Improving solver performance using optimal parameter tuning

Imre Pólik

Lehigh University
Department of Industrial and Systems Engineering

CORS/INFORMS 2009
Toronto

Outline

Motivation

- Modern optimization solvers have a lot of options
  - algorithmic variants, search strategies
  - tolerances
  - stepsizes
  - neighbourhood parameters
  - turning modules on/off
  - number of refinement steps
- Significant effect on performance/accuracy
- Users are not experts about the solver
- Developers/testers have a lot of hidden knowledge
- Most solvers are always run with the default options
  - default options are never optimal (w.p.1)
  - solvers cannot change their settings after a failure
- It is easy to find better settings for a given problem

Approaches

- Goals
  - reduce solution time
  - increase accuracy
  - recover after failures
- Complications
  - categorical, discrete, continuous options
  - some options have sub-options
  - some problems take a long time to solve
- Before running
  - Find a good set of parameters to solve a problem
- While running
  - While solving a problem, change some parameters

Related work

STOP: Selection Tool for Optimization Parameters (Hunsaker at al., 2007)

- implemented in C++, open-source
- only discrete choices
- general machine learning techniques

Game programming: (Chess, Go)

- well-studied area
- improve general playing strength
- prepare for a given opponent
- mostly continuous parameters (weights)

Preconditioners: Iterative methods

Examples with SeDuMi

SeDuMi is

- a solver for symmetric cone programming (LP, SOCP, SDP);
- implements an IPM;
- open source;
- written in Matlab;
- currently maintained and developed at Lehigh University;
- has a broad set of different options.

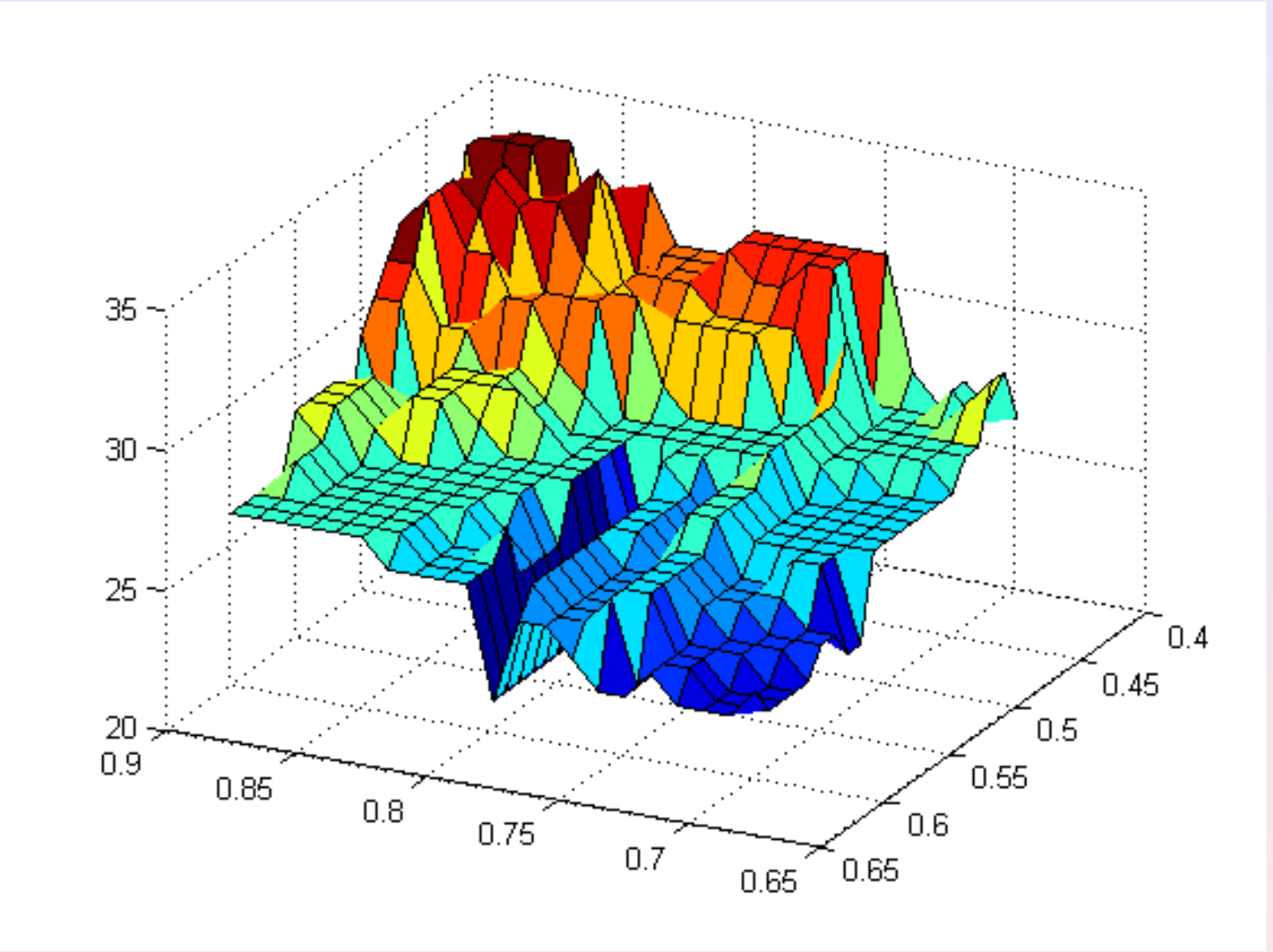
Neighbourhood parameters for IPM

- Defaults: $\beta = 0.5, \vartheta = 0.25$
- Range: $(0, 1)$
- Some optimal values:

Problem	β	ϑ
buck2	0.623	0.798
trto2	0.947	0.7
vibra2	0.96	0.8118

- The default parameters are far from optimal
- Problems arising from discretization
  - Same physical problem
  - Different discretizations yield different problem sizes
  - Solution time: 0.01s to hours
- Idea: tune on small problems
- Goal: reduce the number of iterations

Number of iterations vs (β, ϑ) on trto1



Tuning strategies

- Iterative refinement with exhaustive search
- Global optimization techniques (NOMADm)
- High accuracy is not necessary

Tuning	Testing	Before	After
buck2	buck1	22	22
	buck2	40	35
	buck3	75	65
	buck4	76	67
	buck5	110	81
trto2	trto1	21	19
	trto2	37	30
	trto3	59	48
	trto4	73	65
	trto5	95	80

Step-differentiation in SeDuMi - Background

- Different primal and dual stepsizes in the IPM
- Two line-searches, a bit more work
- Possibly more progress
- Slightly better overall precision
- But: may be too aggressive
- Implemented and *disabled* in SeDuMi by Jos Sturm

Example runs – Failed step-differentiation

```
SeDuMi 1.3, Step-Differentiation
it :  b*y t/tP* t/tD* feas cg cg prec
1 :  -3.16E-001 0.9000 0.9133 -2.46 1 1 4.1E+004
2 :  -3.16E-001 0.9900 0.0000 0.63 1 1 3.4E+003
⋮
13 :  -5.86E-001 0.9106 0.9000 1.36 1 1 2.0E-001
14 :  -5.71E-001 0.9058 0.9000 1.29 1 1 1.0E-001
15 :  -5.71E-001 0.9000 0.0000 1.20 1 1 5.4E-002
16 :  -5.64E-001 0.9244 0.9000 1.10 1 1 1.2E-002
17 :  -5.64E-001 0.9264 0.9000 0.88 1 1 3.9E-003
18 :  -5.64E-001 0.9166 0.9000 0.78 1 1 1.8E-003
19 :  -5.65E-001 0.9190 0.9000 0.72 1 1 6.8E-004
20 :  -5.65E-001 0.9000 0.0000 0.73 1 1 2.1E-004
21 :  -5.66E-001 0.9334 0.9000 0.78 1 1 5.8E-005
⋮
27 :  -5.66E-001 0.9000 0.9000 0.86 1 1 7.3E-007
28 :  -5.67E-001 0.9000 0.9000 0.96 2 2 2.3E-007
29 :  -5.67E-001 0.9000 0.9000 0.99 2 2 7.8E-008
30 :  -5.67E-001 0.9000 0.9000 1.00 2 2 2.4E-008
31 :  -5.67E-001 0.9000 0.9000 1.00 2 2 6.4E-009
```

Example runs – Improved step-differentiation

```
SeDuMi 1.3, Adaptive Step-Differentiation
it :  b*y t/tP* t/tD* feas cg cg prec
1 :  -1.06E-001 0.9000 0.9000 -2.46 1 1 4.1E+004
2 :  -3.54E-001 0.9900 0.9900 -0.33 1 1 8.4E+003
⋮
10 :  -6.39E-001 0.9000 0.9000 1.36 1 1 1.1E+000
11 :  -6.05E-001 0.9000 0.9000 1.19 1 1 5.9E-001
12 :  -5.86E-001 0.9000 0.9000 1.09 1 1 2.8E-001
13 :  -5.78E-001 0.9000 0.9000 1.04 1 1 1.5E-001
14 :  -5.71E-001 0.9012 0.9000 1.03 1 1 6.1E-002
15 :  -5.71E-001 0.9000 0.0000 1.01 1 1 4.8E-002
16 :  -5.68E-001 0.9240 0.9000 1.01 1 1 1.8E-002
17 :  -5.67E-001 0.9000 0.5999 1.00 1 1 9.9E-003
18 :  -5.67E-001 0.9211 0.9000 1.00 1 1 3.8E-003
⋮
25 :  -5.67E-001 0.9078 0.9000 1.00 1 1 4.2E-006
26 :  -5.67E-001 0.9000 0.9041 1.00 2 2 1.3E-006
27 :  -5.67E-001 0.9000 0.8208 1.00 2 2 4.3E-007
28 :  -5.67E-001 0.9013 0.9000 1.00 2 3 7.9E-008
29 :  -5.67E-001 0.9900 0.9900 1.00 4 5 7.5E-009
```

Step-differentiation in SeDuMi - Strategies

- Use step differentiation
  - if numerical problems occur, or
  - after 20 iterations, or
  - if the algorithm is converging.
- These are empirical rules
- Default option in SeDuMi
- Why does step-differentiation lead to these problems?

Future work

- Automate tuning (learning or prior knowledge)
- Understand the effect of parameters
- Statistical analysis of multiple runs
- Borrow ideas from game programming
- Apply machine learning
- Continuous control for tuning while running

Improving solver performance using optimal parameter tuning

Imre Pólik

Lehigh University
Department of Industrial and Systems Engineering

CORS/INFORMS 2009
Toronto