HOPSPACK

Software Framework for Parallel Derivative-Free Optimization

August 24, 2009

Todd Plantenga, Tamara Kolda
Sandia National Laboratories, CA

SAND 2009-5016C

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Outline

1. What is HOPSPACK?
2. How can I solve problems?
3. How can I write my own solver?

Thanks to:

Department of Energy ASCR Program

Joshua Griffin (SAS Institute)

Genetha Gray (Sandia National Laboratories)

Michael Lewis (William & Mary)

Others…

What is HOPSPACK?

- Hybrid Optimization Parallel Search PACKage
- Software framework for solving derivative-free optimization problems
- What does “hybrid” mean?
  - Run multiple solvers concurrently
  - Combine global and local search methods
- Successor to APPSPACK (1999-2009)

Key Features of HOPSPACK

- Parallel evaluations
- Generating Set Search solver (extended)
- Platform support
  - Linux, Mac OSX, Windows Visual Studio
  - MPI for distributed machine architectures
  - Multithreading (MT) for multiple processors/cores
- Framework for new solvers
  - Parallel evaluations, trial point results cached
  - Solvers can initiate and control subproblem solvers
  - C++ code, open source license

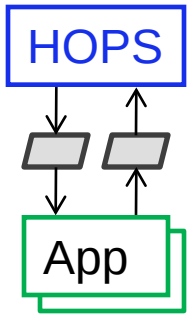
How Can I Solve Problems?

Download HOPSPACK binary

- Multithreaded or serial (no threads)
- MPI version needs to be compiled

Edit a configuration file

- Define bounds, linear constraints, start point
- Specify solvers and their parameters



Wrap application as a separate process

- Communicate via input/output files
- Examples provided

Problem Space

- Derivative-free optimization problems

$$
\begin{array}{lcll}
\text{minimize} & f(x) & f & \text{nonlinear} \\
\text{subject to} & l \leq x \leq u & x & \text{real, integral} \\
 & A_E\, x = b_E & & \\
 & A_I\, x \geq b_I & & \\
 & c_E(x) = 0 & c_E & \text{nonlinear} \\
 & c_I(x) \geq 0 & c_I & \text{nonlinear}
\end{array}
$$

- Functions can be **noisy**, **nonsmooth**, **nonconvex**, **expensive** to compute

Generating Set Search (GSS)

- GSS based on pattern search
  - Mature solver code (initially APPSPACK)
  - Extended for nonlinear constraints and more
- Asynchronous implementation
  - Excellent load balancing
  - Robust to evaluation failure (e.g., *f(x)* undefined)
  - Strong convergence properties for smooth functions
  - Works well for nonsmooth functions
    - "Comparison of Derivative-Free Optimization Methods for Groundwater Supply and Hydraulic Capture **Community Problems**", many authors, 2008, Elsevier.

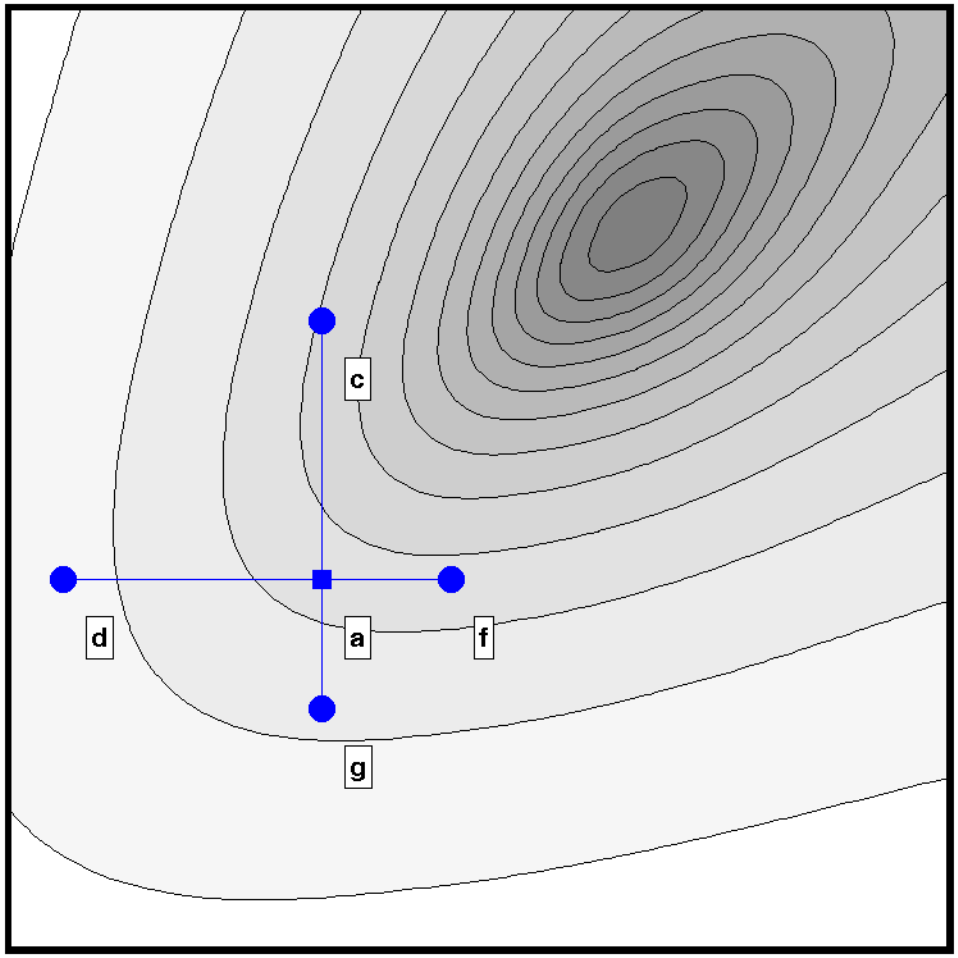
GSS Search Pattern

Best point (a) in center

Search in all directions (c,d)

Shorten when results are worse (f,g)

Can jump to better "oracle" point

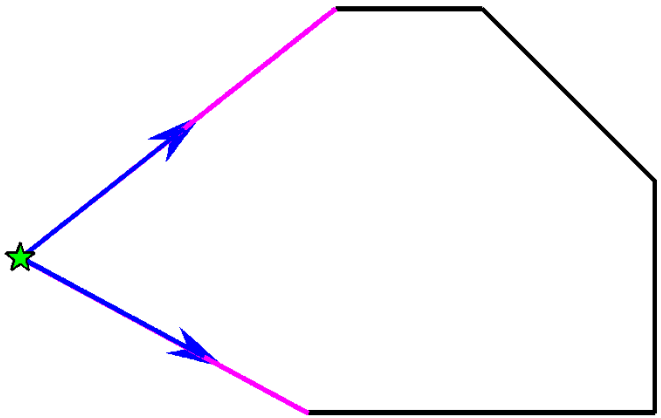


GSS with Linear Constraints

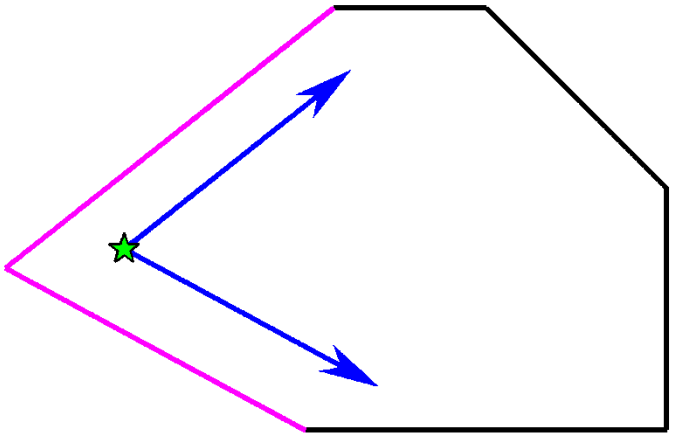
- All trial points linearly feasible
- Project points onto equality constraints
  - Computation methods available for any solver
- Search directions conform to local constraints
  - Span feasible region
  - Robust computation with CDDLIB (Komei Fukuda)
- Snap to active inequalities (within ε)
- LAPACK for linear algebra
  - Netlib, Intel MKL, etc.

GSS Conforming Directions

Search can move parallel to active constraints.



Search can move parallel to “nearby” constraints.



GSS for Nonlinear Constraints

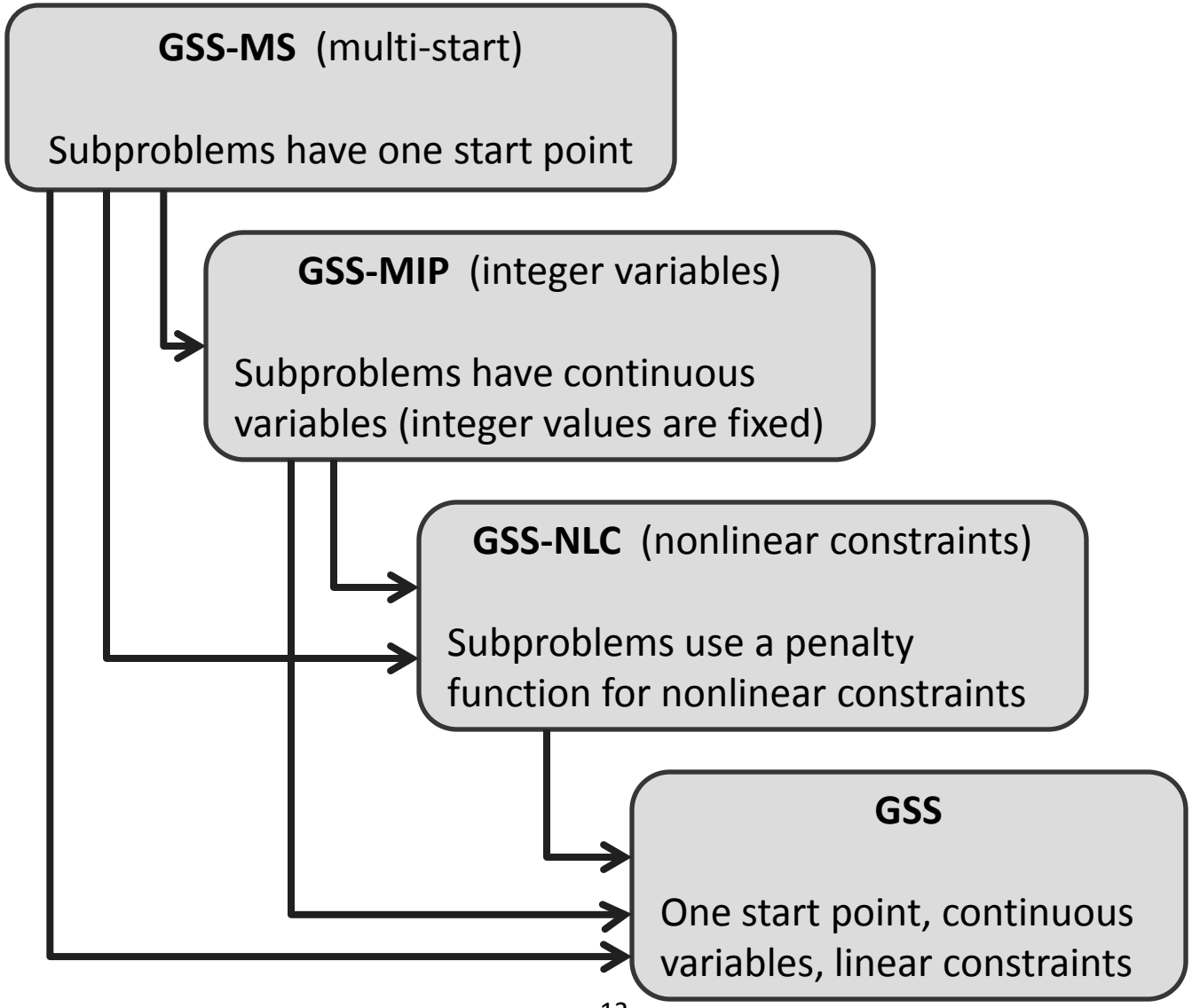
- Choice of penalty methods

$$
\begin{aligned}
\ell_1 : &\quad P_c(x,\rho) = \rho \sum |c_i(x)| \\
\ell_\infty : &\quad P_c(x,\rho) = \rho \max |c_i(x)| \\
\ell_2 : &\quad P_c(x,\rho) = \rho \|c(x)\| \\
\ell_2^2 : &\quad P_c(x,\rho) = \rho \|c(x)\|^2
\end{aligned}
$$

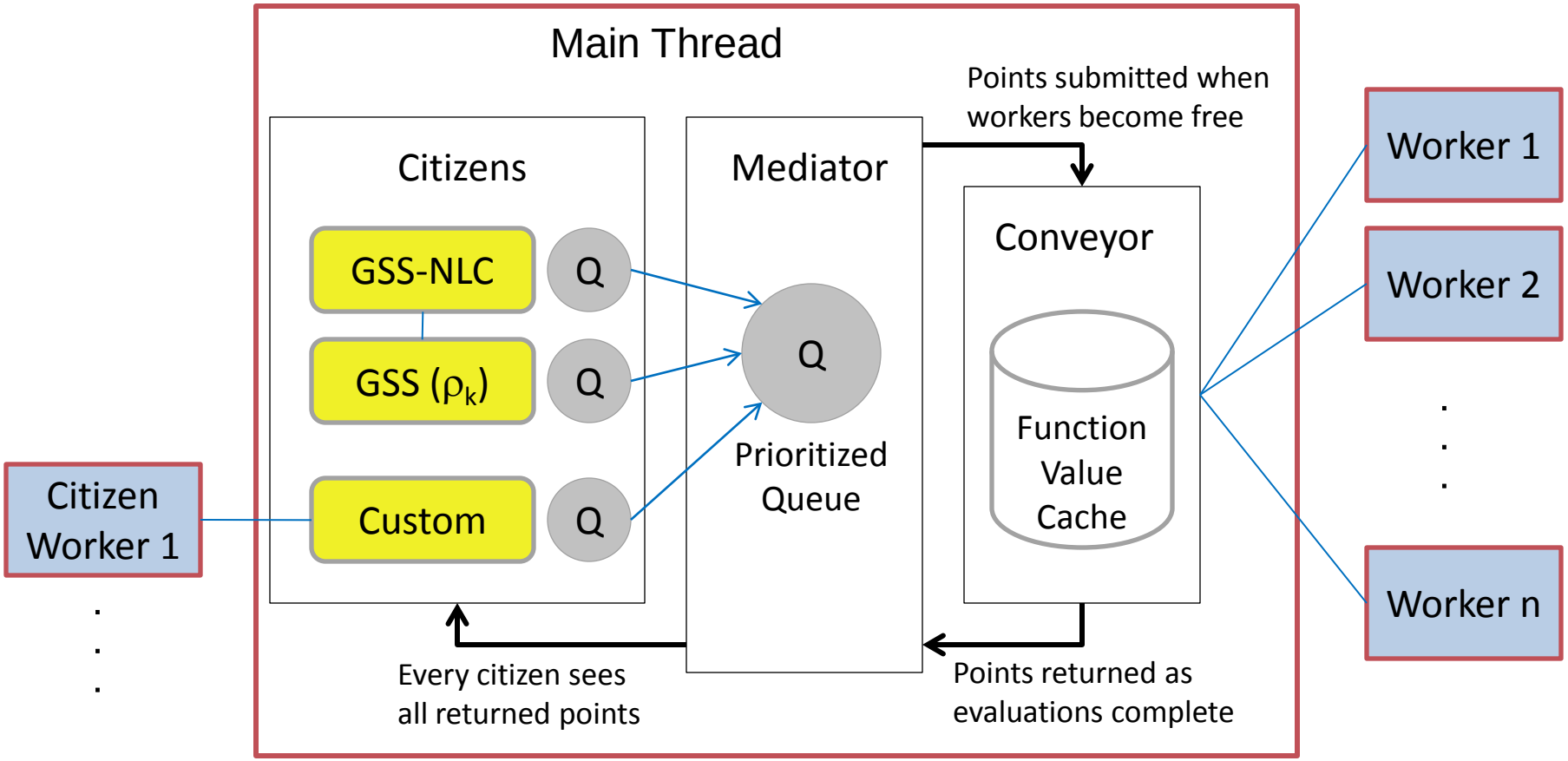
- Increase penalty parameter ρ_k and iterate
  - Solve linearly constrained subproblem with GSS
  - Subproblem convergence becomes tighter

Nonlinearly-constrained optimization using asynchronous parallel generating set search (T. Kolda, J. Griffin). SAND2007-3257, 2007

Hierarchy of GSS Solvers



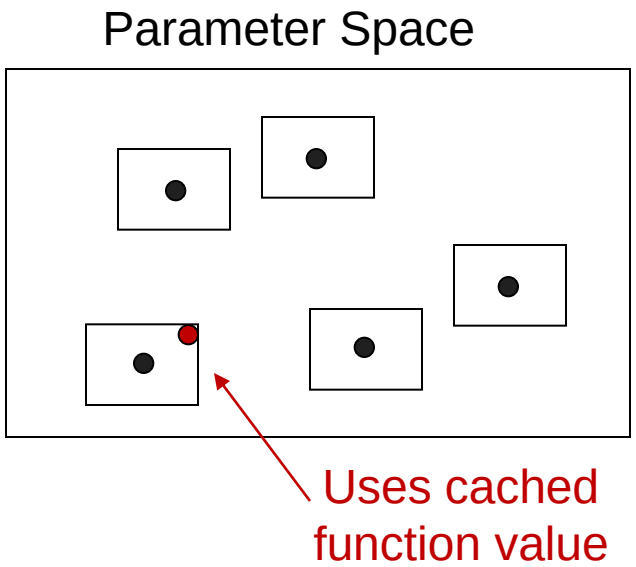
HOPSPACK Architecture



Evaluation Results Cache

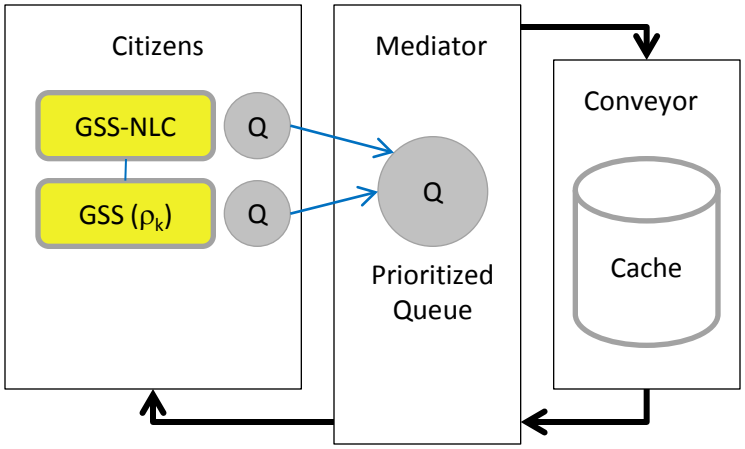
- Program memory cache
  - Fast lookups
  - Cache "hit" range τ

$$\left\| \frac{x_i - y_i}{s_i} \right\|_\infty \leq \tau$$



- Benefits
  - Fewer evaluations (10-40% hit rate)
  - Save to file for restart
- Pending Cache for evaluations in progress

Citizen/Solver Execution



- Citizen iteration
  - Receive list of results
    - Asynchronous
    - From all citizens (configurable)
  - Process (quickly)
  - May launch a subproblem solver
  - Return list of trial points (new or renewed)
- Mediator coordinates
  - Submitted points merged, prioritized
  - Apply stop criteria

Write Your Own Citizen/Solver

- Why write a solver for HOPSPACK?
  - Parallel evaluations (MPI and MT)
  - Cache, Pending Cache
  - Call other solvers for subproblems
  - Linear constraint services: feasibility, projection, snap

How Can I Write My Own Solver?

- Extend Citizen base class
  - Methods: **`preProcess(), exchange(), postProcess()`**
  - Think about running asynchronously
  - Think about using “external / oracle” points
  - Think about providing “external / oracle” points
- Example: use GA to direct local searches (LBNL)
  - Add new citizen class
  - Call GSS as subproblem solvers

Extending HOPSPACK

Several C++ interfaces designed for extension

<u>Examples</u>

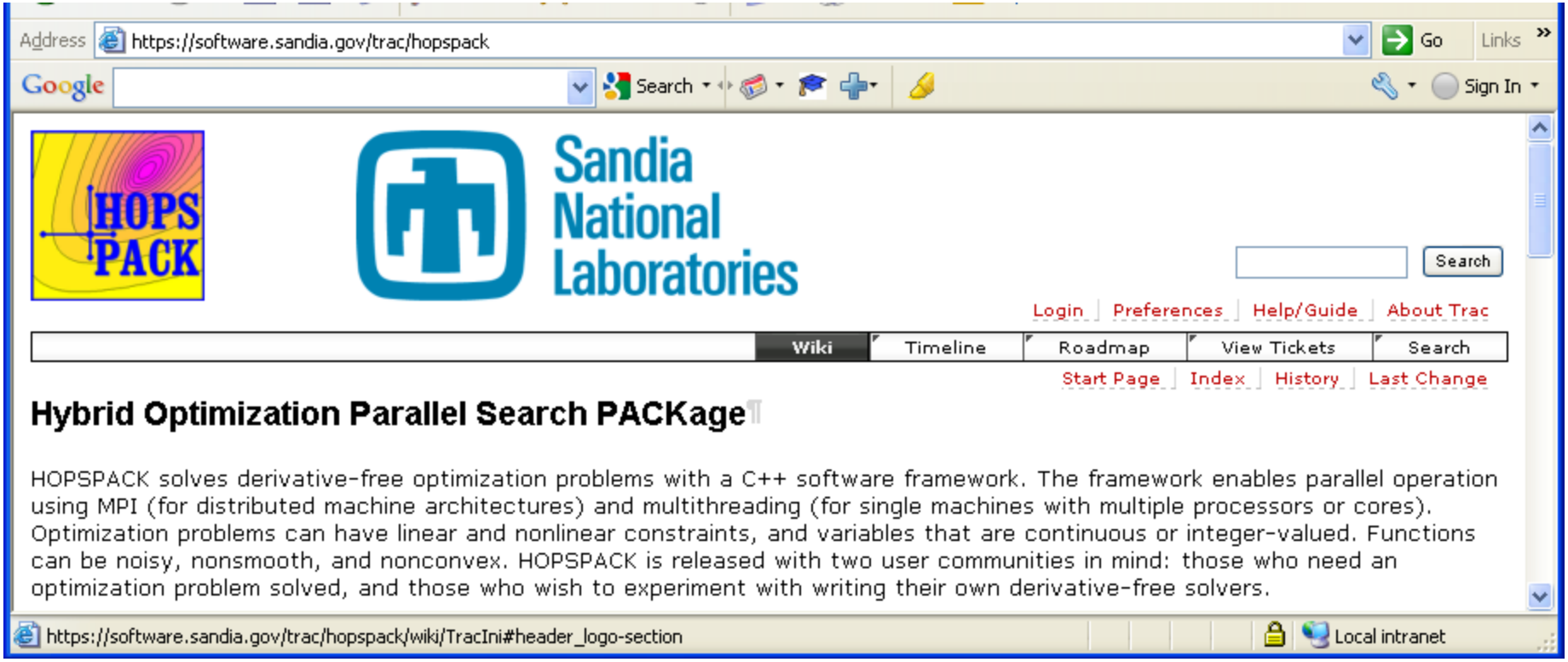
- Combine MPI with multithreading (LBNL)
  - Subclass Executor
  - Alter main()
- Solve problems in AMPL (SNL)
  - Subclass Evaluator to use AMPL Solver Library
  - MT version must be thread safe

Build with CMake

- Open source (http://cmake.org)
  - Download ~5M binary, unzip
  - Embraced at Sandia (Trilinos), VTK
- Automatically finds native C++ compiler
  - Override to use Intel C++ compiler, etc.
- Finds MPI compiler, LAPACK library, …
- Creates native “make” files, dependencies
  - Manually run “make”, Visual Studio, etc.
  - Supports multiple “out of source” builds

Releasing Soon

- https://software.sandia.gov/trac/hopspack
- SVN repository, Wiki, binaries



Summary

- Why use HOPSPACK for an application?
  - Parallel evaluations, MPI or MT
  - Extended GSS solver
  - Binaries for Linux, Windows, Mac OSX
- Why write a solver using HOPSPACK?
  - Framework for parallel evaluations, MPI or MT
  - Use subproblem solvers
  - Build with any C++ compiler, CMake

Email: tplante@sandia.gov

https://software.sandia.gov/trac/hopspack

References

- **Nonlinearly-constrained optimization using asynchronous parallel generating set search** (T. Kolda, J. Griffin). SAND2007-3257, May 2007.
- **Asynchronous parallel generating set search for linearly-constrained optimization** (T. G. Kolda, J. D. Griffin and R. M. Lewis). SISC, 30(4):1892-1924, May 2008.
- **Stationarity Results for Generating Set Search for Linearly Constrained Optimization** (T. G. Kolda, R. M. Lewis and V. Torczon). SIOPT 17(4):943-968, Nov. 2006.
- **A comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems** (K. R. Fowler et al.). Advances in Water Resources, 31:743-757, May 2008.
- **Algorithm 856: APPSPACK 4.0: Asynchronous Parallel Pattern Search for Derivative-Free Optimization** (T. G. Kolda, G. A. Gray). ACM TOMS, 32(3):485-507, Sept. 2006.
- **Revisiting asynchronous parallel pattern search for nonlinear optimization** (T. G. Kolda). SIOPT, 16(2):563-586, Dec. 2005.