WORHP Multi-Core Interface, Parallelisation Approaches for an NLP Solver

6th International Conference on Astrodynamics Tools and Techniques (ICATT)

Sören Geffken, Prof. Dr. Christof Büskens

Center for Industrial Mathematics - Universität Bremen

16. March 2016

Outline

- Nonlinear Optimisation with WORHP
- Technical Implementation of Multi-Core Interface
- New User Interface
- Parallel Operational Modes
- Numerical Results

Funding:
Guidance, Navigation, and Control Systems
„Extension of WORHP to multi- and many-core Architectures“
esa

Nonlinear Optimisation Problem

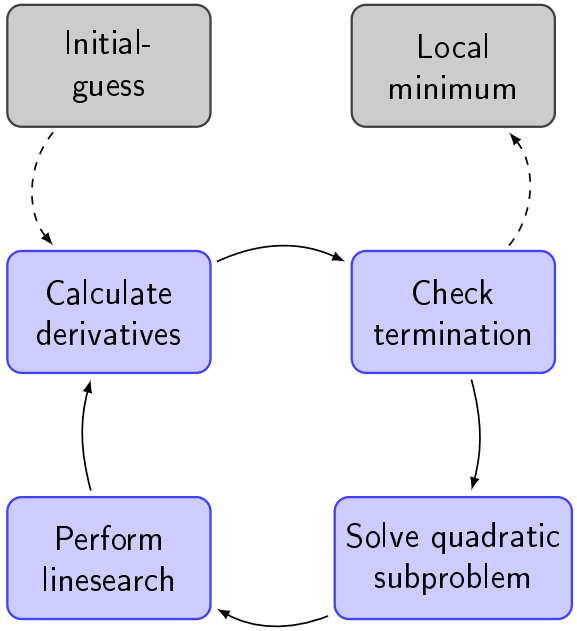
- Objective function $f : \mathbb{R}^{N_x} \to \mathbb{R}$
- Inequality constraints $g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_g}$
- Equality constraints $h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$

$$\begin{aligned} \min_{x \in \mathbb{R}^{N_x}} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

- Twice continously differentiable problem functions
- No convexity asumptions for the constraints

Solution Approach of WORHP

- Local optimisation tool
- Sequential quadratic programming
- Interior point method for quadratic subproblems



Schematic view of NLP solver using SQP method

Quadratic Subproblems

- Quadratic approximation of the problem at every iterate
- Solved using Mehrotra's predictor corrector method

$$\begin{aligned} \min_{d \in \mathbb{R}^{N_x}} \quad & \tfrac{1}{2} d^\top \nabla^2_{xx} L(x^{[k]}, \lambda^{[k]}, \mu^{[k]}) d + \nabla_x f(x^{[k]})^\top d \\ \text{unter} \quad & g(x^{[k]}) + \nabla_x g(x^{[k]})^\top d \leq 0 \\ & h(x^{[k]}) + \nabla_x h(x^{[k]})^\top d = 0 \end{aligned}$$

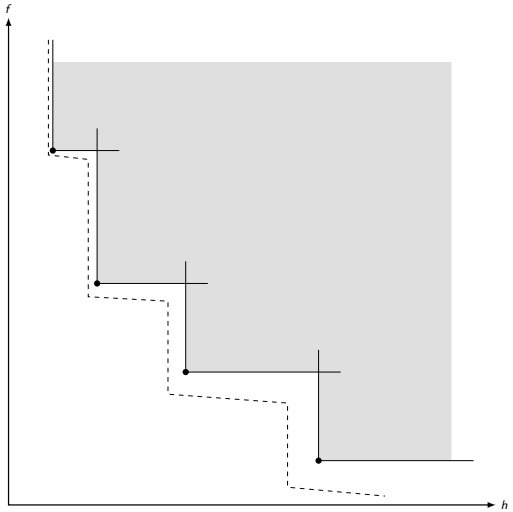
- Hessian $\nabla^2_{xx} L$ may be replaced by approximations (e.g. BFGS)
- Hessian $\nabla^2_{xx} L$ may be regularised by addition of λI, $0 < \lambda \in \mathbb{R}$
- Constraints may be relaxed

Linesearch

- Optimal solution $d^{[k]}$ of quadratic subproblems used as search direction
- Linesearch is performed

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}$$

- Filter with multiple accelerating heuristics (Default)
- Different merit function approaches implemented



Examplary filter hull with enclosing envelope

The Solver WORHP

- Sparse NLP solver
- $> 10^9$ variables
- $> 10^9$ constraints
- **Official ESA NLP solver**
- Parallel linear algebra
- 99.5 % of AMPL CUTEr testset
- 95.2 % of CUTEst testset

- Free for academics
- Industrial standard
  - ECSS-E-40 tailored
  - TRL 7 (unofficial) (TRL 1-2 scientific software)
- Interfaces
  - C/C++
  - FORTRAN
  - MATLAB
  - AMPL
- www.worhp.de

Funding:

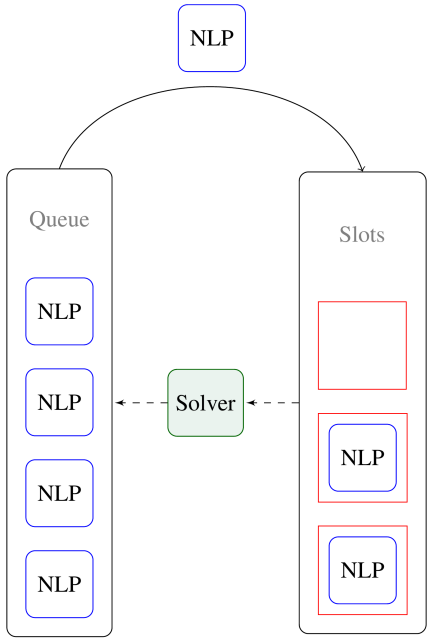
Development:

Motivation for Parallel Approach

- Multiple cores in most computers and notebooks
- SQP - ***sequential*** quadratic programming
  - Basic algorithm is inherently sequential like Newton's Method
- Different parameter settings allow to customise WORHP
  - Parallel runs of different settings highly beneficial
  - E.g. Exact Hessian vs BFGS, filter method vs merit function, feasibility refinement, multiple float parameters like internal tolerances, penalties etc.

Technical Implementation of Solver class

- Explicit threading (Boost / C++11)
- Solver Class (main thread)
  - Generates worker instances
  - Event loop
  - **Receives** notices in lock free queue
  - Upon termination of worker threads continues as defined by operational mode



Solver class organises workload for threads

Worker Threads

- Problem instance with solver data structures
  - Problem structure depending on settings, e.g. derivative structure
  - Simple and fast construction via copy constructors
- Threadfunction *solve()*
- **Send** notices to solver
- Full insight into solver due to internal reverse communication

Implementation of an NLP problem

Problem class:

- Implements $f, g, \nabla f, \nabla g, \nabla^2 L$
- Contains solver data

Prime class:

- Subclass of *Problem*
- Provides problem data, e.g. dimensions, non zeros, initial guess, bounds and derivative structure
- Implements copy constructor

Automatic Parameter Variations

Pattern file

- Blocks of interchangeable settings
- Optional definition of requirements
- Predefined file with common used settings

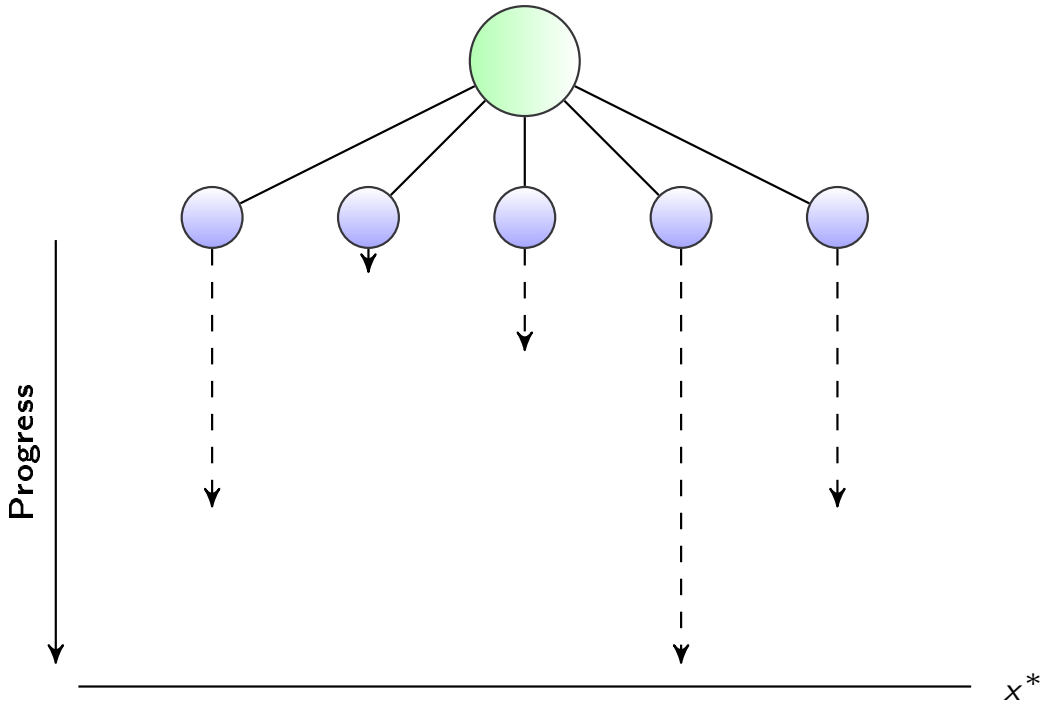
```
//###################################
// This file is organised as follows:
// (Setting) (optional: Requirement)
//###################################
## General Method Parameters
(par->NLPmethod = 3) (par->qp.ipLsMethod == 1)
(par->NLPmethod = 1) ()
## Number of Relaxation Variables
(par->MoreRelax = false) (opt->m > 0)
(par->MoreRelax = true) (opt->m > 0)
## ScaledKKT for Thoroughness
(par->ScaledKKT = true) ()
(par->ScaledKKT = false) ()
```

Possible settings in script language for automatic code generation of parameter variations

First-Across-The-Line Mode

NLP-Solver WORHP

Worker instances



Performance on CUTEst

Method comparison

Method	Optimal	Acceptable	Unsuccessful
Serial	1065	19	48
WMCI	1084	8	40
Change	+19	-11	-8

Thread efficiency

Thread	Options	Solved	%
1	Filter	290	25.62
2	Filter, Feas. Ref.	236	20.85
3	Merit Fct.	244	21.55
4	Filter, BFGS	362	31.98

Timing comparison

	Complete (1132)	Filtered (784)
Serial	526 min	279 min
WMCI	360 min	207 min
Speed-up	1.46	1.35

Parameter Identification Mode

Motivation:

- Applying an optimisation tool is an iterative process
- Identify beneficial parameters for concrete problem

Workflow:

- Test multiple parameter settings
- Evaluate different user defined criteria e.g. speed or solution quality
- Compare results upon termination of all tasks

Application of Parameter Identificaton Mode

- Test all possible combinations
- Evaluate results with respect to speed and quality

Setting	Value 1	Value 2
Linesearch method	Filter	Merit function
Feasibility Refinement	On	Off
Hessian	Analytic	BFGS

```
##        WMCI parameter identification mode          ##
### Least major iterations required by these settings ###

Thread 2:
---------

par->NLPmethod = 1;
par->UserHM = true;
par->RefineFeasibility = 2;

Final values after iteration 7:
Final objective value ..............  1.4290199970E+05
Final constraint violation ........  1.1559653998E-07
Final KKT conditions ...............  9.2700494022E-09
Successful termination: Optimal Solution Found.
-----------------------------------------------------------

 ### Best objective value achieved by these settings ###

Thread 1:
---------

par->NLPmethod = 3;
par->UserHM = true;
par->RefineFeasibility = 2;

Final values after iteration 18:
Final objective value ..............  1.3485126076E+05
Final constraint violation ........  7.0624611073E-07
Final KKT conditions ...............  4.5137218774E-10
Successful termination: Optimal Solution Found.
-----------------------------------------------------------
```

Results of parameter identification mode for DTOC6 example

Summary

Technical Realisation:
- Solver class and worker threads
- Object oriented user interface
- Code generation for parameter variations

Parameter Identification Mode:
- Adapt solver to specific problem
- Improves iterative application of solver
- Increase understanding of solver parameters

First-Across-The-Line Mode:
- Increases robustness and speed
- At least as good as serial run
- Increased memory requirements
- Parallel models or linear algebra difficult

Outlook

- Globalisation via initial guess variation
- Solve different problems in parallel
  - Mixed integer optimisation
  - Pareto fronts
- Inclusion within TransWORHP (WORHP's companion transcriptor method)