Automating the Configuration of Algorithms for Solving Hard Computational Problems

Ph.D. Thesis Defence

Frank Hutter

Supervisory committee:

- Prof. Holger Hoos (supervisor)
- Prof. Kevin Leyton-Brown (co-supervisor)
- Prof. Kevin Murphy (co-supervisor)
- Prof. Alan Mackworth

University Examiners:

- Prof. Michael Friedlander (CS)
- Prof. Lutz Lampe (ECE)

External Examiner: Prof. ?

Chair: Prof. John Nelson (Forestry)

Parameters in Algorithms

Most algorithms have parameters

- Decisions that are left open during algorithm design
  - numerical parameters (*e.g.*, real-valued thresholds)
  - categorical parameters (*e.g.*, which heuristic to use)

Parameters in Algorithms

Most algorithms have parameters

- Decisions that are left open during algorithm design
  - numerical parameters (*e.g.*, real-valued thresholds)
  - categorical parameters (*e.g.*, which heuristic to use)
- Set to maximize empirical performance

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities
- 63 parameters that affect search trajectory

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities
- 63 parameters that affect search trajectory

  > "Integer programming problems are more sensitive to specific parameter settings, so **you may need to experiment with them**." [CPLEX 10.0 user manual, page 130]

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities

- 63 parameters that affect search trajectory

  "Integer programming problems are more sensitive to specific parameter settings, so **you may need to experiment with them**." [CPLEX 10.0 user manual, page 130]

- "Experiment with them"

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities
- 63 parameters that affect search trajectory

  "Integer programming problems are more sensitive to specific parameter settings, so **you may need to experiment with them**." [CPLEX 10.0 user manual, page 130]

- "Experiment with them"
  - Perform manual optimization in 63-dimensional space
  - Complex, unintuitive interactions between parameters

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities
- 63 parameters that affect search trajectory

  "Integer programming problems are more sensitive to specific parameter settings, so **you may need to experiment with them**." [CPLEX 10.0 user manual, page 130]
- "Experiment with them"
  - Perform manual optimization in 63-dimensional space
  - Complex, unintuitive interactions between parameters
  - **Humans are not good at that**

Real-world example for parameterized algorithms: commercial optimization tool CPLEX

- State of the art for mixed integer programming (MIP)
- Large user base
  - Over 1 300 corporations and over 1 000 universities

- 63 parameters that affect search trajectory

  > "Integer programming problems are more sensitive to specific parameter settings, so **you may need to experiment with them**." [CPLEX 10.0 user manual, page 130]

- "Experiment with them"
  - Perform manual optimization in 63-dimensional space
  - Complex, unintuitive interactions between parameters
  - **Humans are not good at that**

  $\rightsquigarrow$ developed the first automated tools for this type of problem

Automated Algorithm Configuration

Automate the setting of algorithm parameters

- Eliminate most tedious part of algorithm design and end use
- Save development time
- Improve performance

Automated Algorithm Configuration

Automate the setting of algorithm parameters

- Eliminate most tedious part of algorithm design and end use
- Save development time
- Improve performance
- First to consider the general problem,
  in particular **many categorical parameters**
  - E.g. 50/63 CPLEX parameters are categorical

$\rightsquigarrow$ Algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches

- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - 1^{st} and 2^{nd} approach
    to configure algorithms with many categorical parameters
- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - 1st and 2nd approach
    to configure algorithms with many categorical parameters
- Demonstrated practical relevance of algorithm configuration
  - CPLEX: up to 23-fold speedup

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - 1st and 2nd approach
    to configure algorithms with many categorical parameters
- Demonstrated practical relevance of algorithm configuration
  - CPLEX: up to 23-fold speedup
  - SAT solver: 500-fold speedup for software verification

Outline

Outline

1. Problem Definition & Intuition

2. Model-Free Search for Algorithm Configuration

3. Model-Based Search for Algorithm Configuration

4. Conclusions

Algorithm Configuration as Function Optimization

Deterministic algorithm with continuous parameters

- "Blackbox function" $f : \mathbb{R}^n \rightarrow \mathbb{R}$
- Can query function at arbitrary points $\boldsymbol{\theta} \in \mathbb{R}^n$

$$\text{Find} \quad \min_{\boldsymbol{\theta} \in \mathbb{R}^n} f(\boldsymbol{\theta})$$

Algorithm Configuration as Function Optimization

Deterministic algorithm with continuous parameters

- "Blackbox function" $f : \mathbb{R}^n \to \mathbb{R}$
- Can query function at arbitrary points $\boldsymbol{\theta} \in \mathbb{R}^n$

$$\text{Find} \quad \min_{\boldsymbol{\theta} \in \mathbb{R}^n} f(\boldsymbol{\theta})$$

Randomized algorithm with continuous parameters

- For each θ: distribution $D_{\boldsymbol{\theta}}$
- Optimize statistical parameter τ (*e.g.*, expected value)

Algorithm Configuration as Function Optimization

Deterministic algorithm with continuous parameters

- "Blackbox function" $f : \mathbb{R}^n \to \mathbb{R}$
- Can query function at arbitrary points $\boldsymbol{\theta} \in \mathbb{R}^n$

$$\text{Find } \min_{\boldsymbol{\theta} \in \mathbb{R}^n} f(\boldsymbol{\theta})$$

Randomized algorithm with continuous parameters

- For each θ: distribution $D_{\boldsymbol{\theta}}$
- Optimize statistical parameter τ (*e.g.*, expected value)
- Can sample from distribution $D_{\boldsymbol{\theta}}$ at arbitrary points $\boldsymbol{\theta} \in \boldsymbol{\Theta}$

$$\text{Find } \min_{\boldsymbol{\theta} \in \mathbb{R}^n} \tau(D_{\boldsymbol{\theta}})$$

Algorithm Configuration: General Case

Difference to "standard" blackbox optimization

- Categorical parameters

Algorithm Configuration: General Case

Difference to "standard" blackbox optimization

- Categorical parameters
- Distribution of costs
  - across multiple repeated runs for randomized algorithms
  - across problem instances

Algorithm Configuration: General Case

Difference to “standard” blackbox optimization

- Categorical parameters
- Distribution of costs
  - across multiple repeated runs for randomized algorithms
  - across problem instances
- Can *terminate* unsuccessful runs *early*

Outline

1. Problem Definition & Intuition
2. Model-Free Search for Algorithm Configuration
   - ParamILS: Iterated Local Search in Configuration Space
   - "Real-World" Applications of ParamILS
3. Model-Based Search for Algorithm Configuration
4. Conclusions

Outline

Simple manual approach for configuration

Start with some parameter configuration

Simple manual approach for configuration

Start with some parameter configuration

Modify a single parameter

Simple manual approach for configuration

Start with some parameter configuration

Modify a single parameter
if *results on benchmark set improve* **then**
    keep new configuration

Simple manual approach for configuration

Start with some parameter configuration
repeat
    Modify a single parameter
    if *results on benchmark set improve* **then**
        keep new configuration
until *no more improvement possible (or "good enough")*

Simple manual approach for configuration

Start with some parameter configuration
repeat
  Modify a single parameter
  if *results on benchmark set improve* **then**
    keep new configuration
until *no more improvement possible (or "good enough")*

$\rightsquigarrow$ Manually-executed **local search**

The ParamILS Framework

Iterated Local Serach in parameter configuration space:

Choose initial parameter configuration θ

Perform *subsidiary local search* on θ

The ParamILS Framework

Iterated Local Serach in parameter configuration space:

Choose initial parameter configuration θ

Perform *subsidiary local search* on θ

While tuning time left:

- $\theta' := \theta$
- Perform *perturbation* on θ
- Perform *subsidiary local search* on θ

The ParamILS Framework

Iterated Local Serach in parameter configuration space:

Choose initial parameter configuration θ
Perform *subsidiary local search* on θ
While tuning time left:

> $\theta' := \theta$
> Perform *perturbation* on θ
> Perform *subsidiary local search* on θ
>
> Based on *acceptance criterion*,
> keep θ or revert to $\theta := \theta'$

The ParamILS Framework

Iterated Local Serach in parameter configuration space:

Choose initial parameter configuration θ
Perform *subsidiary local search* on θ
While tuning time left:

$\theta' := \theta$
Perform *perturbation* on θ
Perform *subsidiary local search* on θ

Based on *acceptance criterion*,
keep θ or revert to $\theta := \theta'$

With probability $p_{restart}$ randomly pick new θ

$\rightsquigarrow$ Performs **biased random walk over local optima**

Instantiations of ParamILS Framework

How to evaluate each configuration?

- BasicILS(N): perform fixed number of N runs to evaluate a configuration θ
  - Blocking: use same N (instance, seed) pairs for each $\boldsymbol{\theta}$

Instantiations of ParamILS Framework

How to evaluate each configuration?

- BasicILS(N): perform fixed number of N runs to evaluate a configuration θ
  - Blocking: use same N (instance, seed) pairs for each $\boldsymbol{\theta}$
- FocusedILS: adaptive choice of $N(\boldsymbol{\theta})$
  - small $N(\boldsymbol{\theta})$ for poor configurations $\boldsymbol{\theta}$
  - large $N(\boldsymbol{\theta})$ only for good $\boldsymbol{\theta}$

Instantiations of ParamILS Framework

How to evaluate each configuration?

- BasicILS(N): perform fixed number of N runs to evaluate a configuration θ
  - Blocking: use same N (instance, seed) pairs for each $\boldsymbol{\theta}$
- FocusedILS: adaptive choice of $N(\boldsymbol{\theta})$
  - small $N(\boldsymbol{\theta})$ for poor configurations $\boldsymbol{\theta}$
  - large $N(\boldsymbol{\theta})$ only for good $\boldsymbol{\theta}$
  - typically outperforms BasicILS

Empirical Comparison to Previous Configuration Procedure

CALIBRA system [Adenso-Diaz & Laguna, '06]

- Based on fractional factorial designs
- Limited to continuous parameters
- Limited to 5 parameters

Empirical Comparison to Previous Configuration Procedure

CALIBRA system [Adenso-Diaz & Laguna, '06]

- Based on fractional factorial designs
- Limited to continuous parameters
- Limited to 5 parameters

Empirical comparison

- FocusedILS typically did better, never worse
- More importantly, much more general

Adaptive Choice of Cutoff Time

- Evaluation of poor configurations takes especially long

Adaptive Choice of Cutoff Time

- Evaluation of poor configurations takes especially long
- Can terminate evaluations early
  - Incumbent solution provides bound
  - Can stop evaluation once bound is reached

Adaptive Choice of Cutoff Time

- Evaluation of poor configurations takes especially long
- Can terminate evaluations early
  - Incumbent solution provides bound
  - Can stop evaluation once bound is reached
- Results
  - Provably never hurts
  - Sometimes substantial speedups (factor 10)

Outline

1. Problem Definition & Intuition

2. Model-Free Search for Algorithm Configuration
    - ParamILS: Iterated Local Search in Configuration Space
    - "Real-World" Applications of ParamILS

3. Model-Based Search for Algorithm Configuration

4. Conclusions

Configuration of ILOG CPLEX

- Recall: 63 parameters, 1.78×10^{38} possible configurations
- Ran FocusedILS for 2 days on 10 machines

Configuration of ILOG CPLEX

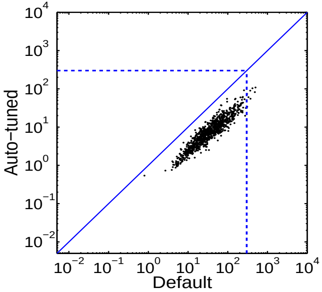
- Recall: 63 parameters, 1.78×10^{38} possible configurations
- Ran FocusedILS for 2 days on 10 machines
- Compared against default

  > "A great deal of algorithmic development effort has been devoted to establishing default ILOG CPLEX parameter settings that achieve good performance on a wide variety of MIP models." [CPLEX 10.0 user manual, page 247]

Configuration of ILOG CPLEX

- Recall: 63 parameters, 1.78×10^{38} possible configurations
- Ran FocusedILS for 2 days on 10 machines
- Compared against default

  "A great deal of algorithmic development effort has been devoted to establishing default ILOG CPLEX parameter settings that achieve good performance on a wide variety of MIP models." [CPLEX 10.0 user manual, page 247]

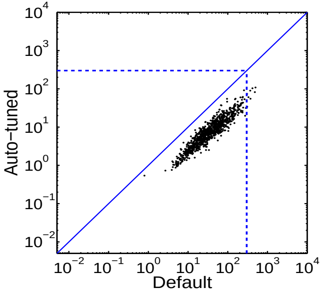


Combinatorial auctions: 7-fold speedup

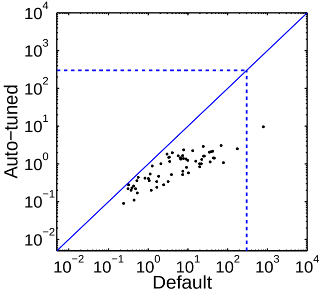
Configuration of ILOG CPLEX

- Recall: 63 parameters, 1.78×10^{38} possible configurations
- Ran FocusedILS for 2 days on 10 machines
- Compared against default

  "A great deal of algorithmic development effort has been devoted to establishing default ILOG CPLEX parameter settings that achieve good performance on a wide variety of MIP models." [CPLEX 10.0 user manual, page 247]



Combinatorial auctions: 7-fold speedup



Mixed integer knapsack: 23-fold speedup

Configuration of SAT Solver for Verification

SAT (propositional satisfiability problem)

- Prototypical $\mathcal{NP}$-hard problem
- Interesting theoretically and in practical applications

Configuration of SAT Solver for Verification

SAT (propositional satisfiability problem)

- Prototypical $\mathcal{NP}$-hard problem
- Interesting theoretically and in practical applications

Formal verification

- Bounded model checking
- Software verification
- Recent progress based on SAT solvers

Configuration of SAT Solver for Verification

SAT (propositional satisfiability problem)

- Prototypical $\mathcal{NP}$-hard problem
- Interesting theoretically and in practical applications

Formal verification

- Bounded model checking
- Software verification
- Recent progress based on SAT solvers

Spear, tree search solver for industrial SAT instances

- 26 parameters, 8.34×10^{17} configurations

Configuration of SAT Solver for Verification

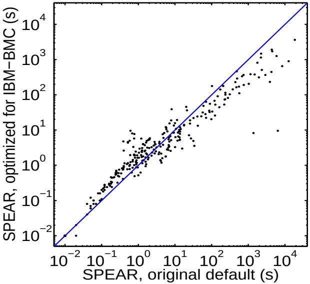
- Ran FocusedILS for 2 days on 10 machines

Configuration of SAT Solver for Verification

- Ran FocusedILS for 2 days on 10 machines
- Compared to manually-engineered default
  - 1 week of performance tuning
  - competitive with the state of the art

Configuration of SAT Solver for Verification

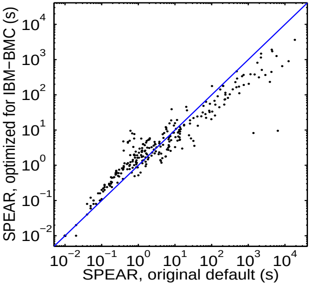
- Ran FocusedILS for 2 days on 10 machines
- Compared to manually-engineered default
  - 1 week of performance tuning
  - competitive with the state of the art



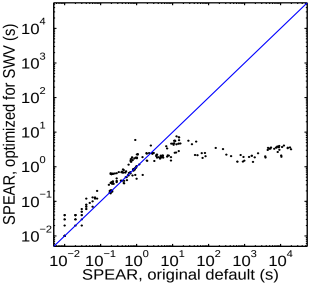
IBM Bounded Model Checking:
4.5-fold speedup

Configuration of SAT Solver for Verification

- Ran FocusedILS for 2 days on 10 machines
- Compared to manually-engineered default
  - 1 week of performance tuning
  - competitive with the state of the art



IBM Bounded Model Checking:
4.5-fold speedup



Software verification: 500-fold speedup
$\rightsquigarrow$ won 2007 SMT competition

Other Fielded Applications of ParamILS

- SAPS, local search for SAT
  - $\rightsquigarrow$ 8-fold and 130-fold speedup

Other Fielded Applications of ParamILS

- SAPS, local search for SAT
  - $\rightsquigarrow$ 8-fold and 130-fold speedup
- SAT4J, tree search for SAT
  - $\rightsquigarrow$ 11-fold speedup

Other Fielded Applications of ParamILS

- SAPS, local search for SAT
  - $\rightsquigarrow$ 8-fold and 130-fold speedup
- SAT4J, tree search for SAT
  - $\rightsquigarrow$ 11-fold speedup
- GLS^{+} for Most Probable Explanation (MPE) problem
  - $\rightsquigarrow$ $>$ 360-fold speedup

Other Fielded Applications of ParamILS

- SAPS, local search for SAT
  - $\rightsquigarrow$ 8-fold and 130-fold speedup
- SAT4J, tree search for SAT
  - $\rightsquigarrow$ 11-fold speedup
- GLS$^+$ for Most Probable Explanation (MPE) problem
  - $\rightsquigarrow$ $>$ 360-fold speedup
- Applications by others
  - Protein folding [Thatchuk, Shmygelska & Hoos '07]
  - Time-tabling [Fawcett, Hoos & Chiarandini '09]
  - Local Search for SAT [Khudabukhsh, Xu, Hoos, & Leyton-Brown '09]

Other Fielded Applications of ParamILS

- SAPS, local search for SAT
  - $\rightsquigarrow$ 8-fold and 130-fold speedup
- SAT4J, tree search for SAT
  - $\rightsquigarrow$ 11-fold speedup
- GLS$^+$ for Most Probable Explanation (MPE) problem
  - $\rightsquigarrow$ $>$ 360-fold speedup
- Applications by others
  - Protein folding [Thatchuk, Shmygelska & Hoos '07]
  - Time-tabling [Fawcett, Hoos & Chiarandini '09]
  - Local Search for SAT [Khudabukhsh, Xu, Hoos, & Leyton-Brown '09]
  - $\rightsquigarrow$ demonstrates versatility & maturity

Outline

1. Problem Definition & Intuition

2. Model-Free Search for Algorithm Configuration

3. Model-Based Search for Algorithm Configuration
   - State of the Art
   - Improvements for Stochastic Blackbox Optimization
   - Beyond Stochastic Blackbox Optimization

4. Conclusions

Model-Based Optimization: Motivation

Fundamentally different approach for algorithm configuration

- So far: discussed local search approach
- Now: alternative choice, based on predictive models

Model-Based Optimization: Motivation

Fundamentally different approach for algorithm configuration

- So far: discussed local search approach
- Now: alternative choice, based on predictive models
  - Model-based optimization was less well developed

  $\leadsto$ emphasis on methodological improvements

Model-Based Optimization: Motivation

Fundamentally different approach for algorithm configuration

- So far: discussed local search approach
- Now: alternative choice, based on predictive models
  - Model-based optimization was less well developed

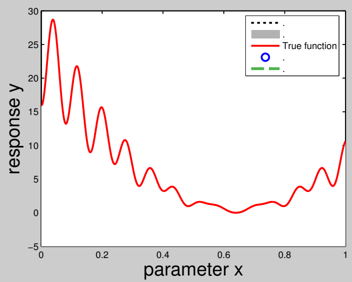
  $\rightsquigarrow$ emphasis on methodological improvements
- In then end: state-of-the-art configuration tool

Outline

1. Problem Definition & Intuition
2. Model-Free Search for Algorithm Configuration
3. Model-Based Search for Algorithm Configuration
    - State of the Art
    - Improvements for Stochastic Blackbox Optimization
    - Beyond Stochastic Blackbox Optimization
4. Conclusions

Model-Based Deterministic Blackbox Optimization (BBO)

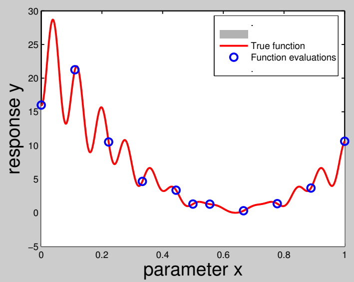
EGO algorithm [Jones, Schonlau & Welch '98]



Model-Based Deterministic Blackbox Optimization (BBO)

EGO algorithm [Jones, Schonlau & Welch '98]

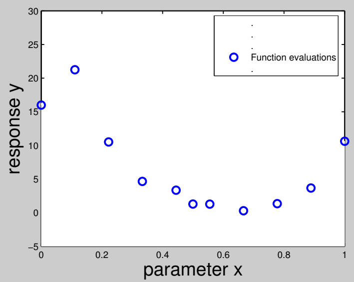
1. Get response values at initial design points



Model-Based Deterministic Blackbox Optimization (BBO)

EGO algorithm [Jones, Schonlau & Welch '98]

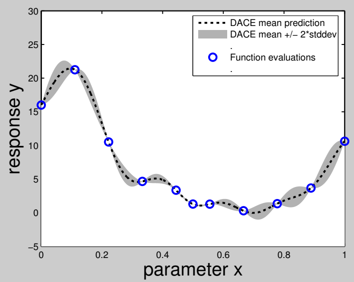
1. Get response values at initial design points



Model-Based Deterministic Blackbox Optimization (BBO)

EGO algorithm [Jones, Schonlau & Welch '98]

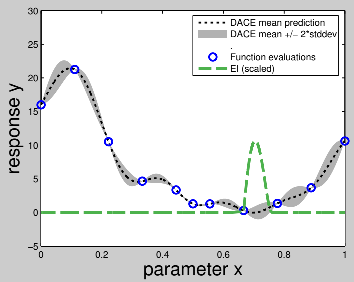
1. Get response values at initial design points
2. Fit a model to the data



Model-Based Deterministic Blackbox Optimization (BBO)

EGO algorithm [Jones, Schonlau & Welch '98]

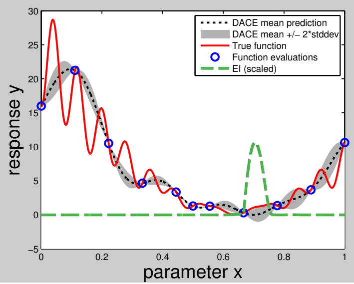
1. Get response values at initial design points
2. Fit a model to the data
3. Use model to pick most promising next design point



Model-Based Deterministic Blackbox Optimization (BBO)

EGO algorithm [Jones, Schonlau & Welch '98]

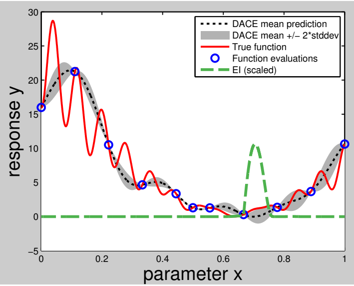
1. Get response values at initial design points
2. Fit a model to the data
3. Use model to pick most promising next design point
4. Repeat 2. and 3. until time is up



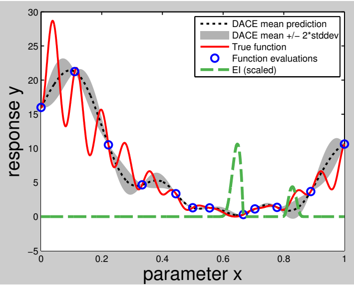
Model-Based Deterministic Blackbox Optimization (BBO)

EGO algorithm [Jones, Schonlau & Welch '98]

1. Get response values at initial design points
2. Fit a model to the data
3. Use model to pick most promising next design point
4. Repeat 2. and 3. until time is up



First step



Second step

Stochastic Blackbox Optimization (BBO): State of the Art

Extensions of EGO algorithm for *stochastic* case

- Sequential Parameter Optimization (SPO)
  [Bartz-Beielstein, Preuss, Lasarczyk, '05-'09]
- Sequential Kriging Optimization (SKO)
  [Huang, Allen, Notz & Zeng, '06]

Stochastic Blackbox Optimization (BBO): State of the Art

Extensions of EGO algorithm for *stochastic* case

- Sequential Parameter Optimization (SPO)
  [Bartz-Beielstein, Preuss, Lasarczyk, '05-'09]
- Sequential Kriging Optimization (SKO)
  [Huang, Allen, Notz & Zeng, '06]

Application domain for stochastic BBO

- *Randomized* algorithms with continuous parameters
- Optimization for single instances

Stochastic Blackbox Optimization (BBO): State of the Art

Extensions of EGO algorithm for *stochastic* case

– Sequential Parameter Optimization (SPO)
  [Bartz-Beielstein, Preuss, Lasarczyk, '05-'09]

– Sequential Kriging Optimization (SKO)
  [Huang, Allen, Notz & Zeng, '06]

Application domain for stochastic BBO

- *Randomized* algorithms with continuous parameters
- Optimization for single instances

Empirical Evaluation

- SPO more robust

Outline

1. Problem Definition & Intuition
2. Model-Free Search for Algorithm Configuration
3. Model-Based Search for Algorithm Configuration
   - State of the Art
   - Improvements for Stochastic Blackbox Optimization
   - Beyond Stochastic Blackbox Optimization
4. Conclusions

Improvements for stochastic BBO

I: Studied SPO components

- Improved component: "intensification mechanism"
  - Increase $N(\theta)$ similarly as in FocusedILS
  - Improved robustness

Improvements for stochastic BBO

I: Studied SPO components

- Improved component: "intensification mechanism"
  - Increase $N(\boldsymbol{\theta})$ similarly as in FocusedILS
  - Improved robustness

II: Better Models

- Compared various probabilistic models
  - Model SPO uses
  - Approximate Gaussian process (GP)
  - Random forest (RF)

Improvements for stochastic BBO

I: Studied SPO components

- Improved component: "intensification mechanism"
  - Increase $N(\boldsymbol{\theta})$ similarly as in FocusedILS
  - Improved robustness

II: Better Models

- Compared various probabilistic models
  - Model SPO uses
  - Approximate Gaussian process (GP)
  - Random forest (RF)
- New models much better
  - Resulting configuration procedure: ActiveConfigurator
  - **Improved state of the art** for model-based stochastic BBO

Improvements for stochastic BBO

I: Studied SPO components

- Improved component: "intensification mechanism"
  - Increase $N(\boldsymbol{\theta})$ similarly as in FocusedILS
  - Improved robustness

II: Better Models

- Compared various probabilistic models
  - Model SPO uses
  - Approximate Gaussian process (GP)
  - Random forest (RF)
- New models much better
  - Resulting configuration procedure: ActiveConfigurator
  - **Improved state of the art** for model-based stochastic BBO
  - *Randomized* algorithm with continuous parameters
  - Optimization for single instances

Outline

Extension I: Categorical Parameters

Models that can handle categorical inputs

- Random forests: out of the box
- Extended (approximate) Gaussian processes
  - new kernel based on weighted Hamming distance

Extension I: Categorical Parameters

Models that can handle categorical inputs

- Random forests: out of the box
- Extended (approximate) Gaussian processes
  - new kernel based on weighted Hamming distance

Application domain

- Algorithms with categorical parameters
- Single instances

Extension I: Categorical Parameters

Models that can handle categorical inputs

- Random forests: out of the box
- Extended (approximate) Gaussian processes
  - new kernel based on weighted Hamming distance

Application domain

- Algorithms with categorical parameters
- Single instances

Empirical evaluation

- **ActiveConfigurator outperformed FocusedILS**

Extension II: Multiple Instances

Models incorporating multiple instances

- Can still learn probabilistic models of algorithm performance
- Model inputs:
  - algorithm parameters
  - instance features

Extension II: Multiple Instances

Models incorporating multiple instances

- Can still learn probabilistic models of algorithm performance
- Model inputs:
  - algorithm parameters
  - instance features

General algorithm configuration

- Algorithms with categorical parameters
- Multiple instances

Extension II: Multiple Instances

Models incorporating multiple instances

- Can still learn probabilistic models of algorithm performance
- Model inputs:
  - algorithm parameters
  - instance features

General algorithm configuration

- Algorithms with categorical parameters
- Multiple instances

Empirical evaluation

- ActiveConfigurator never worse than FocusedILS
- Overall: model-based approaches very promising

Outline

Conclusions

Algorithm configuration

- Is a high-dimensional optimization problem
  - Can be solved by automated approaches
  - Sometimes much better than by human experts

Conclusions

Algorithm configuration

- Is a high-dimensional optimization problem
  - Can be solved by automated approaches
  - Sometimes much better than by human experts
- Can cut development time & improve results

Conclusions

Algorithm configuration

- Is a high-dimensional optimization problem
  - Can be solved by automated approaches
  - Sometimes much better than by human experts
- Can cut development time & improve results

Scaling to very complex problems allows us to

- Build very flexible algorithm frameworks
- Apply automated tool to instantiate framework
  - $\rightsquigarrow$ Generate custom algorithms for different problem types

Conclusions

Algorithm configuration

- Is a high-dimensional optimization problem
  - Can be solved by automated approaches
  - Sometimes much better than by human experts
- Can cut development time & improve results

Scaling to very complex problems allows us to

- Build very flexible algorithm frameworks
- Apply automated tool to instantiate framework
  - $\rightsquigarrow$ Generate custom algorithms for different problem types

Blackbox approaches

- Very general
- Can be used to optimize *your* parameters

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - Model-free Iterated Local Search approach
- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - Model-free Iterated Local Search approach
  - Improved & Extended Sequential Model-Based Optimization
- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - Model-free Iterated Local Search approach
  - Improved & Extended Sequential Model-Based Optimization
- Demonstrated practical relevance of algorithm configuration

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
- Two fundamentally different solution approaches
  - Model-free Iterated Local Search approach
  - Improved & Extended Sequential Model-Based Optimization
- Demonstrated practical relevance of algorithm configuration
  - CPLEX: up to 23-fold speedup
  - SPEAR: 500-fold speedup for software verification

Main Contribution of this thesis

Comprehensive study of the algorithm configuration problem

- Empirical analysis of configuration scenarios
  [Ready for submission]
- Two fundamentally different solution approaches
  - Model-free Iterated Local Search approach [AAAI'07]
  - Improved & Extended Sequential Model-Based Optimization [GECCO'09; EMAA'09]
- Demonstrated practical relevance of algorithm configuration
  - CPLEX: up to 23-fold speedup [JAIR'09]
  - SPEAR: 500-fold speedup for software verification [FMCAD'07]

Important Directions for the Next Few Years

- Improve configuration procedures from practical point of view
  - Mixed categorical/numerical optimization
  - Make easier to use off the shelf

Important Directions for the Next Few Years

- Improve configuration procedures from practical point of view
  - Mixed categorical/numerical optimization
  - Make easier to use off the shelf
- More sophisticated model-based methods
  - Use model to select most informative instance
  - Use model to select best cutoff time
  - Per-instance setting of parameters

Important Directions for the Next Few Years

- Improve configuration procedures from practical point of view
  - Mixed categorical/numerical optimization
  - Make easier to use off the shelf
- More sophisticated model-based methods
  - Use model to select most informative instance
  - Use model to select best cutoff time
  - Per-instance setting of parameters
- Explore other fields of applications

Thanks to

- Supervisory committee
  - Holger Hoos (supervisor)
  - Kevin Leyton-Brown (co-supervisor)
  - Kevin Murphy (co-supervisor)
  - Alan Mackworth
- Further collaborators
  - Domagoj Babić
  - Thomas Bartz-Beielstein
  - Youssef Hamadi
  - Alan Hu
  - Thomas Stützle
  - Dave Tompkins
  - Lin Xu
- LCI and BETA lab faculty and students