Constraint solving meets machine learning and data mining

Algorithm portfolios

Kustaa Kangas

University of Helsinki, Finland

November 8, 2012

Constraint problems

- Boolean satisifiability, integer linear programming etc.

Constraint problems

- Boolean satisifiability, integer linear programming etc.
- NP-hard

Constraint problems

- Boolean satisifiability, integer linear programming etc.
- NP-hard
- Many instances solvable with heuristic algorithms

Constraint problems

- Boolean satisifiability, integer linear programming etc.
- NP-hard
- Many instances solvable with heuristic algorithms
- High variance in performance, from milliseconds to weeks

Constraint problems

- Boolean satisifiability, integer linear programming etc.
- NP-hard
- Many instances solvable with heuristic algorithms
- High variance in performance, from milliseconds to weeks
- Different algorithms are fast on different instances

Constraint problems

- Boolean satisifiability, integer linear programming etc.
- NP-hard
- Many instances solvable with heuristic algorithms
- High variance in performance, from milliseconds to weeks
- Different algorithms are fast on different instances
- Typically no single best algorithm

Algorithm selection

Algorithm selection problem

Given a problem instance, which algorithm should we run?

- Ideally: run the algorithm that's fastest on the instance
- Problem: we cannot know without running the algorithms

Traditional solution: run the average-case best algorithm

- Might be reasonably good
- Can be very bad on some instances
- Ignores algorithms that are good on some instances

Algorithm portfolios

Idea: Use several algorithms to improve expected performance.

Algorithm portfolio

1. a collection of algorithms
2. a strategy for running them

A variety of strategies

Algorithm portfolios

Idea: Use several algorithms to improve expected performance.

Algorithm portfolio

1. a collection of algorithms
2. a strategy for running them

A variety of strategies

- Run all algorithms (sequentially / in parallel)

Algorithm portfolios

Idea: Use several algorithms to improve expected performance.

Algorithm portfolio

1. a collection of algorithms
2. a strategy for running them

A variety of strategies

- Run all algorithms (sequentially / in parallel)
- Select one algorithm based on the instance

Algorithm portfolios

Idea: Use several algorithms to improve expected performance.

Algorithm portfolio

1. a collection of algorithms
2. a strategy for running them

A variety of strategies

- Run all algorithms (sequentially / in parallel)
- Select one algorithm based on the instance
- Anything from between

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers
- Trains an *empirical hardness model* for each algorithm

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers
- Trains an *empirical hardness model* for each algorithm
  - explains how hard instances are and why

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers
- Trains an *empirical hardness model* for each algorithm
  - explains how hard instances are and why
  - an approximate predictor of running time

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers
- Trains an *empirical hardness model* for each algorithm
  - explains how hard instances are and why
  - an approximate predictor of running time
  - predicts hardness based on instance features

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers
- Trains an *empirical hardness model* for each algorithm
  - explains how hard instances are and why
  - an approximate predictor of running time
  - predicts hardness based on instance features
- Selects the algorithm predicted to be fastest

SATzilla

A highly successful SAT portfolio solver

- Uses state-of-the-art SAT solvers
- Trains an *empirical hardness model* for each algorithm
  - explains how hard instances are and why
  - an approximate predictor of running time
  - predicts hardness based on instance features
- Selects the algorithm predicted to be fastest
- Performed well in 2007 SAT Competition
  - 1st place in 3 categories, one 2nd place and one 3rd place

Empirical hardness models

Where do empirical hardness models come from?

Empirical hardness models

Where do empirical hardness models come from?

- They must be learned from data:
  1. a set of algorithms
  2. a set of training instances
  3. a set of instance features

Empirical hardness models

Where do empirical hardness models come from?

- They must be learned from data:
  1. a set of algorithms
  2. a set of training instances
  3. a set of instance features
- We use machine learning to exploit correlations between features and algorithm performance

Empirical hardness models

problem instance	x_1	x_2	x_3	x_4	x_5	x_6	running time
instance 1							
instance 2							
instance 3							
instance 4							
instance 5							
instance 6							
instance 7							
instance 8							
instance 9							
instance 10							
instance 11							
instance 12							
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Empirical hardness models

problem instance	x_1	x_2	x_3	x_4	x_5	x_6	running time
instance 1	18	101	1222	6	1	16	
instance 2	23	8124	241	2	1	8	
instance 3	57	32	683	3	5	42	
instance 4	17	435	153	4	1	10	
instance 5	46	76	346	3	4	30	
instance 6	57	32	327	2	11	12	
instance 7	26	62149	2408	3	2	15	
instance 8	70	226	498	3	4	30	
instance 9	30	194	20060	5	2	25	
instance 10	13	108	614	8	4	3	
instance 11	36	307	556	2	5	11	
instance 12	56	100	728	4	13	7	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Empirical hardness models

problem instance	x_1	x_2	x_3	x_4	x_5	x_6	running time
instance 1	18	101	1222	6	1	16	15698
instance 2	23	8124	241	2	1	8	129
instance 3	57	32	683	3	5	42	14680
instance 4	17	435	153	4	1	10	14
instance 5	46	76	346	3	4	30	493
instance 6	57	32	327	2	11	12	7332
instance 7	26	62149	2408	3	2	15	31709
instance 8	70	226	498	3	4	30	214
instance 9	30	194	20060	5	2	25	131
instance 10	13	108	614	8	4	3	1
instance 11	36	307	556	2	5	11	2026
instance 12	56	100	728	4	13	7	60
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Empirical hardness models

problem instance	x_1	x_2	x_3	x_4	x_5	x_6	running time
instance 1	18	101	1222	6	1	16	15698
instance 2	23	8124	241	2	1	8	129
instance 3	57	32	683	3	5	42	14680
instance 4	17	435	153	4	1	10	14
instance 5	46	76	346	3	4	30	493
instance 6	57	32	327	2	11	12	7332
instance 7	26	62149	2408	3	2	15	31709
instance 8	70	226	498	3	4	30	214
instance 9	30	194	20060	5	2	25	131
instance 10	13	108	614	8	4	3	1
instance 11	36	307	556	2	5	11	2026
instance 12	56	100	728	4	13	7	60
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
new instance							

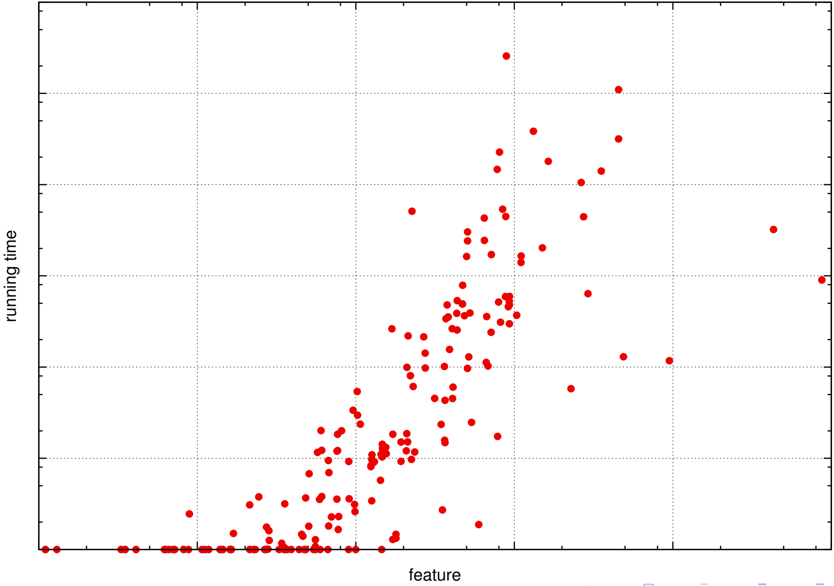
Empirical hardness models

problem instance	x_1	x_2	x_3	x_4	x_5	x_6	running time
instance 1	18	101	1222	6	1	16	15698
instance 2	23	8124	241	2	1	8	129
instance 3	57	32	683	3	5	42	14680
instance 4	17	435	153	4	1	10	14
instance 5	46	76	346	3	4	30	493
instance 6	57	32	327	2	11	12	7332
instance 7	26	62149	2408	3	2	15	31709
instance 8	70	226	498	3	4	30	214
instance 9	30	194	20060	5	2	25	131
instance 10	13	108	614	8	4	3	1
instance 11	36	307	556	2	5	11	2026
instance 12	56	100	728	4	13	7	60
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
new instance	62	3190	1716	3	14	5	

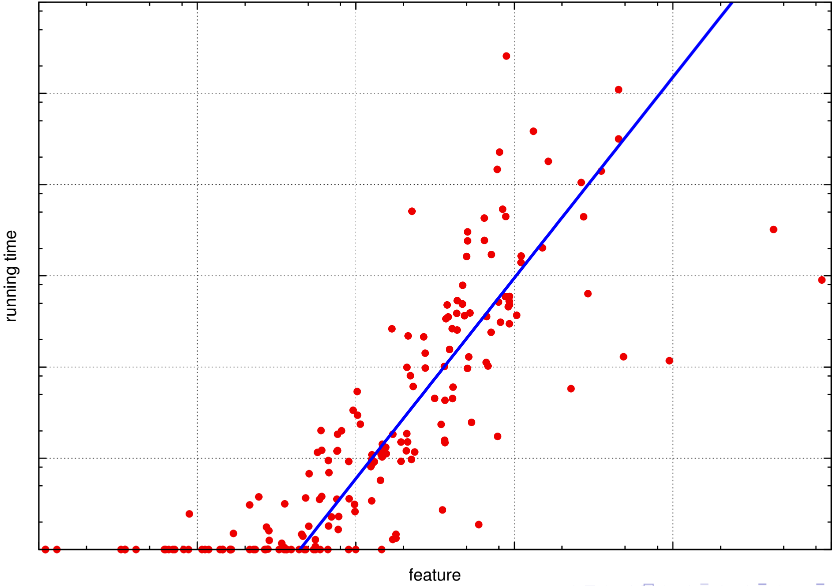
Empirical hardness models

problem instance	x_1	x_2	x_3	x_4	x_5	x_6	running time
instance 1	18	101	1222	6	1	16	15698
instance 2	23	8124	241	2	1	8	129
instance 3	57	32	683	3	5	42	14680
instance 4	17	435	153	4	1	10	14
instance 5	46	76	346	3	4	30	493
instance 6	57	32	327	2	11	12	7332
instance 7	26	62149	2408	3	2	15	31709
instance 8	70	226	498	3	4	30	214
instance 9	30	194	20060	5	2	25	131
instance 10	13	108	614	8	4	3	1
instance 11	36	307	556	2	5	11	2026
instance 12	56	100	728	4	13	7	60
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
new instance	62	3190	1716	3	14	5	?

Linear regression



Linear regression



Linear regression

Not limited to just one variable

- For features $x_1, x_2, \ldots, x_m$ we fit a *hyperplane* f_w of form

$$f_w(x) = w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

- We fit $w_1, w_2, \ldots, w_m$ to minimize prediction error, e.g.

$$\sum_{i=1}^{n} (f_w(x_i) - y_i)^2$$

where y_i is the running time on instance i.

Linear regression

Easily minimized by setting

$$w = (\Phi^T \Phi)^{-1} \Phi^T y$$

where Φ is the feature matrix.

- Dominated by matrix inversion, which is $\mathcal{O}(n^3)$
- Used by SATzilla
- Simple and works well in practice

Identifying features

Success of the model depends crucially on the features.

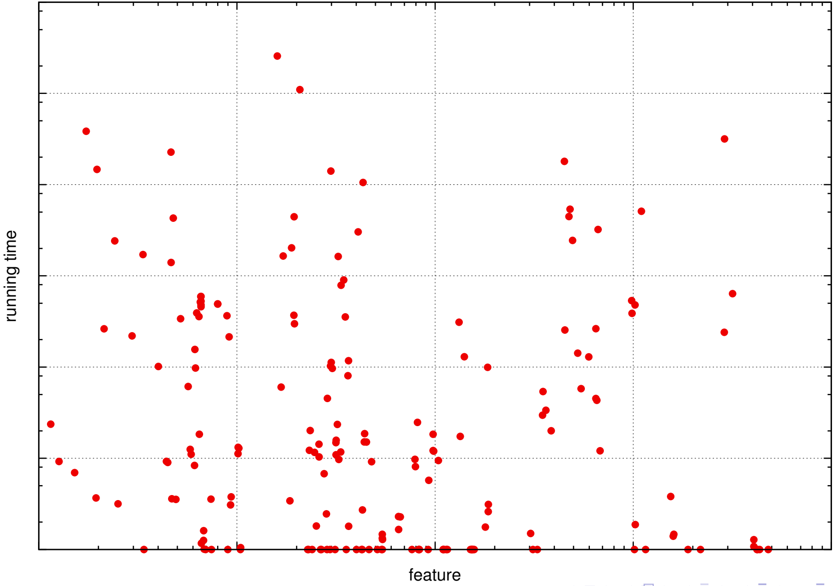
Features must be

- Correlated with running time
- Cheap to compute
  - Feature computation is part of portfolio's running time!

How do we find such features?

- Features are problem-specific
- No automatic way to find them
- Requires domain expertise

Linear regression



SAT features

SATzilla uses 84 features related to e.g.

- instance size
  - number of variables, number of clauses
  - ratio between these two
- balance
  - ratio of positive and negative literals
  - fraction of binary and ternary clauses
- variable–clause graph
  - variable degrees: average, min, max
  - clause degrees: average, min, max
- local search probe statistics
  - number of steps to a local optimum
  - average improvement per step
- proximity to Horn formula

Feature selection

Features can be

- uninformative: no correlation with running time
- redundant: highly correlated with other features

Problematic:

Feature selection

Features can be

- uninformative: no correlation with running time
- redundant: highly correlated with other features

Problematic:

- Unnecessary feature computation

Feature selection

Features can be

- uninformative: no correlation with running time
- redundant: highly correlated with other features

Problematic:

- Unnecessary feature computation
- Learned models are harder to interpret

Feature selection

Features can be

- uninformative: no correlation with running time
- redundant: highly correlated with other features

Problematic:

- Unnecessary feature computation
- Learned models are harder to interpret
- Regression becomes unstable

Feature selection

Useless features can be pruned automatically

- A subset of features can be evaluated by cross-validation

Feature selection

Useless features can be pruned automatically

- A subset of features can be evaluated by cross-validation
- Exhaustive search of all subsets
  - Infeasible for many features

Feature selection

Useless features can be pruned automatically

- A subset of features can be evaluated by cross-validation
- Exhaustive search of all subsets
  - Infeasible for many features
- Greedy heuristic search

Feature selection

Useless features can be pruned automatically

- A subset of features can be evaluated by cross-validation
- Exhaustive search of all subsets
  - Infeasible for many features
- Greedy heuristic search
  - Forward selection: start with no features, add greedily

Feature selection

Useless features can be pruned automatically

- A subset of features can be evaluated by cross-validation
- Exhaustive search of all subsets
  - Infeasible for many features
- Greedy heuristic search
  - Forward selection: start with no features, add greedily
  - Backward elimination: start with all features, remove greedily

Feature selection

Useless features can be pruned automatically

- A subset of features can be evaluated by cross-validation
- Exhaustive search of all subsets
  - Infeasible for many features
- Greedy heuristic search
  - Forward selection: start with no features, add greedily
  - Backward elimination: start with all features, remove greedily
  - Sequential replacement: add and replace greedily

Basis function expansion

Linear regression is limited to linear correlations.

- Problematic: not all useful correlations are linear

Basis function expansion

Linear regression is limited to linear correlations.

- Problematic: not all useful correlations are linear
- Generalizing regression gets complicated

Basis function expansion

Linear regression is limited to linear correlations.

- Problematic: not all useful correlations are linear
- Generalizing regression gets complicated
- Quadratically dependent on x
  $\iff$ linearly dependent on x^2

Basis function expansion

Linear regression is limited to linear correlations.

- Problematic: not all useful correlations are linear
- Generalizing regression gets complicated
- Quadratically dependent on x
  $\iff$ linearly dependent on x^2
- Solution: add functions of original features
- Known as basis function expansion

Basis function expansion

SATzilla uses quadratic expansion:

- Add all pairwise products of features

Basis function expansion

SATzilla uses quadratic expansion:

- Add all pairwise products of features
- Number of features can explode: $84^2 = 7056$

Basis function expansion

SATzilla uses quadratic expansion:

- Add all pairwise products of features
- Number of features can explode: $84^2 = 7056$
- Regression becomes slow

Basis function expansion

SATzilla uses quadratic expansion:

- Add all pairwise products of features
- Number of features can explode: $84^2 = 7056$
- Regression becomes slow
- Can lead to overfitting

Basis function expansion

SATzilla uses quadratic expansion:

- Add all pairwise products of features
- Number of features can explode: $84^2 = 7056$
- Regression becomes slow
- Can lead to overfitting
- Many new features are useless: feature selection before and after expansion!

Terminated runs

What if gathering running time data takes too long?

- Algorithms can run literally for weeks
- Such runs must be terminated prematurely
- How to use these runs to build models?

Terminated runs

Solution 1: discard all such runs

- Not very sensible
- We want to learn that such instances are hard

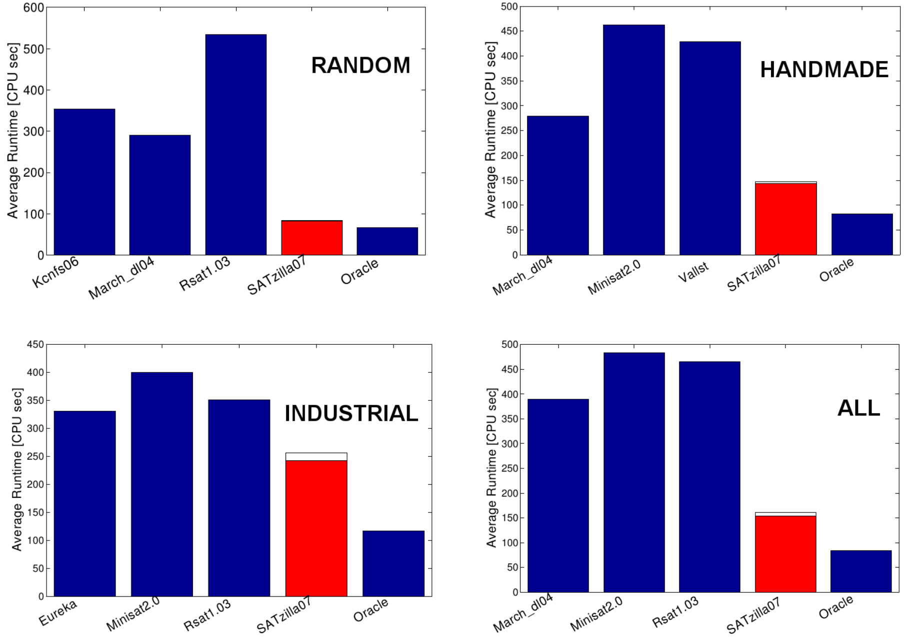
Solution 2: pretend they stopped at the cutoff limit

- Better: takes hardness into account
- Still systematically underestimates hardness

Terminated runs

Solution 3: treat cutoff times as lower bounds

- Known as *censoring* in statistics
- Makes use of all information available


RANDOM
Average Runtime [CPU sec]
0
100
200
300
400
500
600
Kcnfs06
March_dl04
Rsat1.03
SATzilla07
Oracle
HANDMADE
Average Runtime [CPU sec]
0
50
100
150
200
250
300
350
400
450
500
March_dl04
Minisat2.0
Vallst
SATzilla07
Oracle
INDUSTRIAL
Average Runtime [CPU sec]
0
50
100
150
200
250
300
350
400
450
Eureka
Minisat2.0
Rsat1.03
SATzilla07
Oracle
ALL
Average Runtime [CPU sec]
0
50
100
150
200
250
300
350
400
450
500
March_dl04
Minisat2.0
Rsat1.03
SATzilla07
Oracle

Algorithm portfolios

Have been applied to various constraint problems:

- Boolean satisfiability (SAT)
- MaxSAT
- Mixed integer programming
- Constraint satisfaction (CSP)
- Combinatorial auctions
- Answer set programming
- Zero-one integer programming

Conclusions

- Algorithm portfolios can improve expected performance when no single algorithm is dominant.
- Particularly useful for NP-hard constraint problems where the running times exhibit high variance.
- In addition to predicting running time, empirical hardness models are valuable tools for understanding the hardness of problems.