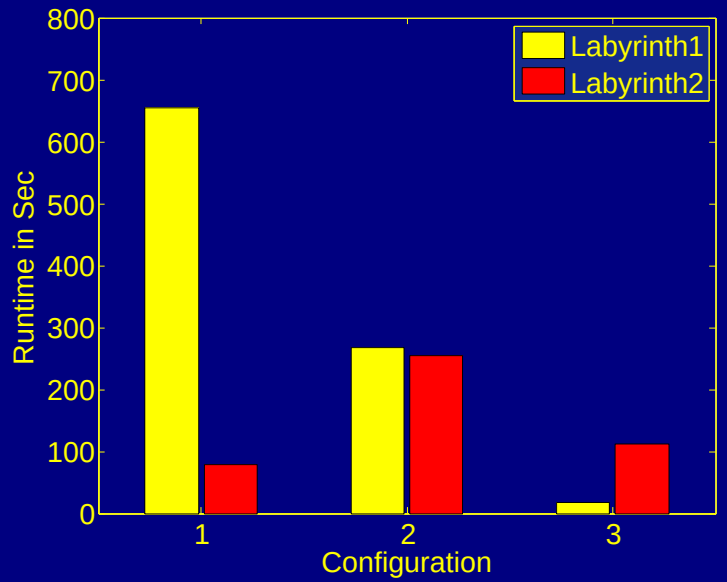
Solution: Per-instance configuration



Motivation: Per-Instance Configuration

Problem (1): Problems are sensible to different solver configurations.
Question: How to decide automatically what is a good configuration for a given problem?

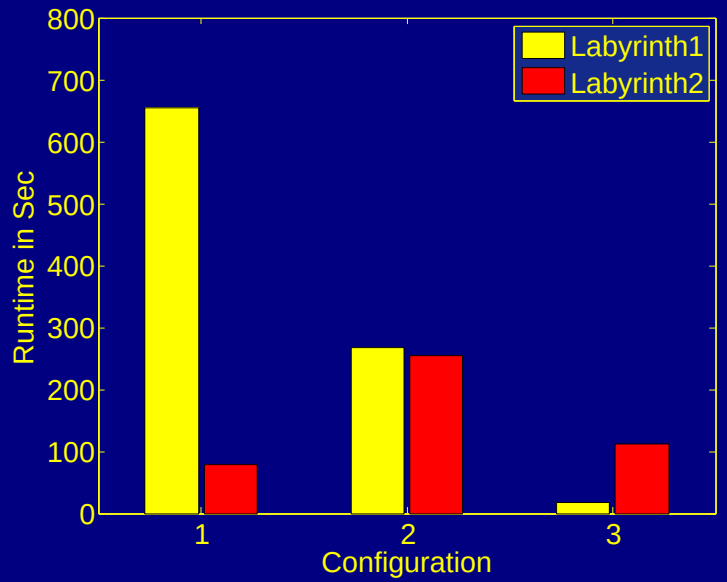
Problem (2): There is no best configuration for all kind of problems.
Solution: Per-instance configuration



Motivation: Per-Instance Configuration

Problem (1): Problems are sensible to different solver configurations.
Question: How to decide automatically what is a good configuration for a given problem?

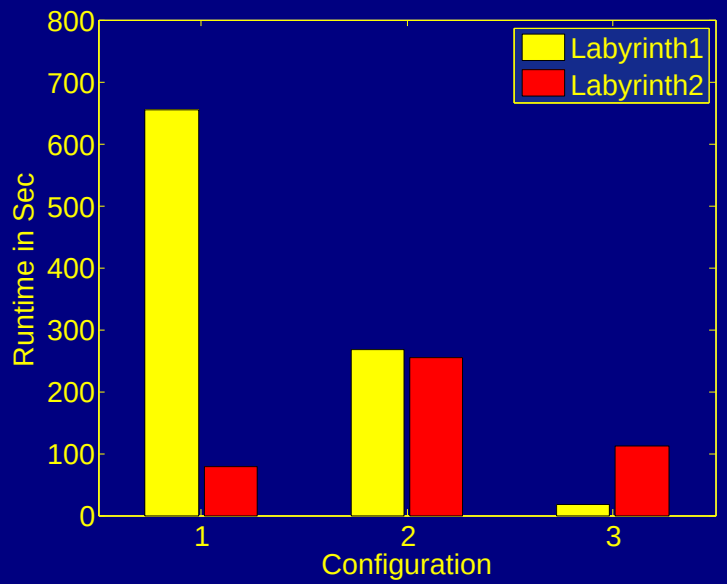
Problem (2): There is no best configuration for all kind of problems.
Solution: Per-instance configuration



Motivation: Per-Instance Configuration

Problem (1): Problems are sensible to different solver configurations.
Question: How to decide automatically what is a good configuration for a given problem?

Problem (2): There is no best configuration for all kind of problems.
Solution: Per-instance configuration



Outline

Features

Features of instances can be mapped to runtime

- Plain instance features:
  - Number of atoms
  - Number of different types of rules
  - ...
- Features after preprocessing
  - Tight?
  - Equivalence between atoms and bodys
  - Number of different types of constraints
  - ...
- Solving with *clasp* till 4 restarts were performed
- Search features after each restart:
  - Number of choices
  - Number of conflicts
  - Number of different types of learnt nogoods
  - Number of deleted nogoods
  - Average length of jumps
  - ...

All in all **84 features** will be calculated.

ML-Models

- To predict the quality of a configuration per instance a regression is used

$$\vec{f} \mapsto \mathbb{R}$$

- Examples of possible regression techniques:
  - Ridge Regression (SATzilla: [2])
  - Random Regression Forrest (SMAC: [4])
  - Support Vector Regression [6]
- Important for a good prediction are
  - Representative Training set
  - Features

Outline

Workflow: Gringo+Clasp

Figure: Architecture of gringo | clasp

Workflow: Gringo+Claspfolio

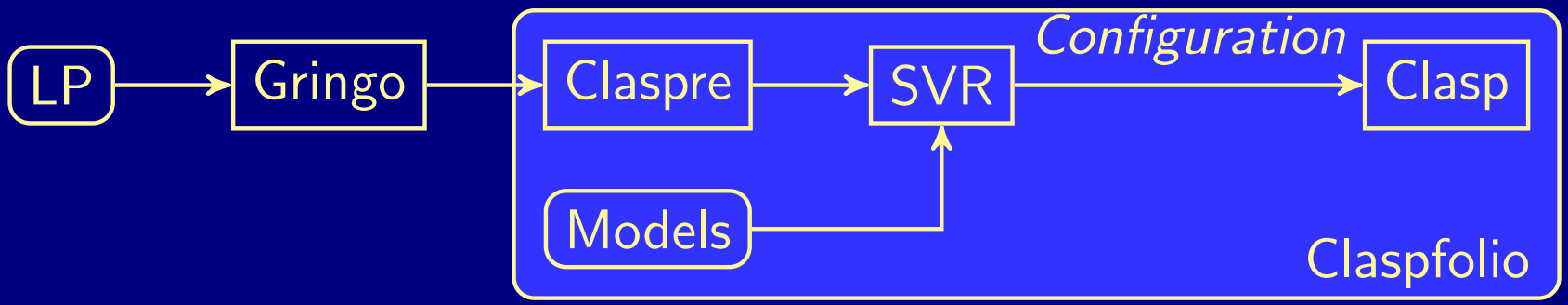


Figure: Architecture of claspfolio

[1] SVR from the libSVM-Package [7]

[2] Inspired by SATzilla [2]

Training and Run

Training:

1. Instance set I and portfolio of configurations C
2. Compute features $f(\vec{i})$ for all $i \in I$
3. Compute runtimes $t(i, c)$ if all tuples $(i, c) \in I \times C$
4. Train a regression model $s_c : \vec{f} \mapsto \mathbb{R}$ for each configuration

$$s_c(i) = \frac{\min_j (t(i, j))}{t(i, c)}$$

Run for a given (unknown) instance i:

1. Compute with claspre features $f(i)$
2. Predict score for each configuration $s_c(i)$ based on the trained models and the computed features
3. Run clasp with best scored configuration $argmin_c(s_c(i))$

Portfolio

Our portfolio of configurations consists of different strategies for:

- Decision Heuristic
- Restarts
- Deletion
- Preprocessing
- Combination of these strategies

- Claspfolio Version 0.8 : 12 configurations
- Claspfolio Version 1.0 : 25 configurations
- Future: Claspfolio Version 1.1 : Approach from Hydra [5] to collect our portfolio

Outline

Benchmark Settings

- Intel Xeon E5520 machine equipped with 2.26 GHz and 6 GB RAM per core
- Systems:
  - *clasp* : clasp 1.3.4
  - *claspfolio*^{b} : theoretical best choice on the portfolio configurations
  - *claspfolio* : claspfolio 0.8.0
  - *claspfolio*^{v} : 10-fold cross validation[1]
  - *paramILS*^{c} : clasp tuned with paramILS[2] on *each* problem class
  - *paramILS*^{a} : clasp tuned with paramILS on *all* problem class
- Benchmark classes of the ASP Competition 2009

[1] dividing instances in training and test sets

[2] paramILS: focused iterated local search tool to tune solver [3]

Benchmark Results

Benchmark Class	#	clasp	claspfolio^b	claspfolio	×	claspfolio^v	×
15Puzzle	37	510	111	208	2.4	254	2.0
BlockedNQueens	65	412	139	264	1.5	410	1.0
ConnectDomSet	21	1,428	30	53	26.9	649	2.2
GraphColouring	23	17,404	5,746	5,867	2.9	5,867	2.9
GraphPartitioning	13	135	57	69	1.9	97	1.4
Hanoi	29	458	35	175	2.6	233	2.0
Labyrinth	29	1,249	112	785	1.5	2,537	0.5
MazeGeneration	28	3,652	558	581	6.2	567	6.4
SchurNumbers	29	726	41	399	1.8	957	0.7
Sokoban	29	18	12	57	0.3	54	0.3
Solitaire	22	2,494	73	317	7.8	1,610	1.5
WeightDomSet	29	3,572	5	1,147	3.1	5,441	0.6
WireRouting	23	1,223	43	144	8.4	289	4.2
Total	377	33,281	6,962	10,066	3.3	18,965	1.8

Table: Runtimes in seconds and speedups on benchmark classes of the 2009 ASP competition

Benchmark Results(2)

Benchmark Class	#	paramILS^c	paramILS^a	claspfolio	claspfolio^v	clasp
15Puzzle	37	104	322	208	254	510
BlockedNQueens	65	212	352	264	410	412
ConnectDomSet	21	28	686	53	649	1,428
GraphColouring	23	7,596	10,865	5,867	5,867	17,404
GraphPartitioning	13	39	86	69	97	135
Hanoi	29	35	147	175	233	458
Labyrinth	29	462	3,080	785	2,537	1,249
MazeGeneration	28	700	2,610	581	567	3,652
SchurNumbers	29	278	871	399	957	726
Sokoban	29	11	18	57	54	18
Solitaire	22	2,374	4,357	317	1,610	2,494
WeightDomSet	29	8	2,649	1,147	5,441	3,572
WireRouting	23	87	535	144	289	1,223
Total	377	11,934	26,578	10,066	18,965	33,281

Table: Comparison with paramILS on benchmark classes of the 2009 ASP competition

Outline

Summary

- Per-instance configuration is faster than a single configuration
- Important factors for predicting configuration performance
  - Features
  - Good portfolio of configurations

Release of claspfolio version 1.0.0[3] : **Today!**

- Future work:
  - Importance of features
  - New features
  - Updates of (SVR-)models
  - Hydra to compute portfolio

[3] `http://potassco.sourceforge.net`

Summary

Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.:

Conflict-driven answer set solving.
In IJCAI'07, AAAI Press (2007) 386–392

Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.:

SATzilla: Portfolio-based algorithm selection for SAT.
JAIR **32** (2008) 565–606

Hutter, F., Hoos, H., Leyton-Brown, K., Stützle, T.:

ParamILS: An automatic algorithm configuration framework.
JAIR **36** (2009) 267–306

Hutter, F., Hoos, H., Leyton-Brown, K.:

Sequential Model-Based Optimization for General Algorithm Configuration.
LION **32** (2011) *to appear*

Xu, L., Hoos, H., Leyton-Brown, K.:

Hydra: Automatically configuring algorithms for portfolio-based selection.
In AAAI'10, AAAI Press (2010) 210–216

Basak, D., Pal, S., Patranabis, D.:

Support vector regression.
NIP **11**(10) (2007) 203–224

C. Chang and C. Li:

LIBSVM: a library for support vector machines
Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`