Time-Bounded Sequential Parameter Optimization

Frank Hutter, Holger H. Hoos,
Kevin Leyton-Brown, Kevin P. Murphy

Department of Computer Science
University of British Columbia
Canada
{hutter, hoos, kevinlb, murphyk}@cs.ubc.ca

Automated Parameter Optimization

Most algorithms have parameters

- Decisions that are left open during algorithm design
- Instantiate to optimize empirical performance

Automated Parameter Optimization

Most algorithms have parameters

- Decisions that are left open during algorithm design
- Instantiate to optimize empirical performance
- E.g. local search
  - neighbourhoods, restarts, types of perturbations, tabu length (or range for it), etc

Automated Parameter Optimization

Most algorithms have parameters

- Decisions that are left open during algorithm design
- Instantiate to optimize empirical performance
- E.g. local search
  - neighbourhoods, restarts, types of perturbations, tabu length (or range for it), etc
- E.g., tree search
  - Branching heuristics, no-good learning, restarts, pre-processing, etc

Automated Parameter Optimization

Most algorithms have parameters

- Decisions that are left open during algorithm design
- Instantiate to optimize empirical performance
- E.g. local search
  - neighbourhoods, restarts, types of perturbations, tabu length (or range for it), etc
- E.g., tree search
  - Branching heuristics, no-good learning, restarts, pre-processing, etc

Automatically find good instantiation of parameters

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

Parameter Optimization Methods

- Lots of work on numerical parameters, e.g.
  - *CALIBRA* [Adenso-Diaz & Laguna, '06]
  - Population-based, e.g. *CMA-ES* [Hansen et al, '95-present]

Parameter Optimization Methods

- Lots of work on numerical parameters, e.g.
  - *CALIBRA* [Adenso-Diaz & Laguna, '06]
  - Population-based, e.g. *CMA-ES* [Hansen et al, '95-present]
- Categorical parameters
  - Racing algorithms, *F-Race* [Birattari et al., '02-present]
  - Iterated Local Search, *ParamILS* [Hutter et al., AAAI '07 & JAIR'09]

Parameter Optimization Methods

- Lots of work on numerical parameters, e.g.
  - *CALIBRA* [Adenso-Diaz & Laguna, '06]
  - Population-based, e.g. *CMA-ES* [Hansen et al, '95-present]
- Categorical parameters
  - Racing algorithms, *F-Race* [Birattari et al., '02-present]
  - Iterated Local Search, *ParamILS* [Hutter et al., AAAI '07 & JAIR'09]
- Success of parameter optimization
  - Many parameters (*e.g.*, CPLEX with 63 parameters)
  - Large speedups (sometimes orders of magnitude!)
  - For many problems: SAT, MIP, time-tabling, protein folding, ...

Limitations of Model-Free Parameter Optimization

Model-free methods only return the best parameter setting

- Often that is all you need
  - E.g.: end user can customize algorithm

Limitations of Model-Free Parameter Optimization

Model-free methods only return the best parameter setting

- Often that is all you need
  - E.g.: end user can customize algorithm
- But sometimes we would like to know more
  - How important is each of the parameters?
  - Which parameters interact?
  - For which types of instances is a parameter setting good?

  $\rightsquigarrow$ Inform algorithm designer

Limitations of Model-Free Parameter Optimization

Model-free methods only return the best parameter setting

- Often that is all you need
  - E.g.: end user can customize algorithm
- But sometimes we would like to know more
  - How important is each of the parameters?
  - Which parameters interact?
  - For which types of instances is a parameter setting good?

  $\rightsquigarrow$ Inform algorithm designer

Response surface models can help

- Predictive models of algorithm performance with given parameter settings

Sequential Parameter Optimization (SPO)

- Original SPO [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization

Sequential Parameter Optimization (SPO)

- Original SPO [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization
- Studied SPO components [Hutter et al, GECCO-09]
  - Want completely automated tool
  - $\rightsquigarrow$ More robust version: SPO^+

Sequential Parameter Optimization (SPO)

- Original SPO [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization
- Studied SPO components [Hutter et al, GECCO-09]
  - Want completely automated tool
  - $\rightsquigarrow$ More robust version: SPO$^+$
- **This work: TB-SPO, reduce computational overheads**

Sequential Parameter Optimization (SPO)

- Original SPO [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization
- Studied SPO components [Hutter et al, GECCO-09]
  - Want completely automated tool
  - $\rightsquigarrow$ More robust version: SPO$^+$
- **This work: TB-SPO, reduce computational overheads**
- Ongoing work: extend TB-SPO to handle
  - Categorical parameters
  - Multiple benchmark instances

Sequential Parameter Optimization (SPO)

- Original SPO [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization
- Studied SPO components [Hutter et al, GECCO-09]
  - Want completely automated tool
  - $\rightsquigarrow$ More robust version: SPO$^+$
- **This work: TB-SPO, reduce computational overheads**
- Ongoing work: extend TB-SPO to handle
  - Categorical parameters
  - Multiple benchmark instances
  - Very promising results for both

Outline

Outline

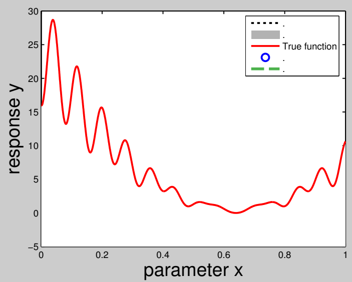
1. Sequential Model-Based Optimization

2. Reducing the Computational Overhead Due To Models

3. Conclusions

Sequential Model-Based Optimization (SMBO)

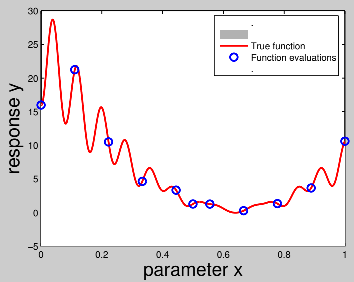
Blackbox function optimization; function = algo. performance



Sequential Model-Based Optimization (SMBO)

Blackbox function optimization; function = algo. performance

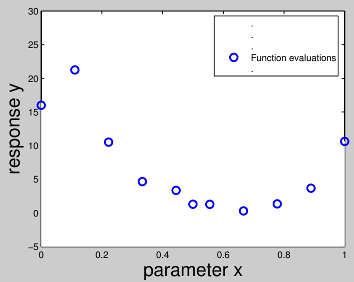
0. Run algorithm with initial parameter settings



Sequential Model-Based Optimization (SMBO)

Blackbox function optimization; function = algo. performance

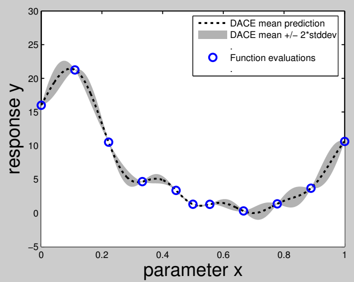
0. Run algorithm with initial parameter settings



Sequential Model-Based Optimization (SMBO)

Blackbox function optimization; function = algo. performance

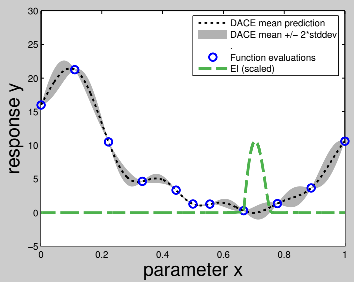
0. Run algorithm with initial parameter settings
1. Fit a model to the data



Sequential Model-Based Optimization (SMBO)

Blackbox function optimization; function = algo. performance

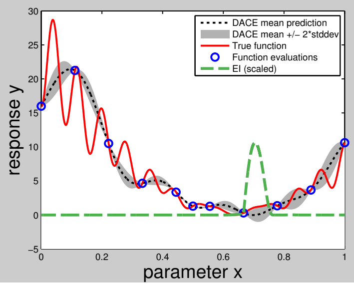
0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting



Sequential Model-Based Optimization (SMBO)

Blackbox function optimization; function = algo. performance

0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting
3. Perform an algorithm run with that parameter setting

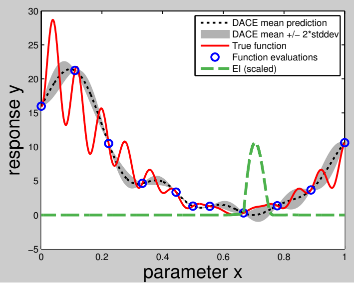


Sequential Model-Based Optimization (SMBO)

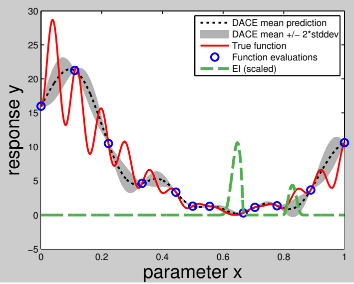
Blackbox function optimization; function = algo. performance

0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting
3. Perform an algorithm run with that parameter setting

- Repeat 1-3 until time is up



First step



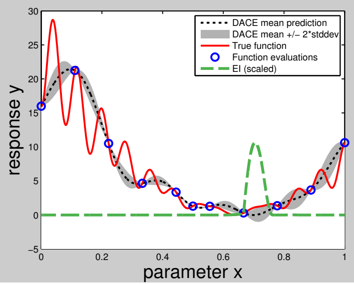
Second step

Computational Overhead due to Models: Example

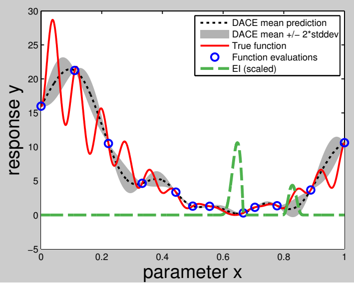
Example times

0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting
3. Perform an algorithm run with that parameter setting

- Repeat 1-3 until time is up



First step



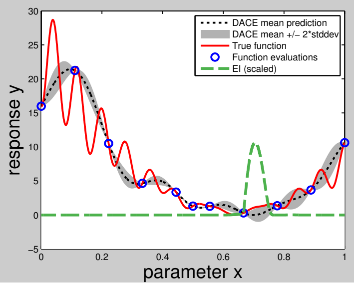
Second step

Computational Overhead due to Models: Example

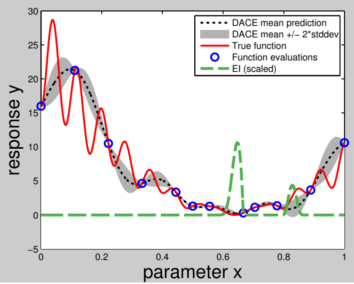
Example times

0. Run algorithm with initial parameter settings 1000s
1. Fit a model to the data
2. Use model to pick promising parameter setting
3. Perform an algorithm run with that parameter setting

- Repeat 1-3 until time is up



First step



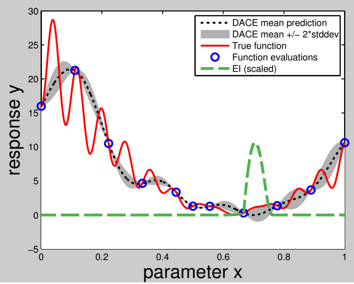
Second step

Computational Overhead due to Models: Example

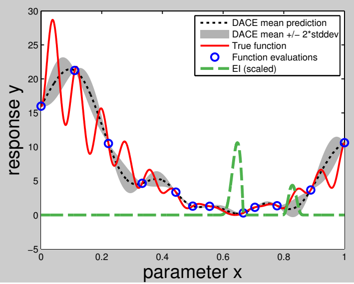
Example times

0. Run algorithm with initial parameter settings 1000s
1. Fit a model to the data 50s
2. Use model to pick promising parameter setting
3. Perform an algorithm run with that parameter setting

- Repeat 1-3 until time is up



First step



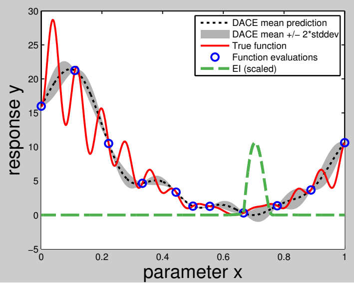
Second step

Computational Overhead due to Models: Example

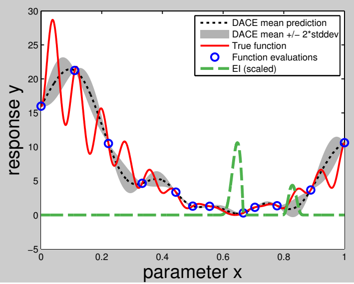
Example times

0. Run algorithm with initial parameter settings 1000s
1. Fit a model to the data 50s
2. Use model to pick promising parameter setting 20s
3. Perform an algorithm run with that parameter setting

- Repeat 1-3 until time is up



First step



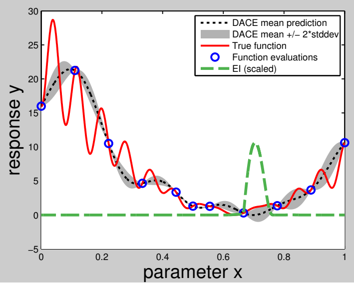
Second step

Computational Overhead due to Models: Example

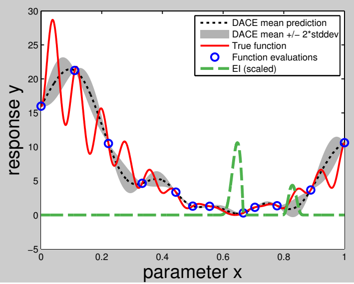
Example times

0. Run algorithm with initial parameter settings 1000s
1. Fit a model to the data 50s
2. Use model to pick promising parameter setting 20s
3. Perform an algorithm run with that parameter setting 10s

- Repeat 1-3 until time is up



First step



Second step

Outline

1. Sequential Model-Based Optimization

2. Reducing the Computational Overhead Due To Models
   - Do More Algorithm Runs To Bound Model Overhead
   - Using a Cheaper (and Better!) Model

3. Conclusions

Outline

1. Sequential Model-Based Optimization

2. Reducing the Computational Overhead Due To Models
    - Do More Algorithm Runs To Bound Model Overhead
    - Using a Cheaper (and Better!) Model

3. Conclusions

Removing the costly initial design (phase 0)

- How to choose number of param. settings in initial design?
  - Too large: take too long to evaluate all of the settings
  - Too small: poor first model, might not recover

Removing the costly initial design (phase 0)

- How to choose number of param. settings in initial design?
  - Too large: take too long to evaluate all of the settings
  - Too small: poor first model, might not recover
- Our solution: simply drop the initial design
  - Instead: interleave random settings during the search
  - Much better anytime performance

Overhead due to Models

Central SMBO algorithm loop

- Repeat: Example times
  1. Fit model using performance data gathered so far 50s
  2. Use model to select promising parameter setting 20s
  3. Perform algorithm run(s) with that parameter setting 10s

$\rightsquigarrow$ Only small fraction of time spent actually running algorithms

Overhead due to Models

Central SMBO algorithm loop

- Repeat: Example times
  1. Fit model using performance data gathered so far 50s
  2. Use model to select promising parameter setting 20s
  3. Perform algorithm run(s) with that parameter setting 10s

$\rightsquigarrow$ Only small fraction of time spent actually running algorithms

Solution 1

- Do more algorithm runs to bound model overhead
  - Select not one but many promising points (little overhead)
  - Perform runs for at least as long as phases 1 and 2 took

Which Setting to Perform How Many Runs for

Heuristic Mechanism

- Compare one configuration θ at a time to the incumbent θ_{inc}
- Stop once time bound is reached

Which Setting to Perform How Many Runs for

Heuristic Mechanism

- Compare one configuration θ at a time to the incumbent θ_{inc}
  - Use mechanism from SPO^+:
- Stop once time bound is reached

Which Setting to Perform How Many Runs for

Heuristic Mechanism

- Compare one configuration θ at a time to the incumbent θ_{inc}
  - Use mechanism from SPO$^+$:
  - Incrementally perform runs for θ until either
    - Empirical performance for θ worse than for θ_{inc} $\rightsquigarrow$ drop θ
    - Performed as many runs for θ as for θ_{inc} $\rightsquigarrow$ θ becomes new θ_{inc}
- Stop once time bound is reached

Which Setting to Perform How Many Runs for

Heuristic Mechanism

- Compare one configuration θ at a time to the incumbent θ_{inc}
  - Use mechanism from SPO^+:
  - Incrementally perform runs for θ until either
    - Empirical performance for θ worse than for θ_{inc} $\rightsquigarrow$ drop θ
    - Performed as many runs for θ as for θ_{inc} $\rightsquigarrow$ θ becomes new θ_{inc}
- Stop once time bound is reached

Algorithms

- TB-SPO
  - Get ordered list of promising parameter settings using model
  - Interleave random settings: 2nd, 4th, etc

Which Setting to Perform How Many Runs for

Heuristic Mechanism

- Compare one configuration θ at a time to the incumbent θ_{inc}
  - Use mechanism from SPO$^+$:
  - Incrementally perform runs for θ until either
    - Empirical performance for θ worse than for θ_{inc} $\rightsquigarrow$ drop θ
    - Performed as many runs for θ as for θ_{inc} $\rightsquigarrow$ θ becomes new θ_{inc}
- Stop once time bound is reached

Algorithms

- TB-SPO
  - Get ordered list of promising parameter settings using model
  - Interleave random settings: 2nd, 4th, etc
  - Compare one param. setting at a time to incumbent
  - Nice side effect: additional runs on good random settings

Which Setting to Perform How Many Runs for

Heuristic Mechanism

- Compare one configuration θ at a time to the incumbent θ_{inc}
  - Use mechanism from SPO$^+$:
  - Incrementally perform runs for θ until either
    - + Empirical performance for θ worse than for θ_{inc} $\rightsquigarrow$ drop θ
    - + Performed as many runs for θ as for θ_{inc} $\rightsquigarrow$ θ becomes new θ_{inc}
- Stop once time bound is reached

Algorithms

- TB-SPO
  - Get ordered list of promising parameter settings using model
  - Interleave random settings: 2nd, 4th, etc
  - Compare one param. setting at a time to incumbent
  - Nice side effect: additional runs on good random settings
- "Strawman" algorithm: TB-Random
  - Only use random settings
  - Compare one param. setting at a time to incumbent

Experimental validation: setup

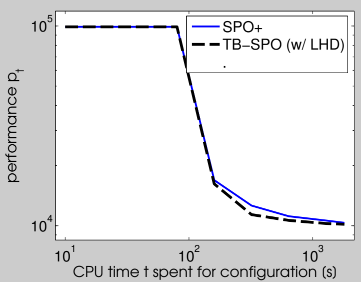
- Optimizing SLS algorithm SAPS
  - Prominent SAT solver with 4 continuous parameters
  - Previously used to evaluate parameter optimization approaches

Experimental validation: setup

- Optimizing SLS algorithm SAPS
  - Prominent SAT solver with 4 continuous parameters
  - Previously used to evaluate parameter optimization approaches
- Seven different SAT instances
  - 1 Quasigroups with holes (QWH) instance used previously
  - 3 instances from Quasigroup completion (QCP)
  - 3 instances from Graph colouring based on smallworld graphs (SWGCP)

Experimental validation: results

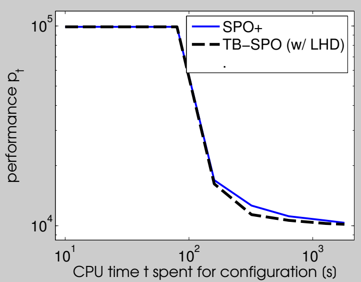
SAPS-QWH instance



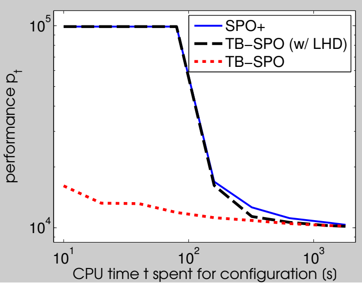
Both methods with same LHD

Experimental validation: results

SAPS-QWH instance



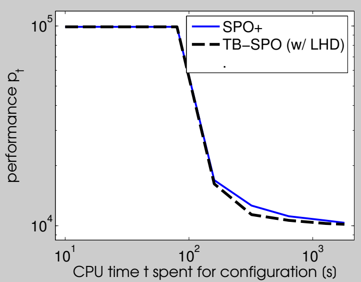
Both methods with same LHD



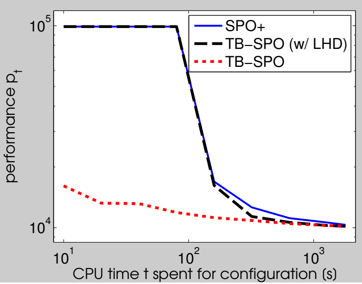
TB-SPO with empty LHD

Experimental validation: results

SAPS-QWH instance



Both methods with same LHD

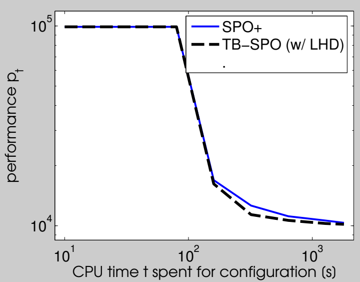


TB-SPO with empty LHD

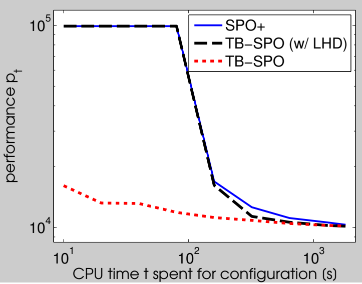
Scenario	SPO$^+$	TB-SPO	*pval1*
SAPS-QCP-MED $[\cdot 10^{-2}]$	4.50 ± 0.31	4.32 ± 0.21	$\mathbf{4 \cdot 10^{-3}}$
SAPS-QCP-Q075	3.77 ± 9.72	$\mathbf{0.19 \pm 0.02}$	$\mathbf{2 \cdot 10^{-6}}$
SAPS-QCP-Q095	49.91 ± 0.00	$\mathbf{2.20 \pm 1.17}$	$\mathbf{1 \cdot 10^{-10}}$
SAPS-QWH $[\cdot 10^{3}]$	10.7 ± 0.76	10.1 ± 0.58	$\mathbf{6 \cdot 10^{-3}}$
SAPS-SWGCP-MED	49.95 ± 0.00	0.18 ± 0.03	$\mathbf{1 \cdot 10^{-10}}$
SAPS-SWGCP-Q075	50 ± 0	0.24 ± 0.04	$\mathbf{1 \cdot 10^{-10}}$
SAPS-SWGCP-Q095	50 ± 0	$\mathbf{0.25 \pm 0.05}$	$\mathbf{1 \cdot 10^{-10}}$

Experimental validation: results

SAPS-QWH instance



Both methods with same LHD



TB-SPO with empty LHD

Scenario	SPO$^+$	TB-SPO	TB-RANDOM	*pval1*	*pval2*
SAPS-QCP-MED $[\cdot 10^{-2}]$	4.50 ± 0.31	4.32 ± 0.21	$\mathbf{4.23 \pm 0.15}$	$\mathbf{4 \cdot 10^{-3}}$	0.17
SAPS-QCP-Q075	3.77 ± 9.72	$\mathbf{0.19 \pm 0.02}$	$\mathbf{0.19 \pm 0.01}$	$\mathbf{2 \cdot 10^{-6}}$	0.78
SAPS-QCP-Q095	49.91 ± 0.00	$\mathbf{2.20 \pm 1.17}$	2.64 ± 1.24	$\mathbf{1 \cdot 10^{-10}}$	0.12
SAPS-QWH $[\cdot 10^{3}]$	10.7 ± 0.76	10.1 ± 0.58	$\mathbf{9.88 \pm 0.41}$	$\mathbf{6 \cdot 10^{-3}}$	0.14
SAPS-SWGCP-MED	49.95 ± 0.00	0.18 ± 0.03	$\mathbf{0.17 \pm 0.02}$	$\mathbf{1 \cdot 10^{-10}}$	0.37
SAPS-SWGCP-Q075	50 ± 0	0.24 ± 0.04	$\mathbf{0.22 \pm 0.03}$	$\mathbf{1 \cdot 10^{-10}}$	0.08
SAPS-SWGCP-Q095	50 ± 0	$\mathbf{0.25 \pm 0.05}$	0.28 ± 0.10	$\mathbf{1 \cdot 10^{-10}}$	0.89

Outline

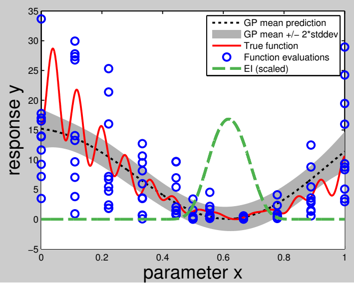
1. Sequential Model-Based Optimization

2. Reducing the Computational Overhead Due To Models
   - Do More Algorithm Runs To Bound Model Overhead
   - Using a Cheaper (and Better!) Model

3. Conclusions

2 Different GP Models for Noisy Optimization

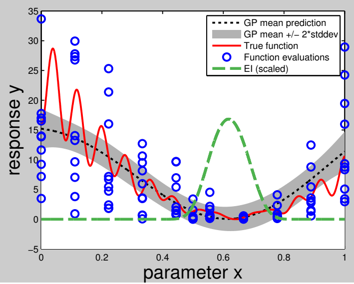
- Model I
  - Fit standard GP assuming Gaussian observation noise



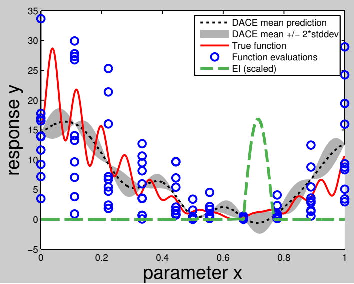
Model I: noisy fit of original response

2 Different GP Models for Noisy Optimization

- Model I
  - Fit standard GP assuming Gaussian observation noise
- Model II (used in SPO, SPO^+, and $\mathrm{TB\text{-}SPO}$)
  - Compute empirical mean of responses at each param. setting
  - Fit noise-free GP to those means



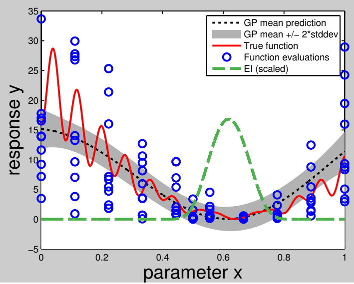
Model I: noisy fit of original response



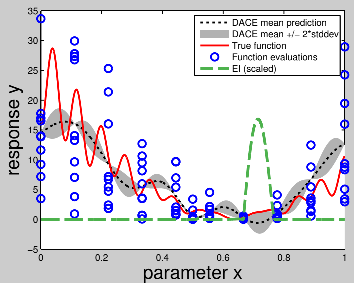
Model II: noise-free fit of empir. means

2 Different GP Models for Noisy Optimization

- Model I
  - Fit standard GP assuming Gaussian observation noise
- Model II (used in SPO, SPO^+, and TB-SPO)
  - Compute empirical mean of responses at each param. setting
  - Fit noise-free GP to those means
  - But assumes empirical means are perfect (even when based on just 1 run!)



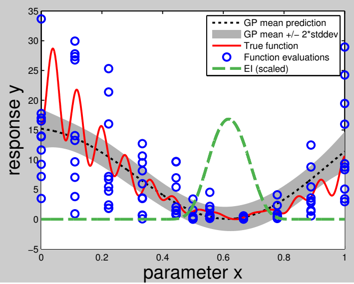
Model I: noisy fit of original response



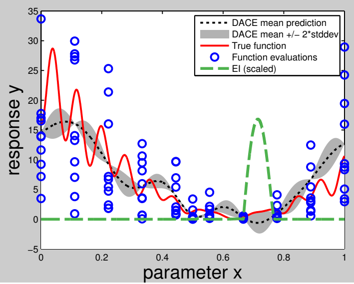
Model II: noise-free fit of empir. means

2 Different GP Models for Noisy Optimization

- Model I
  - Fit standard GP assuming Gaussian observation noise
- Model II (used in SPO, SPO^{+}, and $\mathrm{TB\text{-}SPO}$)
  - Compute empirical mean of responses at each param. setting
  - Fit noise-free GP to those means
  - But assumes empirical means are perfect (even when based on just 1 run!)
  - Cheaper (here 11 means *vs* 110 raw data points)



Model I: noisy fit of original response



Model II: noise-free fit of empir. means

How much faster is the approximate Gaussian Process?

Complexity of Gaussian process regression (GPR)

- n data points
- Basic GPR equations: inverting $n \times n$ matrix
- Numerical optimization of hyper-parameters: h steps

How much faster is the approximate Gaussian Process?

Complexity of Gaussian process regression (GPR)

- n data points
- Basic GPR equations: inverting $n \times n$ matrix
- Numerical optimization of hyper-parameters: h steps

$\rightsquigarrow$ $O(h \cdot n^3)$ for model fitting

How much faster is the approximate Gaussian Process?

Complexity of Gaussian process regression (GPR)

- n data points
- Basic GPR equations: inverting $n \times n$ matrix
- Numerical optimization of hyper-parameters: h steps

$\rightsquigarrow$ $O(h \cdot n^3)$ for model fitting

- $O(n^2)$ for each model prediction

How much faster is the approximate Gaussian Process?

Complexity of Gaussian process regression (GPR)

- n data points
- Basic GPR equations: inverting $n \times n$ matrix
- Numerical optimization of hyper-parameters: h steps

$\rightsquigarrow$ $O(h \cdot n^3)$ for model fitting

- $O(n^2)$ for each model prediction

Complexity of projected process (PP) approximation

- *Active set* of p data points $\rightsquigarrow$ only invert $p \times p$ matrix
- Throughout: use $p = 300$

How much faster is the approximate Gaussian Process?

Complexity of Gaussian process regression (GPR)

- n data points
- Basic GPR equations: inverting $n \times n$ matrix
- Numerical optimization of hyper-parameters: h steps

$\rightsquigarrow$ $O(h \cdot n^3)$ for model fitting

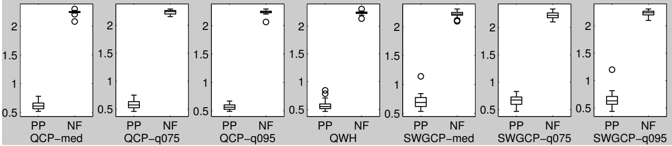
- $O(n^2)$ for each model prediction

Complexity of projected process (PP) approximation

- *Active set* of p data points $\rightsquigarrow$ only invert $p \times p$ matrix
- Throughout: use $p = 300$
- $O(n \cdot p^2 + h \cdot p^3)$ for model fitting
- $O(p^2)$ for each model prediction

Empirical Evaluation of the Model

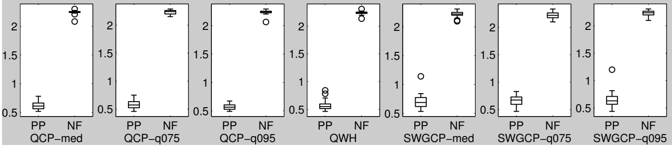
Empirical time performance (1 000 data points)



Log_{10} of CPU time (in seconds)

Empirical Evaluation of the Model

Empirical time performance (1 000 data points)



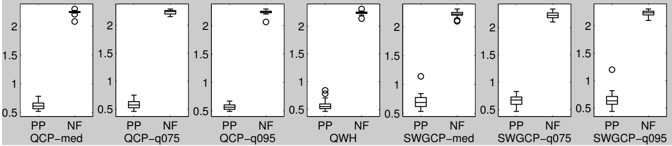
Log_{10} of CPU time (in seconds)

Empirical model quality

- Measures correlation between
  - how promising the model judges a parameter setting to be
  - true performance of that parameter setting (evaluated offline)

Empirical Evaluation of the Model

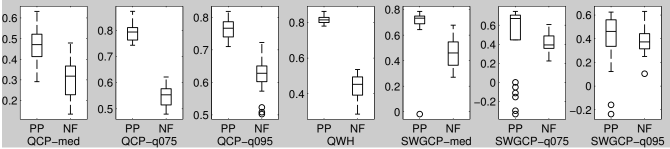
Empirical time performance (1 000 data points)



Log$_{10}$ of CPU time (in seconds)

Empirical model quality

- Measures correlation between
  - how promising the model judges a parameter setting to be
  - true performance of that parameter setting (evaluated offline)



Correlation (high is good, 1 is optimal)

Final Evaluation

- Comparing:
  - R: TB-Random
  - S: TB-SPO

Final Evaluation

- Comparing:
  - R: TB-Random
  - S: TB-SPO
  - P: TB-SPO(PP)

Final Evaluation

- Comparing:
  - R: TB-Random
  - S: TB-SPO
  - P: TB-SPO(PP)
  - F: FocusedILS (variant of ParamILS; limited by discretization)

Final Evaluation

- Comparing:
  - R: TB-Random
  - S: TB-SPO
  - P: TB-SPO(PP)
  - F: FocusedILS (variant of ParamILS; limited by discretization)

Scenario	TB-Random	TB-SPO	TB-SPO(PP)	FocusedILS
SAPS-QCP-MED $[\cdot 10^{-2}]$	4.23 ± 0.15	4.32 ± 0.21	$\mathbf{4.13 \pm 0.14}$	5.12 ± 0.41
SAPS-QCP-Q075	0.19 ± 0.01	0.19 ± 0.02	$\mathbf{0.18 \pm 0.01}$	0.24 ± 0.02
SAPS-QCP-Q095	2.64 ± 1.24	2.20 ± 1.17	$\mathbf{1.44 \pm 0.53}$	2.99 ± 3.20
SAPS-QWH $[\cdot 10^{3}]$	9.88 ± 0.41	10.1 ± 0.58	$\mathbf{9.42 \pm 0.32}$	10.6 ± 0.49
SAPS-SWGCP-MED	0.17 ± 0.02	0.18 ± 0.03	$\mathbf{0.16 \pm 0.02}$	0.27 ± 0.12
SAPS-SWGCP-Q075	0.22 ± 0.03	0.24 ± 0.04	$\mathbf{0.21 \pm 0.02}$	0.35 ± 0.08
SAPS-SWGCP-Q095	0.28 ± 0.10	0.25 ± 0.05	$\mathbf{0.23 \pm 0.05}$	0.37 ± 0.16

- TB-SPO(PP) best on all 7 instances
- Good models **do** help

Outline

1. Sequential Model-Based Optimization
2. Reducing the Computational Overhead Due To Models
3. Conclusions

Conclusions

Parameter optimization

- Can be performed by automated approaches
  - Sometimes much better than by human experts
  - Automation can cut development time & improve results

Conclusions

Parameter optimization

- Can be performed by automated approaches
  - Sometimes much better than by human experts
  - Automation can cut development time & improve results

Sequential Parameter Optimization (SPO)

- Uses predictive models of algorithm performance
- Can inform algorithm designer about parameter space

Conclusions

Parameter optimization

- Can be performed by automated approaches
  - Sometimes much better than by human experts
  - Automation can cut development time & improve results

Sequential Parameter Optimization (SPO)

- Uses predictive models of algorithm performance
- Can inform algorithm designer about parameter space

Time-Bounded SPO

- Eliminates Computational Overheads of SPO
  - No need for costly initial design
  - Bounds the time spent building and using the model
  - Uses efficient approximate Gaussian process model
  - $\rightsquigarrow$ Practical for parameter optimization in a time budget

Conclusions

Parameter optimization

- Can be performed by automated approaches
  - Sometimes much better than by human experts
  - Automation can cut development time & improve results

Sequential Parameter Optimization (SPO)

- Uses predictive models of algorithm performance
- Can inform algorithm designer about parameter space

Time-Bounded SPO

- Eliminates Computational Overheads of SPO
  - No need for costly initial design
  - Bounds the time spent building and using the model
  - Uses efficient approximate Gaussian process model
  - $\rightsquigarrow$ Practical for parameter optimization in a time budget
- Clearly outperforms previous SPO versions and ParamILS

Current & Future Work

- Generalizations of TB-SPO to handle
  - Categorical parameters
  - Multiple benchmark instances

Current & Future Work

- Generalizations of TB-SPO to handle
  - Categorical parameters
  - Multiple benchmark instances
- Applications of Automated Parameter Optimization
  - Optimization of MIP solvers [to be submitted to CP-AI-OR]

Current & Future Work

- Generalizations of TB-SPO to handle
  - Categorical parameters
  - Multiple benchmark instances
- Applications of Automated Parameter Optimization
  - Optimization of MIP solvers [to be submitted to CP-AI-OR]
- Use models to gain scientific insights
  - Importance of each parameter
  - Interaction of parameters
  - Interaction of parameters and instances features

Current & Future Work

- Generalizations of TB-SPO to handle
  - Categorical parameters
  - Multiple benchmark instances
- Applications of Automated Parameter Optimization
  - Optimization of MIP solvers [to be submitted to CP-AI-OR]
- Use models to gain scientific insights
  - Importance of each parameter
  - Interaction of parameters
  - Interaction of parameters and instances features
- Per-instance approaches
  - Build joint model of instance features and parameters
  - Given a new unseen instance:
    - Compute instance features (fast)
    - Use parameter setting predicted to be best for those features