APPSPACK

A Software Package for Asynchronous and Fault-Tolerant Parallel Pattern Search for Derivative-Free Optimization

Tamara G. Kolda
Sandia National Labs

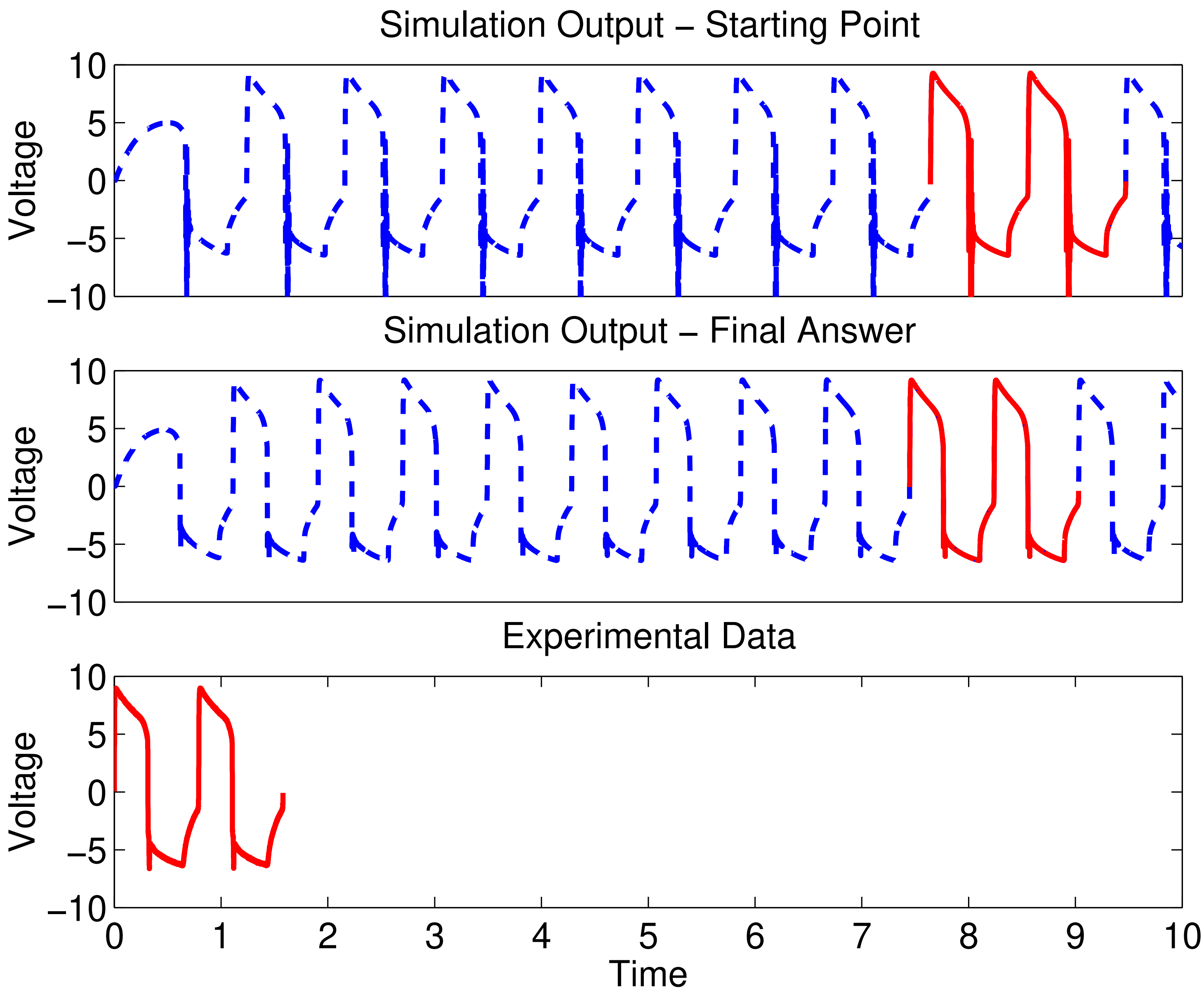
Engineering Example: Determining the Characteristics of a Circuit

- **Collaborator:** Ken Marx (SNL/CA, 8418)
- **Variables:** inductances, capacitances, diode saturation currents, transistor gains, leakage inductances, and transformer core parameters
- **Simulation Code:** SPICE3

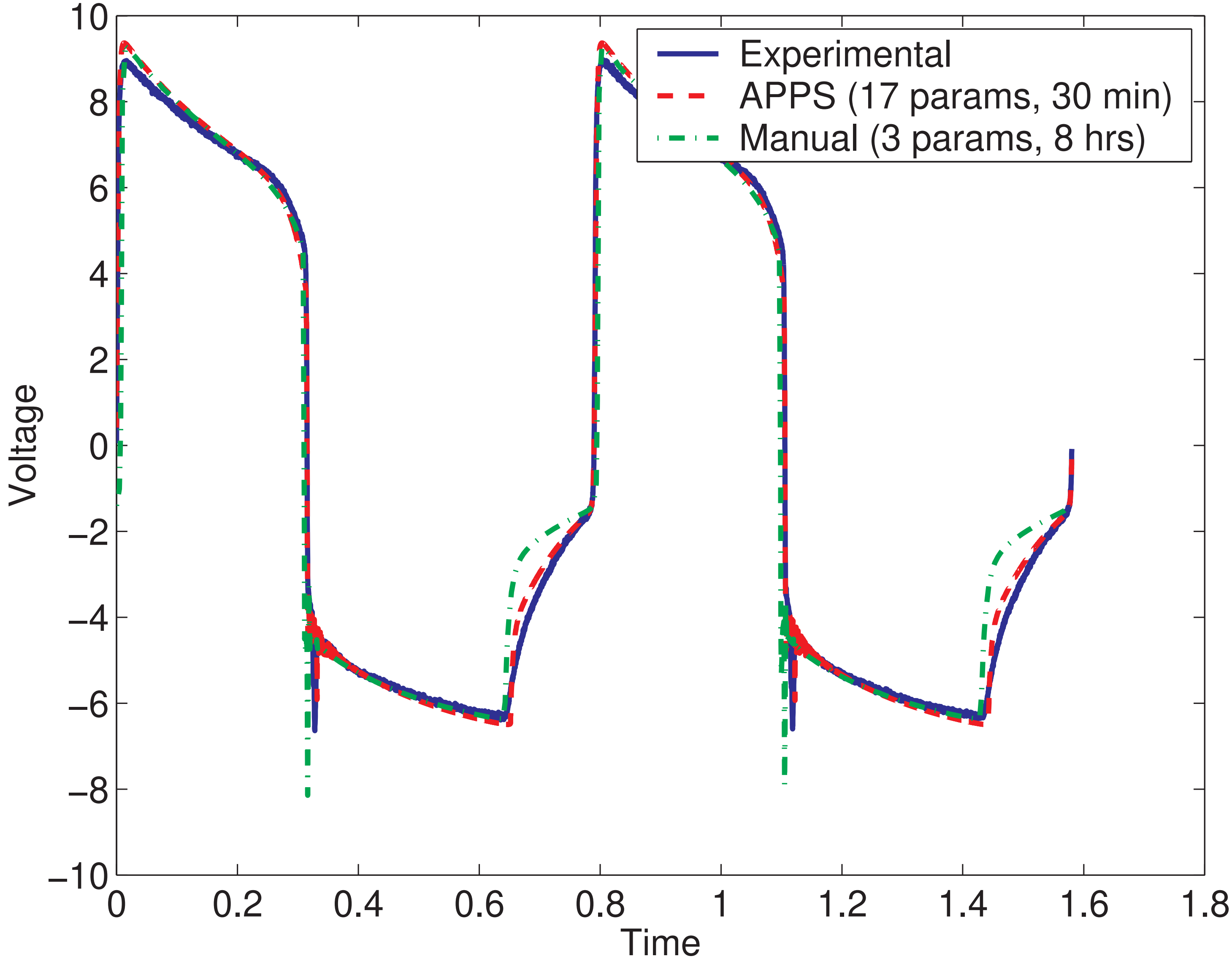
$$f(x) = \sum_{t=1}^{N} \left(V_t^{\mathrm{SIM}}(x) - V_t^{\mathrm{EXP}} \right)^2 ,$$

$$\begin{aligned}
x &= \text{17 unknown characteristics} \\
V_t^{\mathrm{SIM}}(x) &= \text{Simulation voltage at time } t \\
V_t^{\mathrm{EXP}} &= \text{Experimental voltage at time } t \\
N &= \text{Number of time steps (2700)}
\end{aligned}$$

Extracting Data from SPICE3 Output



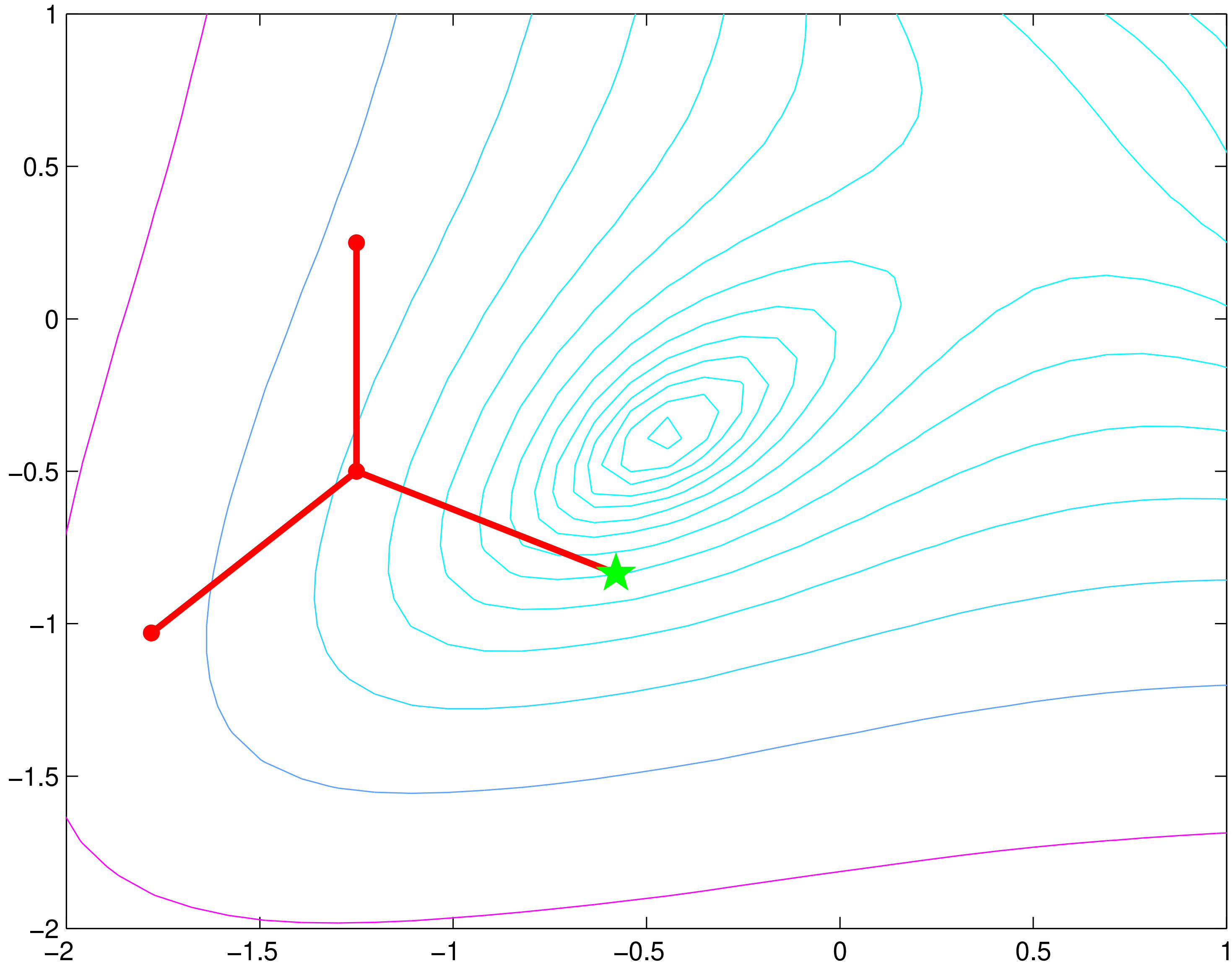
APPS Solution is Faster and More Accurate than Hand-Tuning



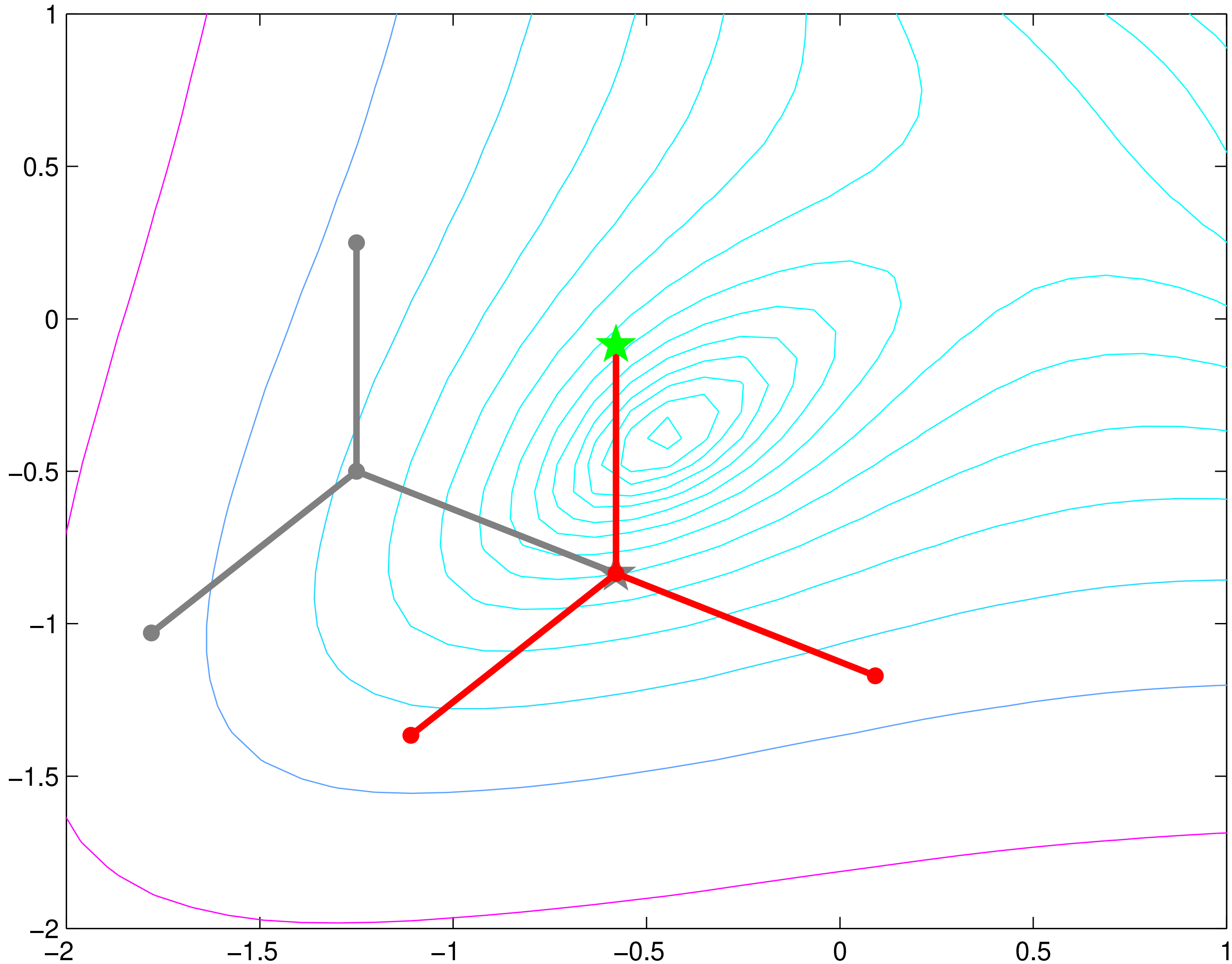
Pattern Search Methods Only Need Function Values — No Derivatives

- Pattern search was very popular in the 1960s
  - Box, Hooke & Jeeves, etc.
- The 1990s saw a renewal of interest due to . . .
  - Parallelizability of Pattern Search
    PDS: Torczon (1992)
  - Theoretical Convergence Results for Pattern Search
    Unconstrained: Torczon (1991, 1997)
    Constrained: Lewis & Torczon (1999, 2000)

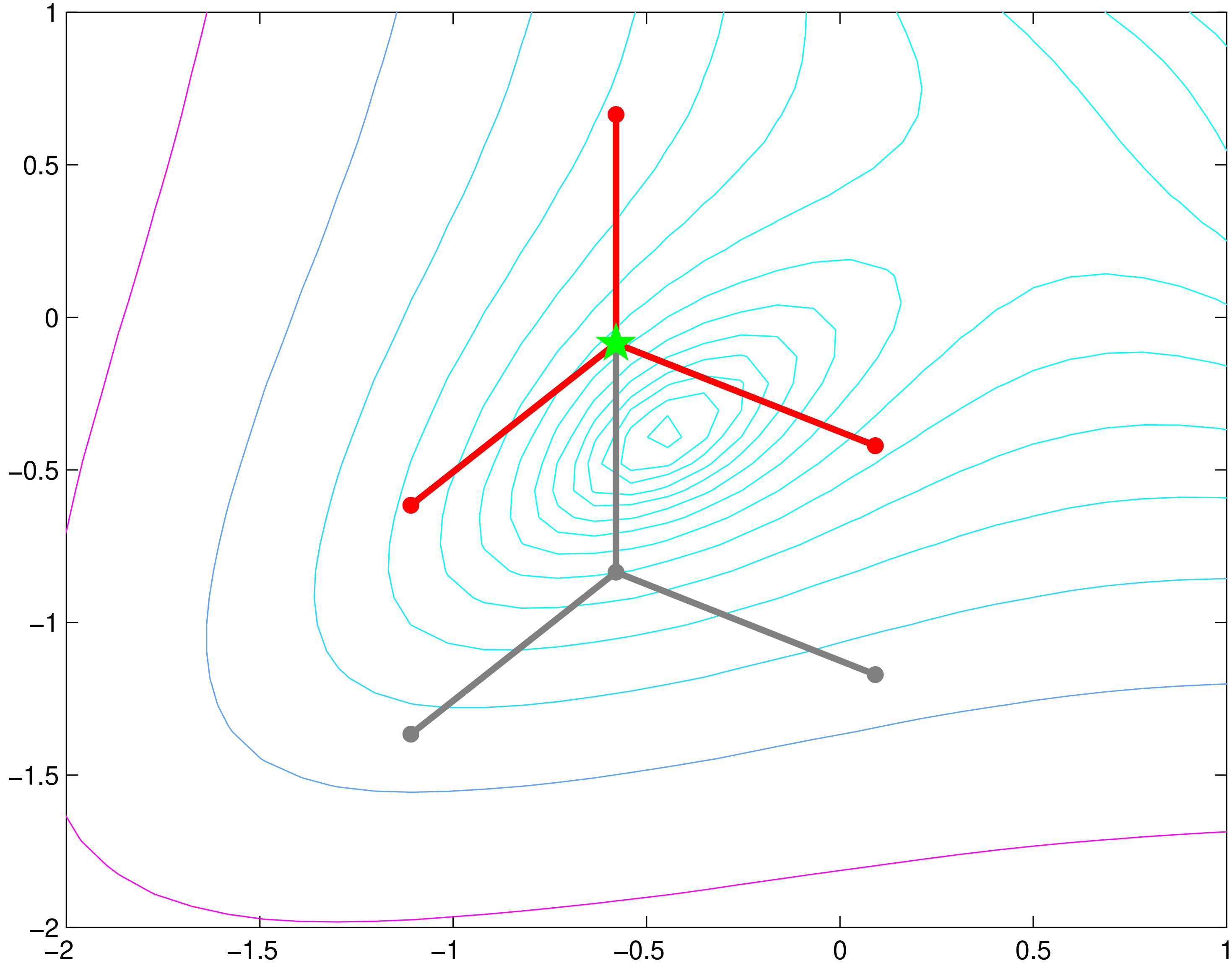
Parallel Pattern Search



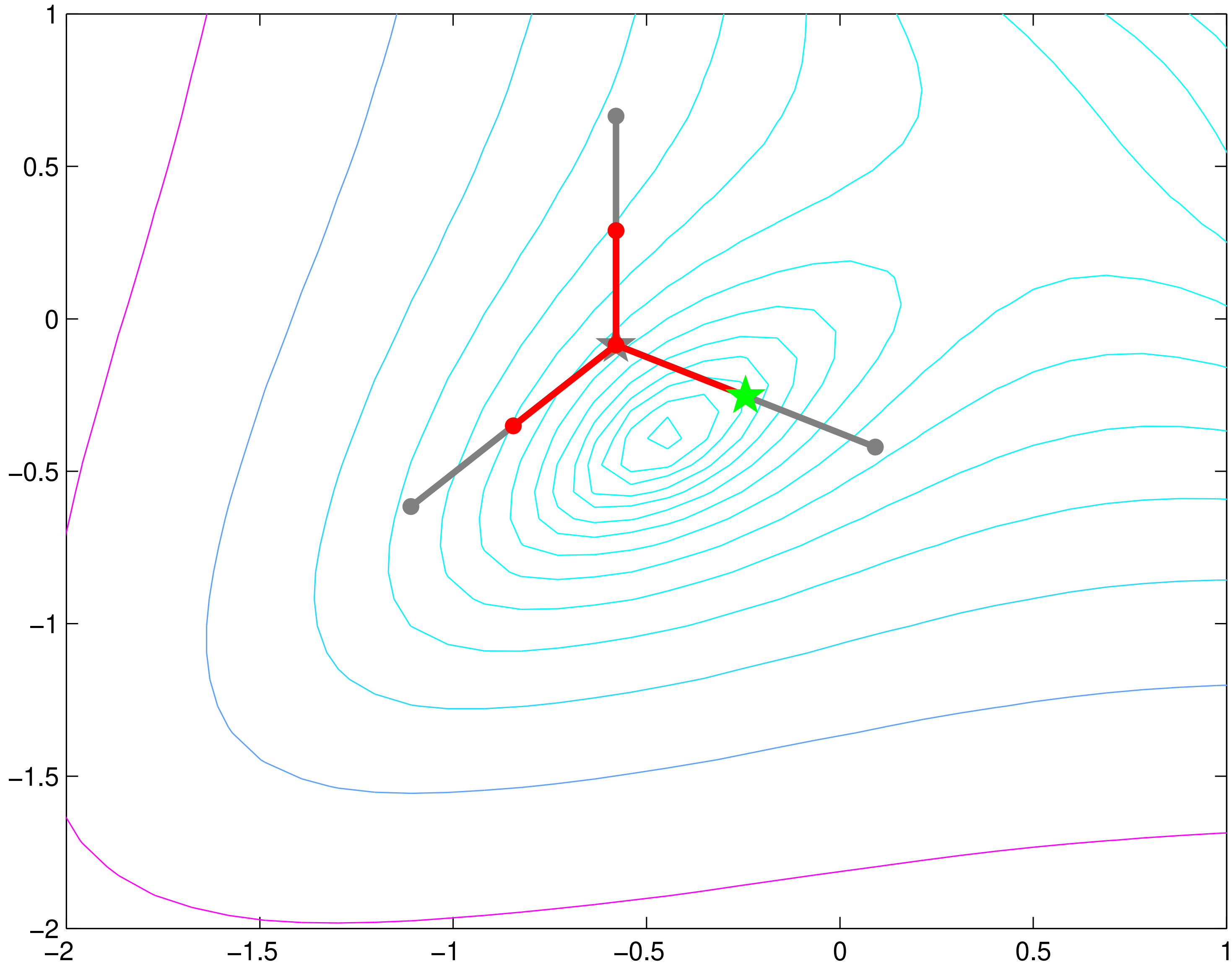
Parallel Pattern Search



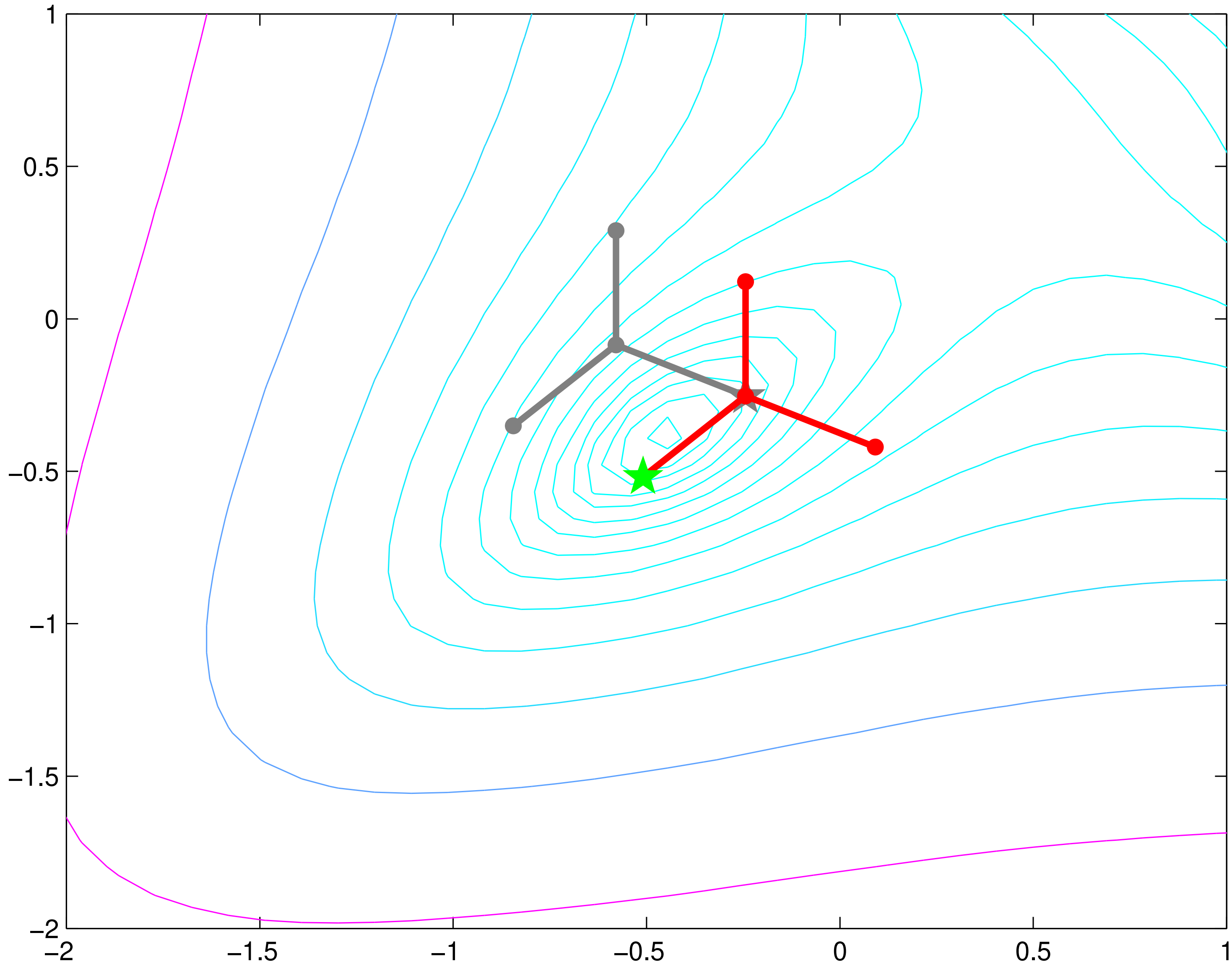
Parallel Pattern Search



Parallel Pattern Search



Parallel Pattern Search



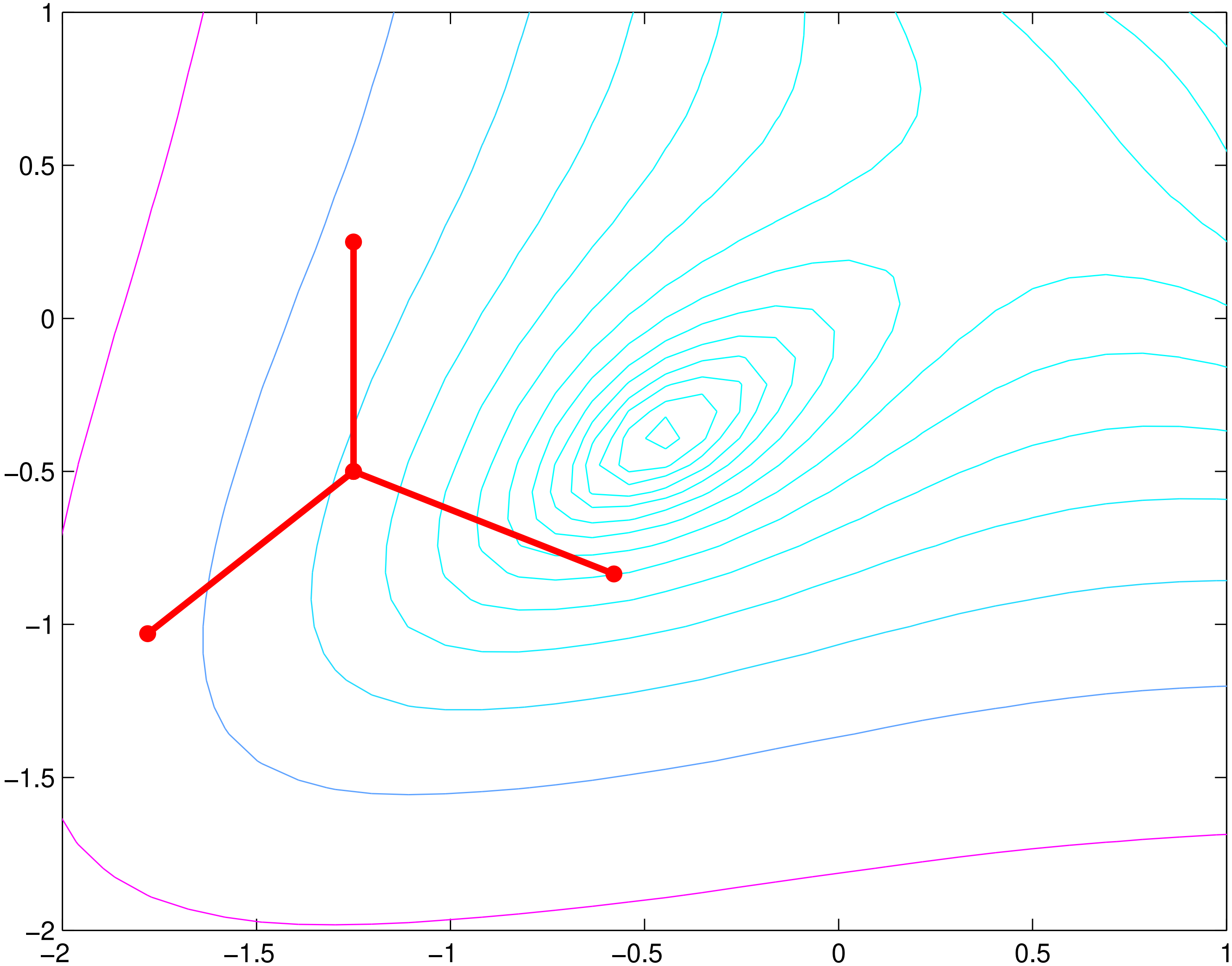
Asynchronous PPS Reduces Idle Time

- Remove synchronization bottleneck
  - Different running times for simulations
  - Varying processor loads
  - Heterogeneous architectures
- Easier to incorporate fault tolerance
  - COTS systems

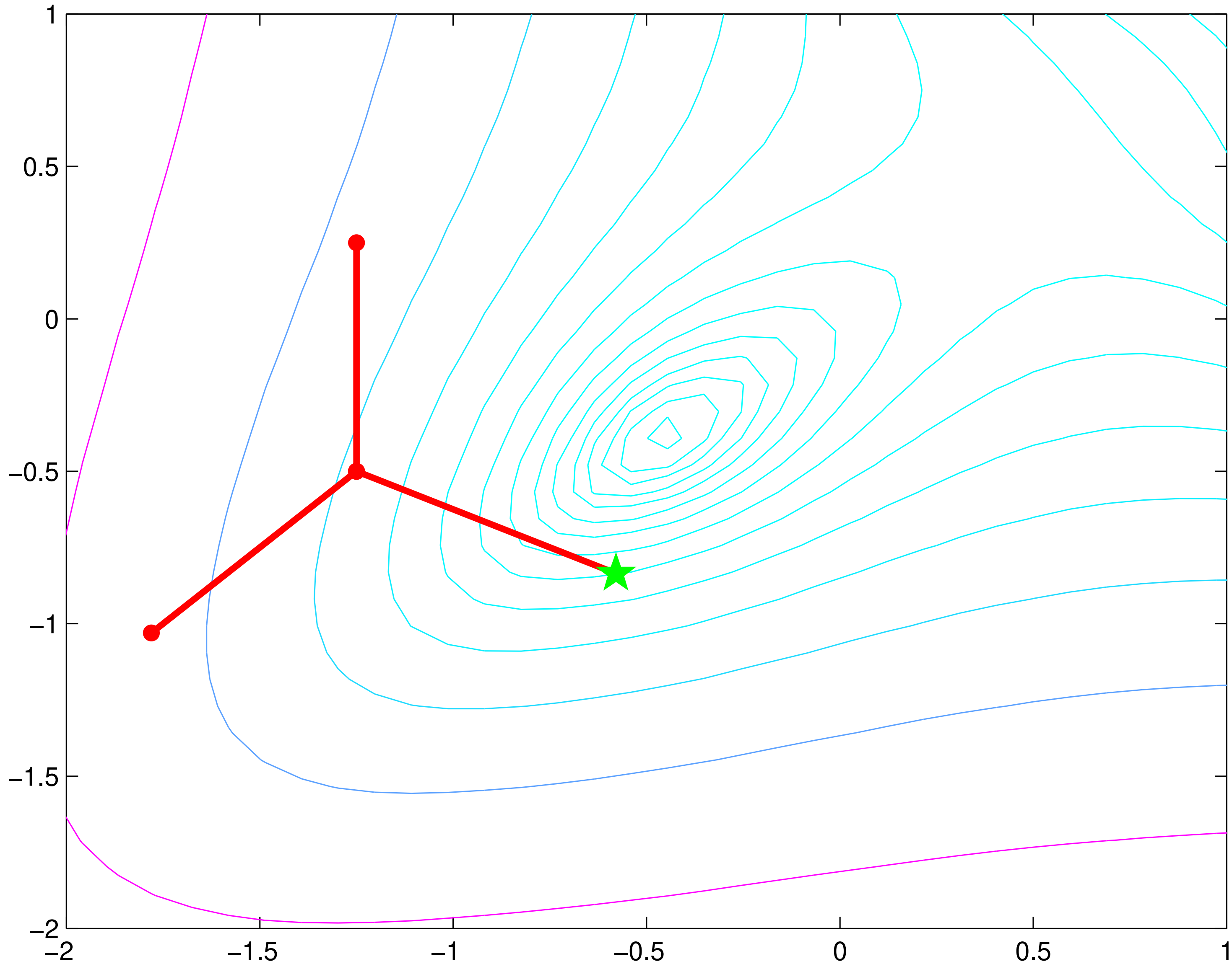
APPS/APPSPACK Collaborators

- Patty Hough
- Virginia Torczon
- Alton Patrick (Summer Intern)
- Sarah Brown (Summer Intern)

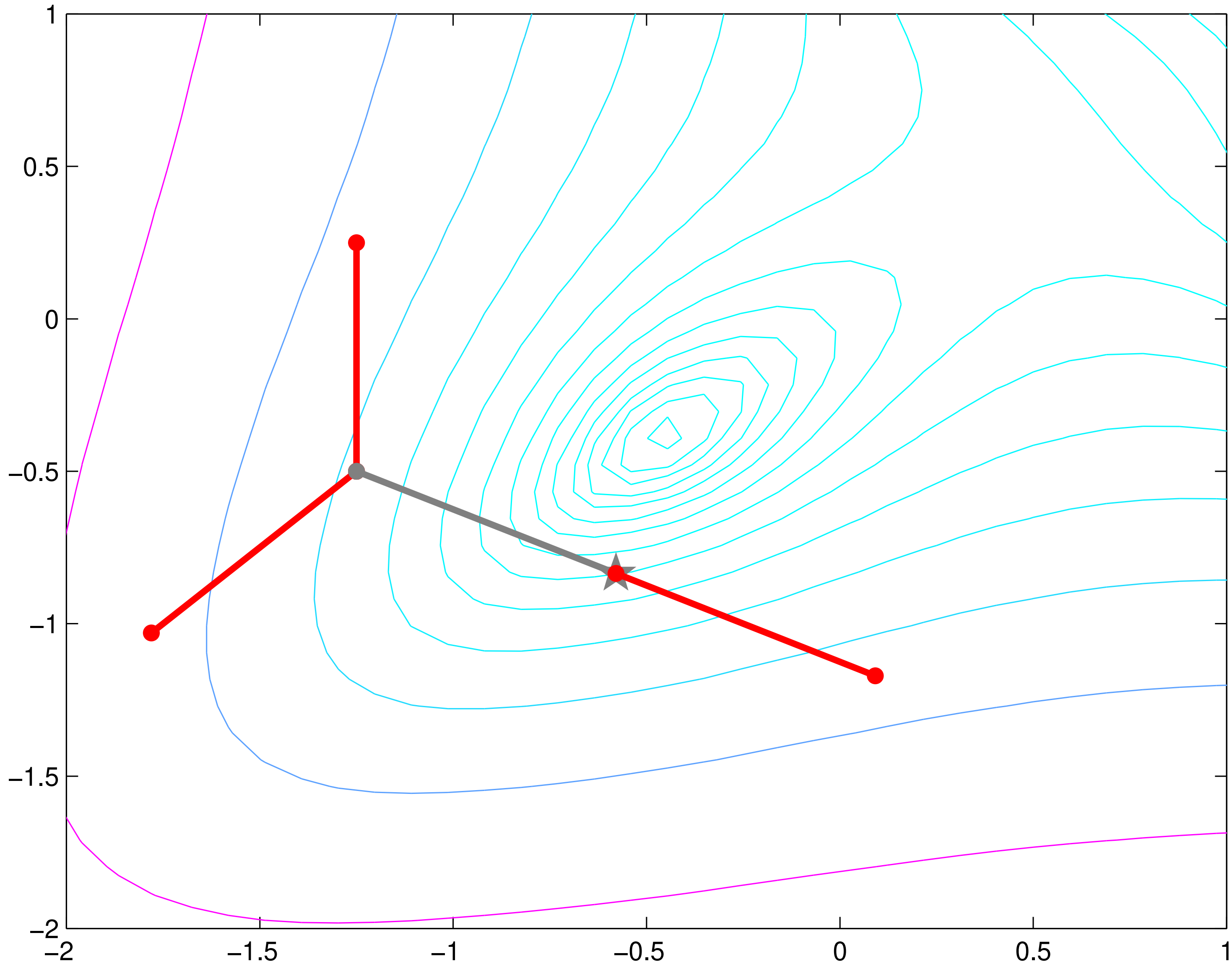
Asynchronous Parallel Pattern Search



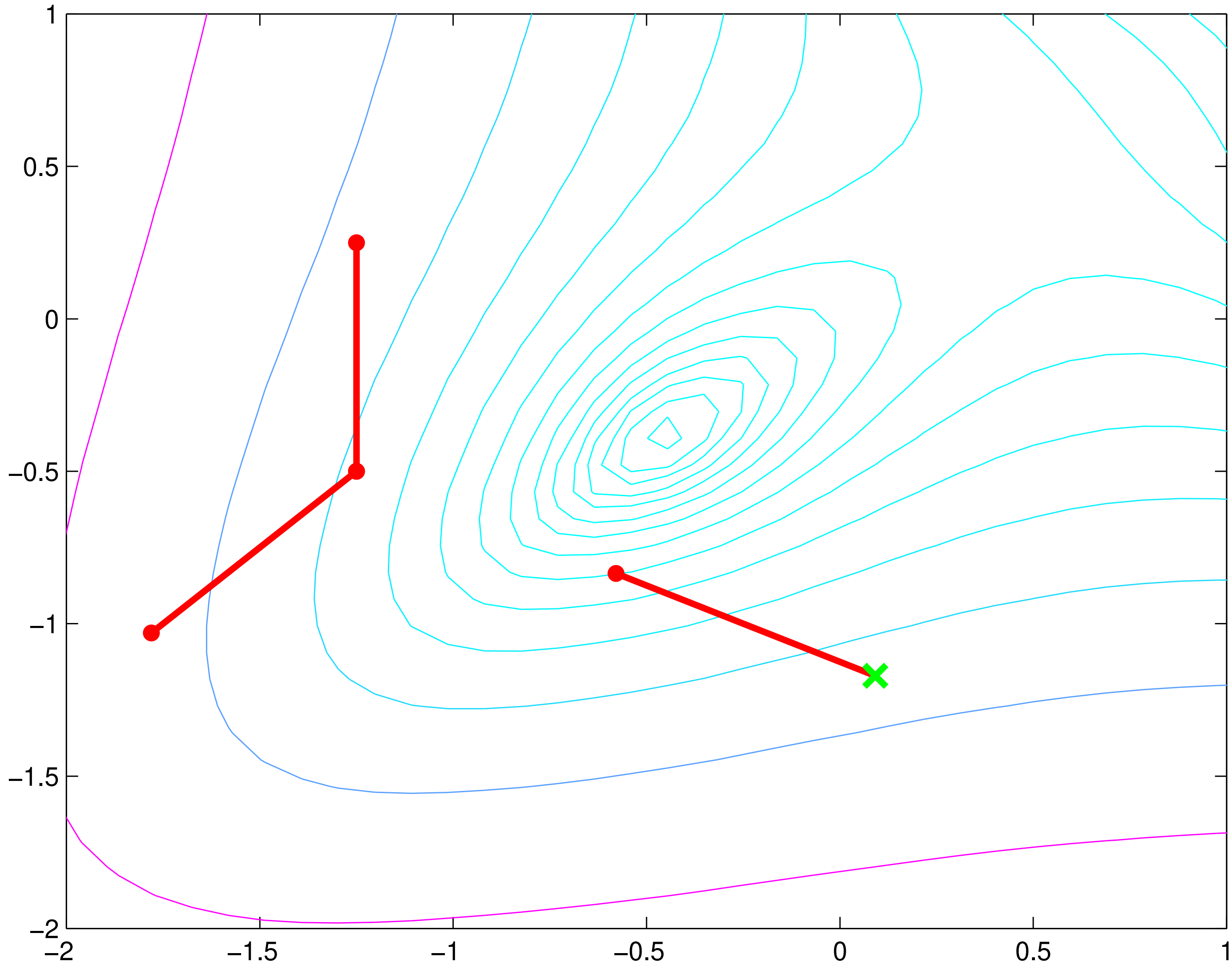
Asynchronous Parallel Pattern Search



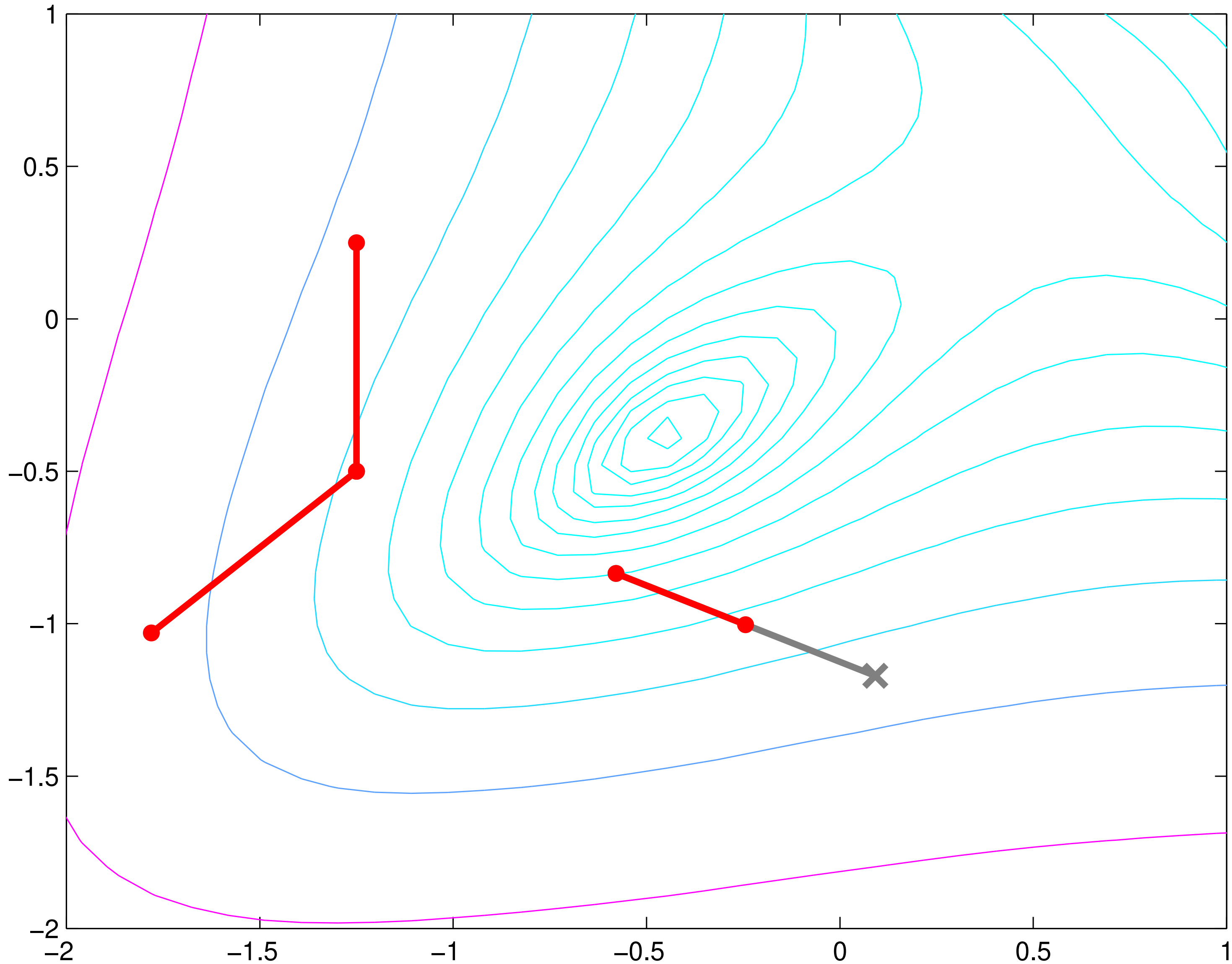
Asynchronous Parallel Pattern Search



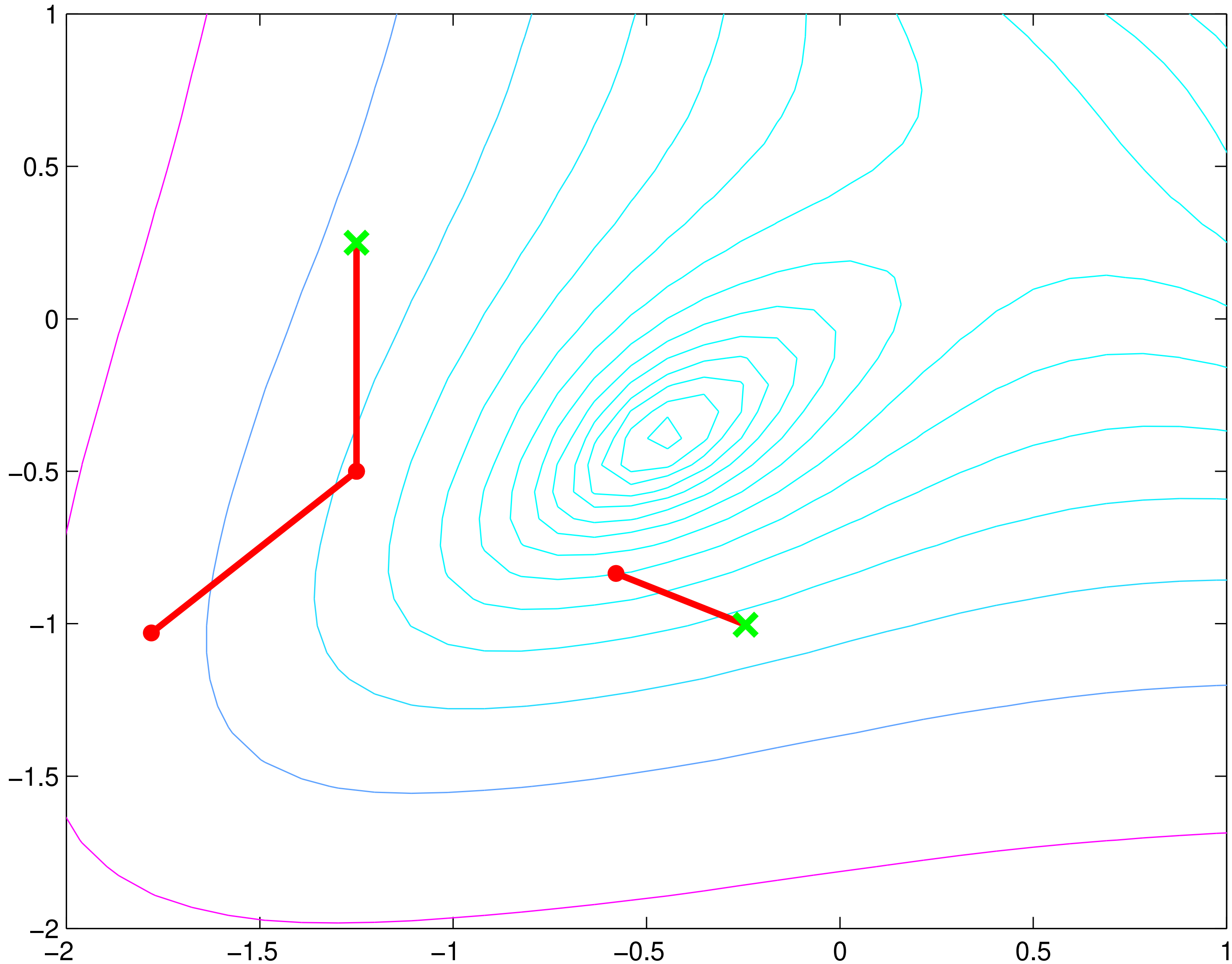
Asynchronous Parallel Pattern Search



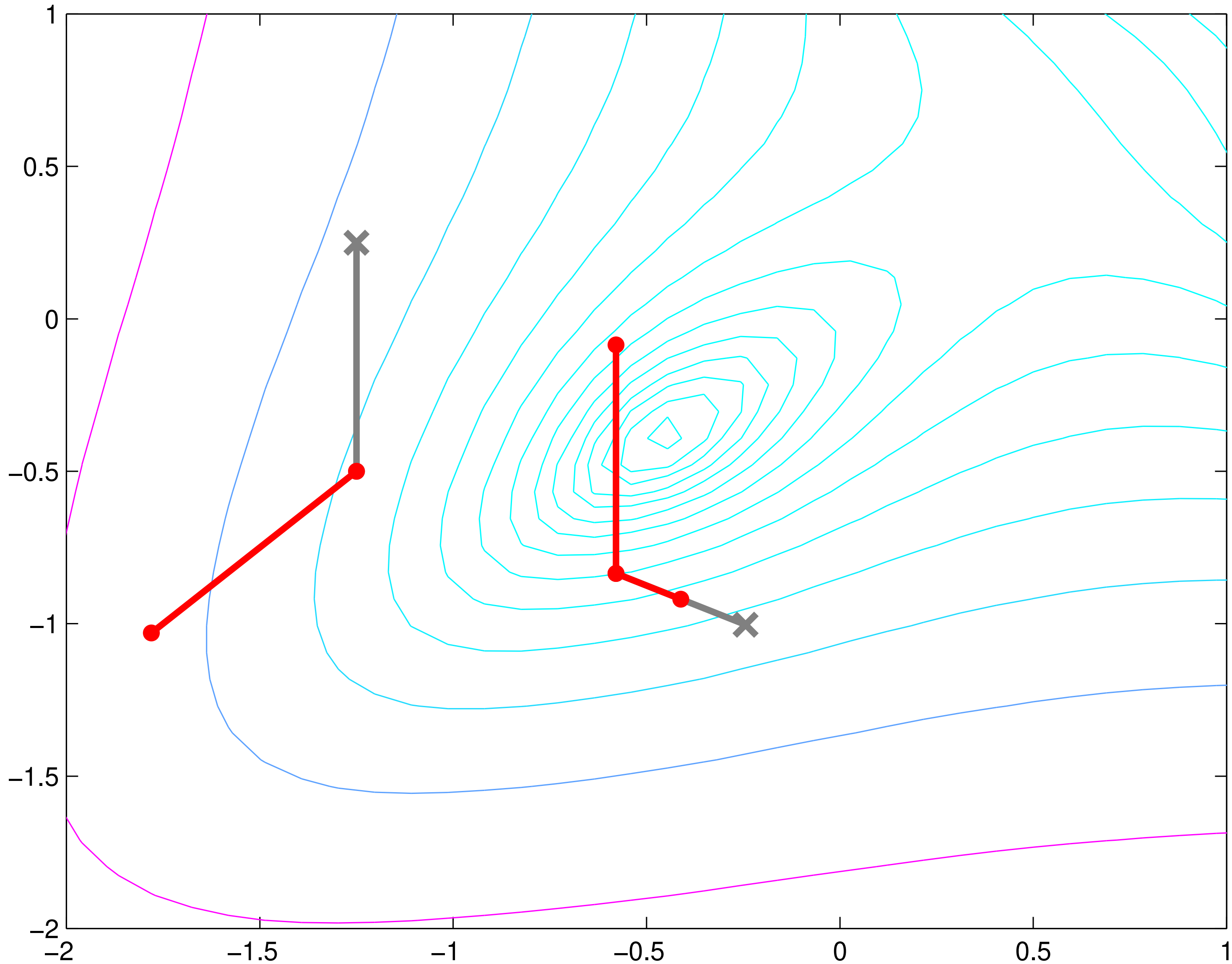
Asynchronous Parallel Pattern Search



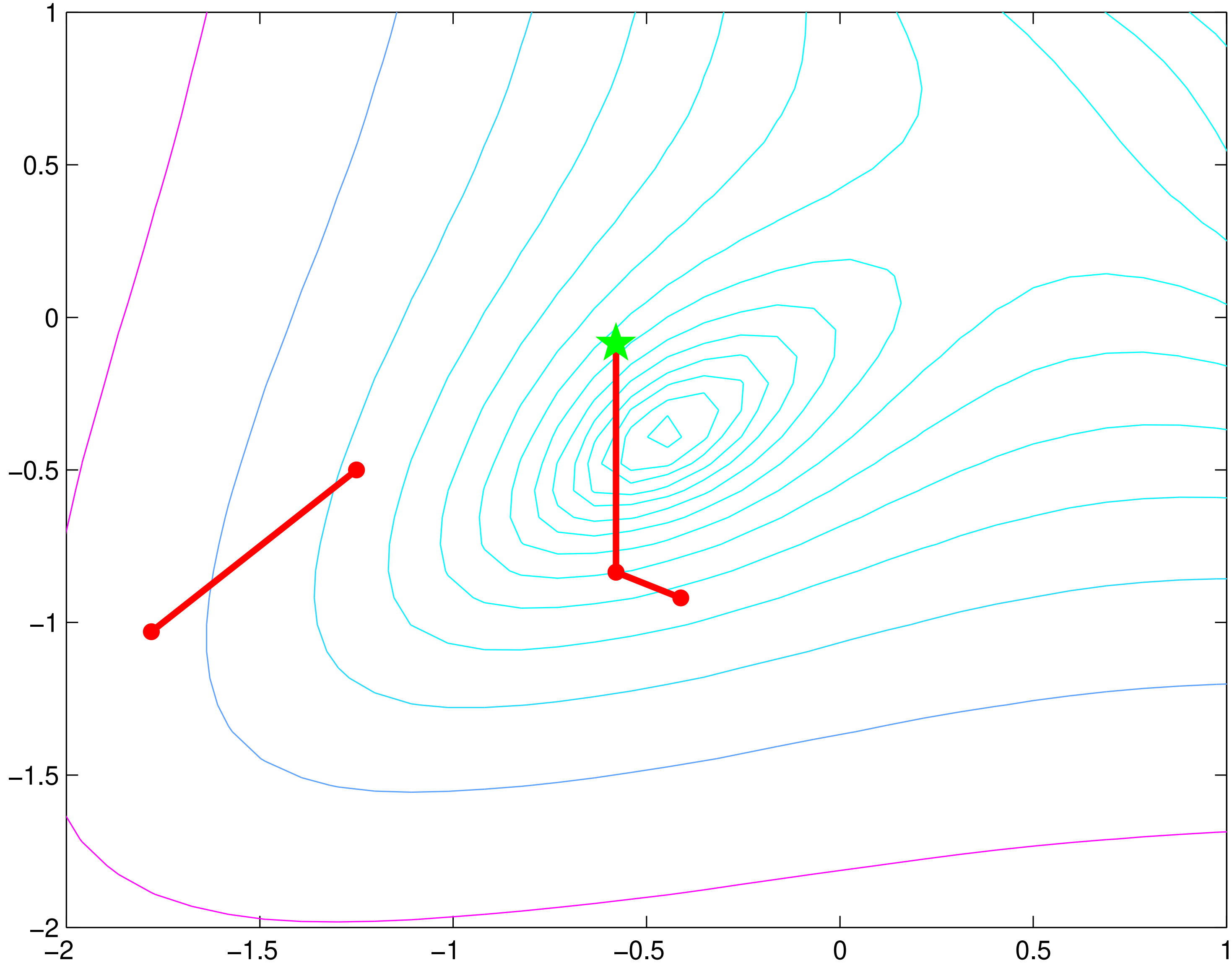
Asynchronous Parallel Pattern Search



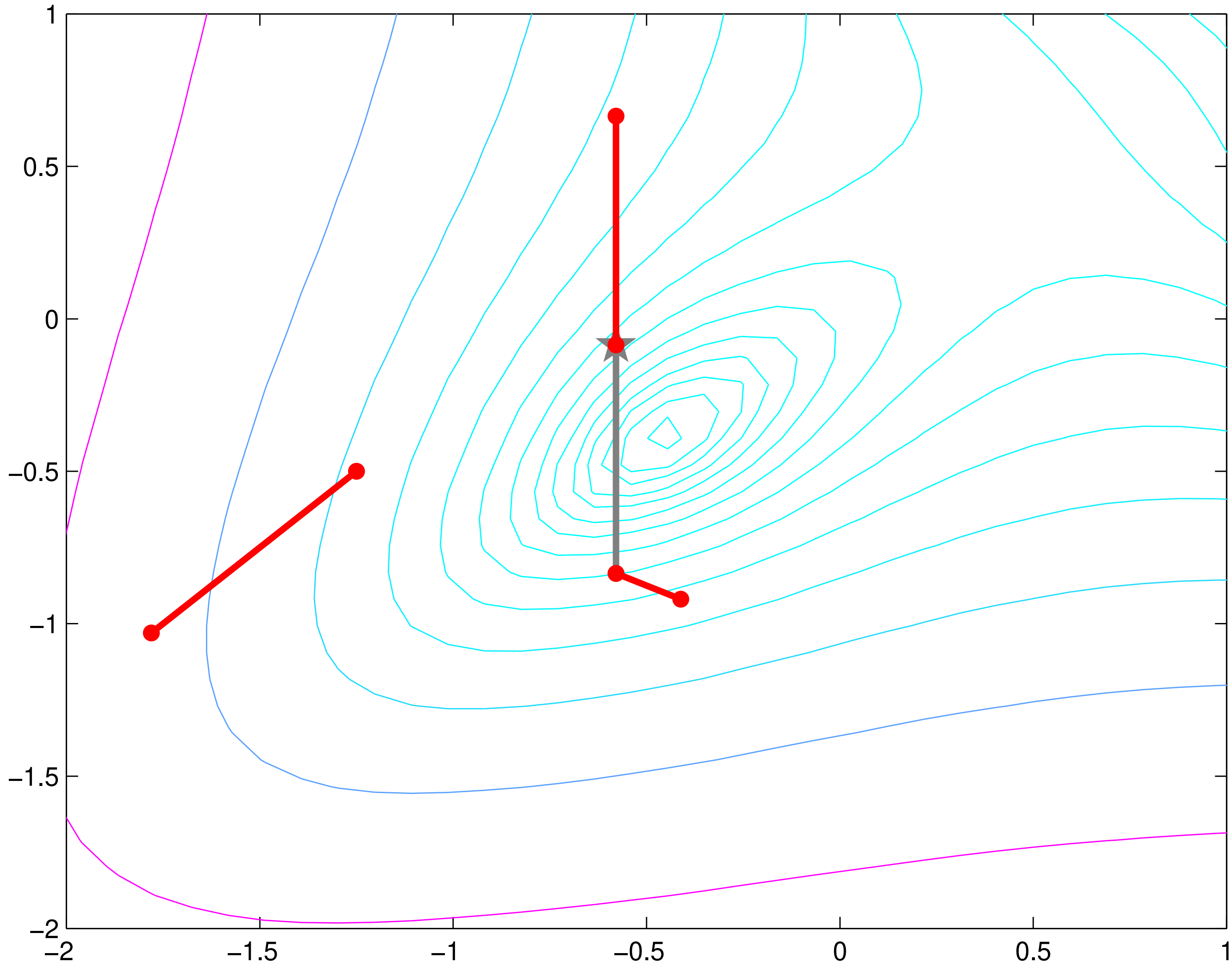
Asynchronous Parallel Pattern Search



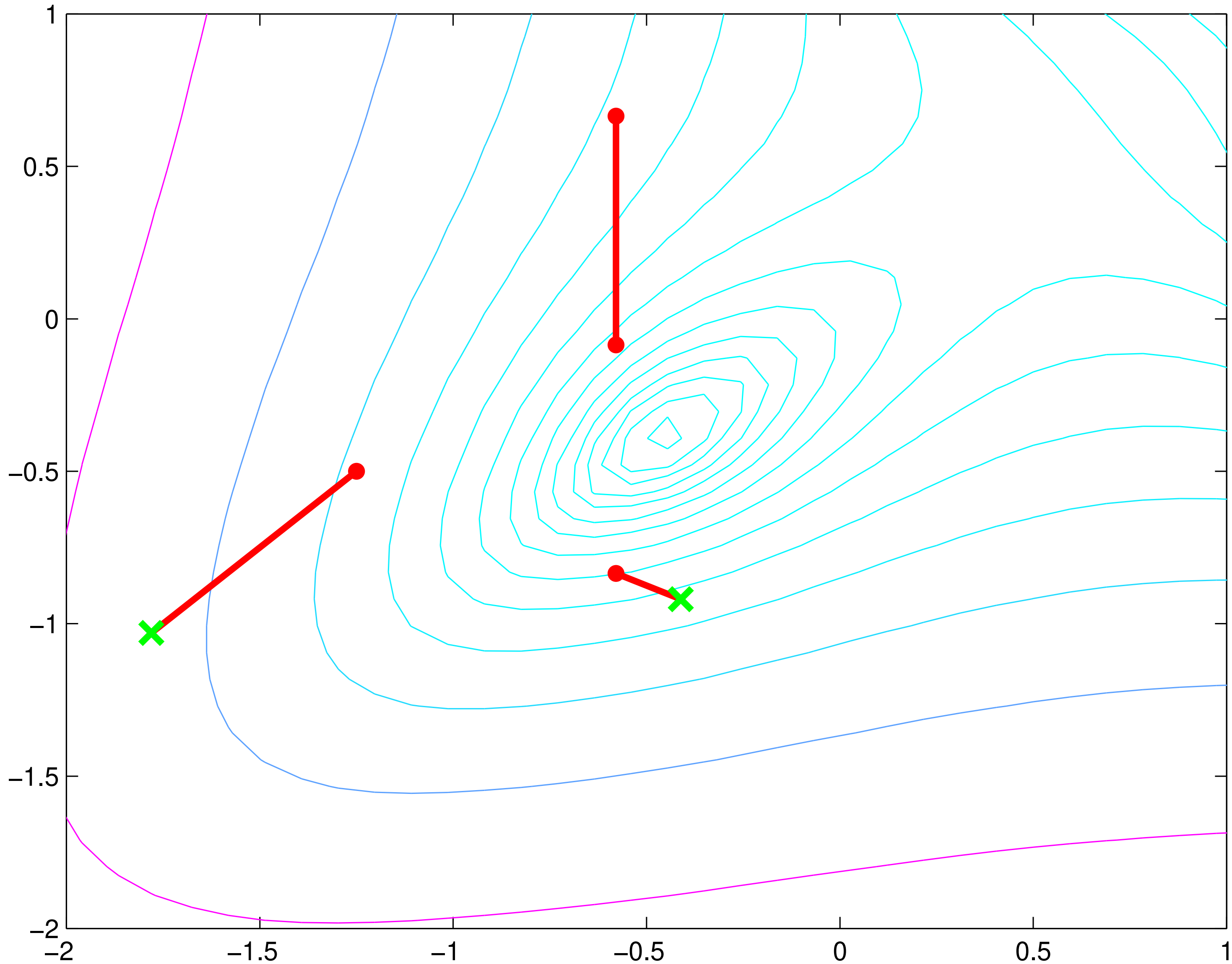
Asynchronous Parallel Pattern Search



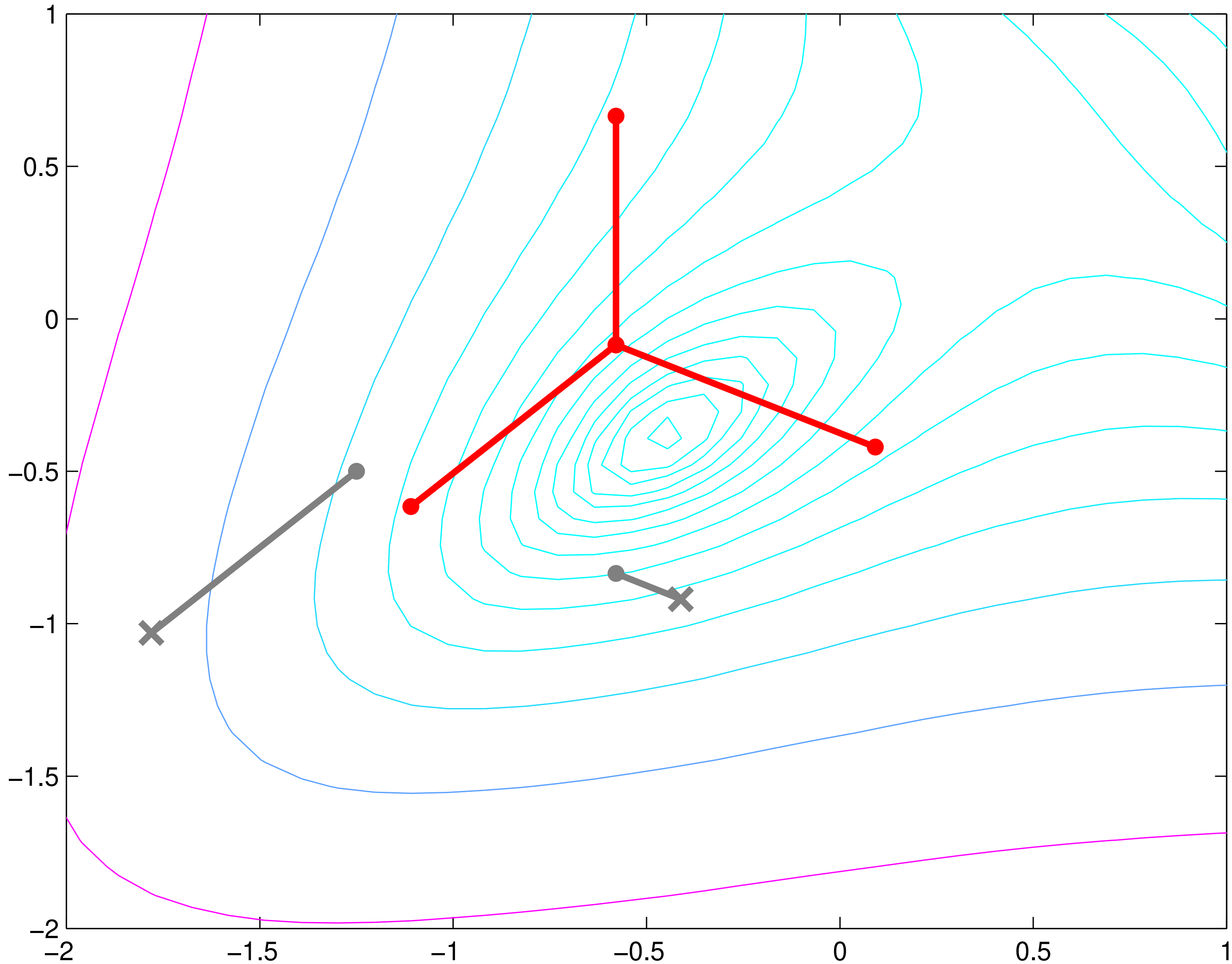
Asynchronous Parallel Pattern Search



Asynchronous Parallel Pattern Search



Asynchronous Parallel Pattern Search



APPS Dramatically Reduces Idle Time for the Circuit Simulation Problem

$$f(x) = \sum_{t=1}^{N} \left(V_t^{\mathrm{SIM}}(x) - V_t^{\mathrm{EXP}} \right)^2 \quad \text{(17 variables)}$$

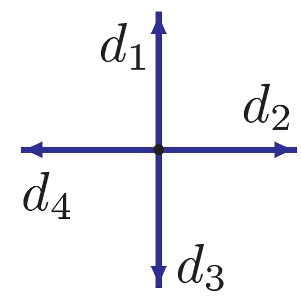
- Search directions are $\pm$ Unit Vectors (34) plus Random
- The variables are bound constrained; i.e., $l_i \leq x_i \leq u_i$

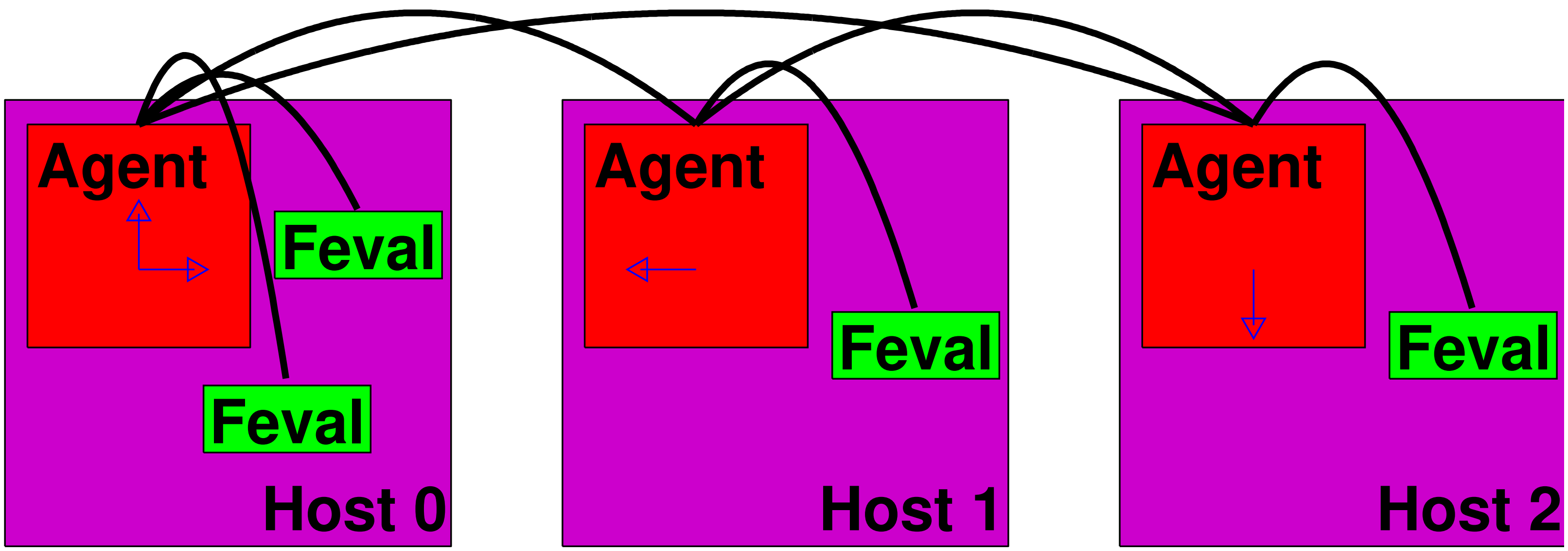
Method	Procs	f(x*)	Function Evals	Idle Time	Total Time
APPS	34	26.2	57.5	111.92	1330.55
APPS	50	26.9	50.6	63.22	807.29
PPS	34	28.8	53.0	521.48	1712.24
PPS	50	34.9	47.0	905.48	1646.53

APPS Distributes Control via Agents

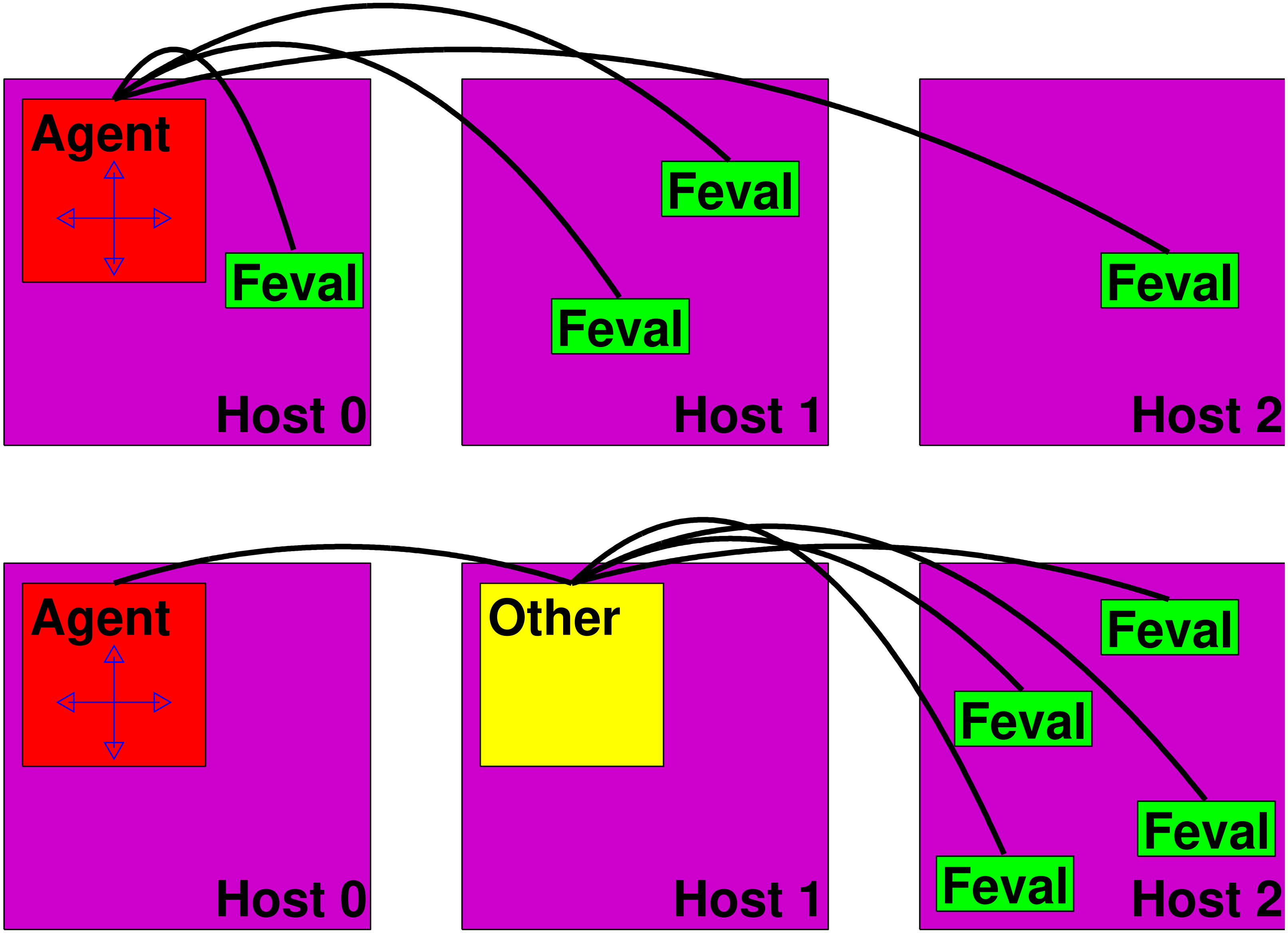
- The work of the pattern search is divided among *agents*, one per host.
- Each agent is in charge of some subset of the *search pattern*.
- Agents make algorithmic decisions based only on available information — *there is no synchronization*.

2-D Search Pattern





Single-Agent Alternatives



APPSPACK supports PVM, MPI, and Custom Communications Architectures

- PVM (single or multiple agents)
  - Fault Tolerance for Processes
  - Dynamic Addition and Deletion of Hosts
  - Process Spawning
- MPI (single agent only) [Alton Patrick]
  - Industry Standard
  - All Processes Launched at Start-Up
  - *LAM-MPI may be more like PVM*
- Other (single agent only)
  - Custom Solution

Simulation/Cost Function Interface

- Set-Up
  - Set starting point, bound constraints, scaling, etc.
- Non-Blocking Spawn
  - Given a point, return an id.
  - "Start" the function evaluation. Usually this means spawning a worker process to do the function evaluation, and communicating with the worker via PVM/MPI or file I/O.
- Non-Blocking Receive
  - Triggered by certain messages, usually sent by a worker.
  - "Finish" the function evaluation, which usually means communicate with the worker via PVM/MPI or file I/O. Also do any necessary clean-up like deleting files.
  - Return id with function value.

APPSPACK is Fault Tolerant!

Circuit Simulation Example: The “fault” versions have a failure in the simulation or agent every 30 seconds.

Method	**Initial Procs**	$\mathbf{f(x*)}$	**Total Time**
APPS	34	26.2	1330.55
APPS	34-faults	27.8	1618.46
PPS	34	28.8	1712.25
APPS	50	26.9	807.29
APPS	50-faults	54.2	1041.14
PPS	50	34.9	1646.53

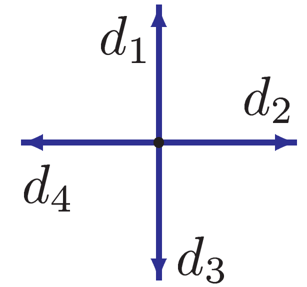
There is no version of PPS that supports faults.

Constraints Affect the Choice of Search Directions

Barrier Function:

$$\hat{f}(x) = \begin{cases} f(x) & \text{if } x \in \Omega \\ +\infty & \text{otherwise} \end{cases}$$

2-D Search Pattern



- Bound Constraints
  - Must include plus/minus unit directions in search pattern.
  - Implemented in APPSPACK 2.X
- Linear Constraints & Nonlinear Constraints *Work in progress!!* [V. Torczon & M. Lewis]
  - Search pattern must include the generators of all tangent cones in an ϵ-neighborhood of the currect iterate.
  - Current implementation simply uses a barrier function.

Using the Right Search Directions Makes a Big Difference!

Circuit Simulation Results

Type	Search Dirs	Final f(x)	Time (sec)	Idle (sec)	Func. Evals
BC	34	25.3	2750	448	36
UC	34	45.1	2440	414	39

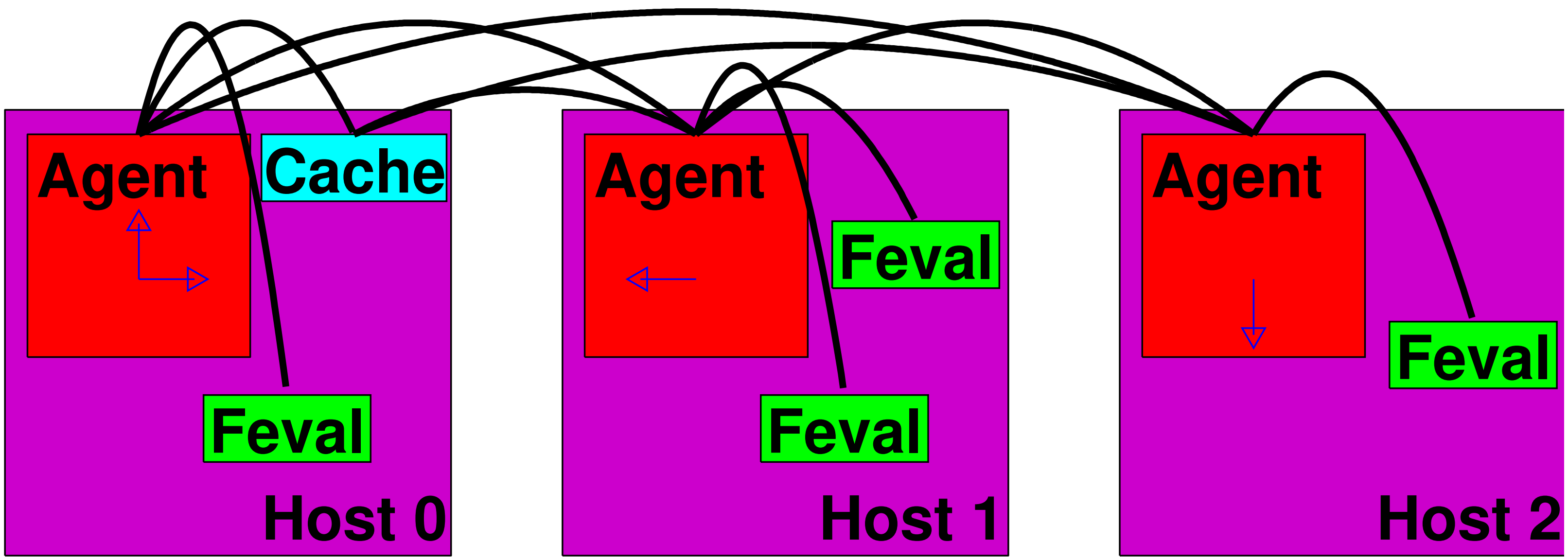
BC = Bound Constrained (i.e., right search directions)

UC = Unconstrained (i.e., wrong search directions)

Cache Avoids Repeated Cost Function Evaluations

[Alton Patrick]

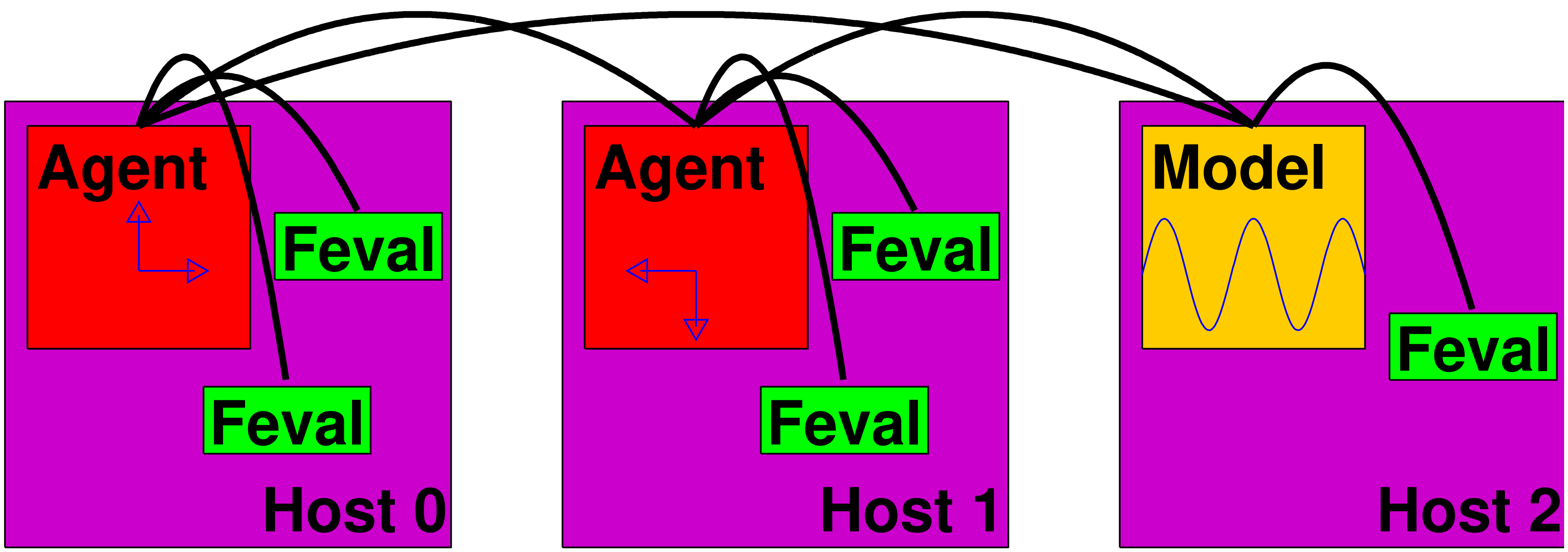
- The cache is a separate process*, and all agents communicate with the cache before and after each function evaluation.
- Cache based on splay tree (binary tree)
- Component-wise ordering of points with nearby points considered equal.



Surrogates Can Decrease Solution Time for Pattern Search

[Sarah Brown, Danny Dunlavy, T. Giunta, M. Lewis, V. Torczon]

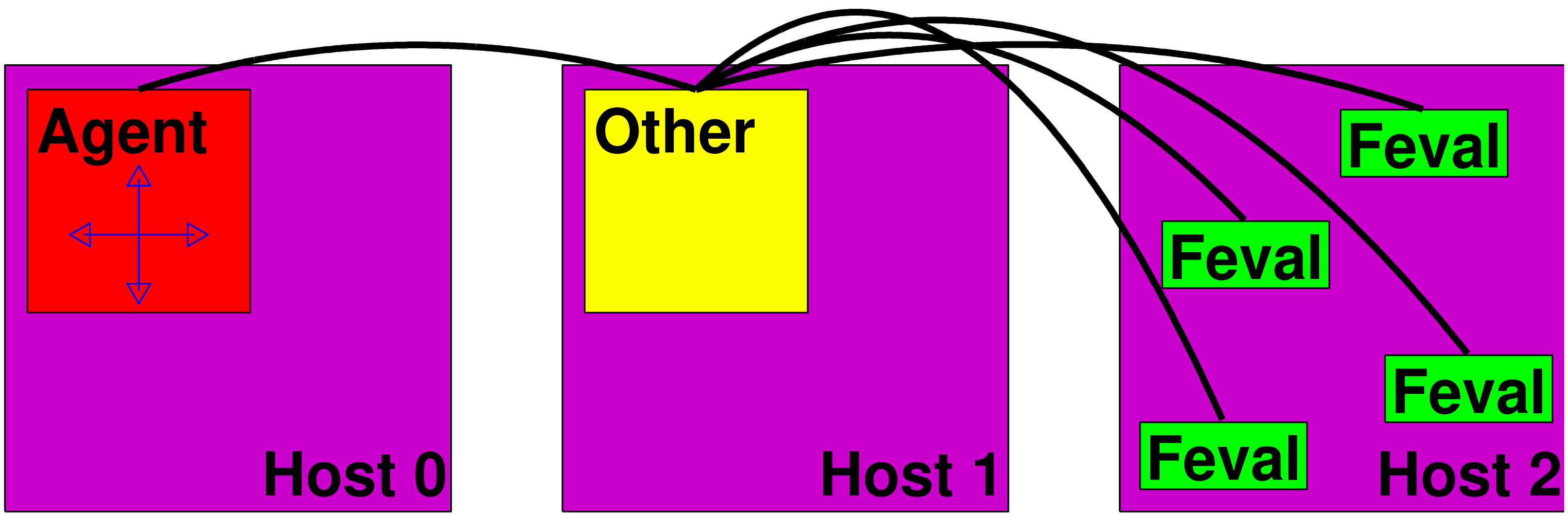
- *Work in progress!!*
- Uses function values from APPS agents to build a model.
- Predicts a minimal point.
- Evaluates predicted point.
- If predicted point is good, returns it to APPS agents.



DAKOTA/SGOPT/APPSPACK

[Boyd Schimel, Bill Hart, Mike Eldred]

- *Work in progress!!*
- APPSPACK replaces pattern search in SGOPT.
- SGOPT uses DAKOTA for the function evaluations.
- DAKOTA has the new capability to asynchronously start and stop function evaluations.



Contact Me To Get Started With APPSPACK

Tammy Kolda

tgkolda@sandia.gov

294-4769

For More Information...

http://csmr.ca.sandia.gov/projects/apps.html

http://csmr.ca.sandia.gov/~tgkolda/