Asynchronous Parallel Pattern Search with Bound Constraints

Tamara G. Kolda
Sandia Nationals Labs

This research was sponsored by the U.S. Department of Energy.

Outline

- Motivation
- Positive Spanning Sets
- Asynchronous Parallel Pattern Search (APPS)
- Bound Constraints for APPS
- Example: An Electrical Circuit Simulation
- Conclusions & Future Work
- APPSPACK Software

Motivation

Our goal is to solve the bound constrained optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & l \leq x \leq u \end{array}$$

Characterized by...

- An expensive function evaluation,
- A gradient that cannot (for all practical purposes) be calculated, and
- Function values that are too noisy to yield reliable gradient approximations.

This characterization is true of many optimization engineering problems, especially when the code is “black box”.

Choosing the Search Directions for Unconstrained APPS

The pattern must be chosen so that it positively spans $\Re^n$.

Defn: A set of vectors $\{d_1, \ldots, d_p\}$ *positively spans* $\Re^n$ if any vector $x \in \Re^n$ can be written as

$$x = \alpha_1 d_1 + \cdots + \alpha_p d_p, \quad \alpha_i \geq 0 \;\; \forall\, i.$$

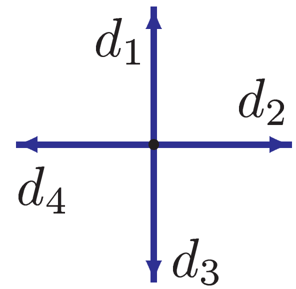
That is, any vector can be written as a *nonnegative* linear combination of the basis vectors.

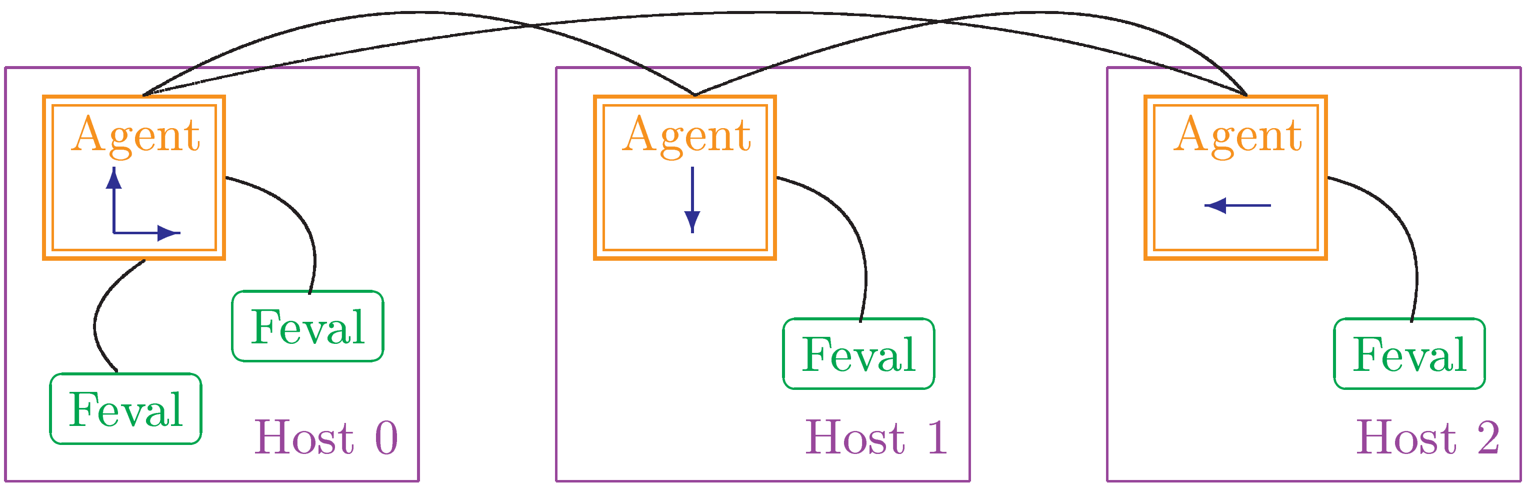
Fact: If $\{d_1, \ldots, d_p\}$ positively spans $\Re^n$, then for any $x \neq 0$, there exists d_i such that $d_i^T x > 0$.

APPS Implementation

- 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



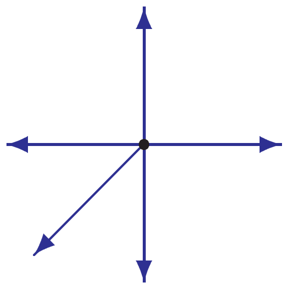


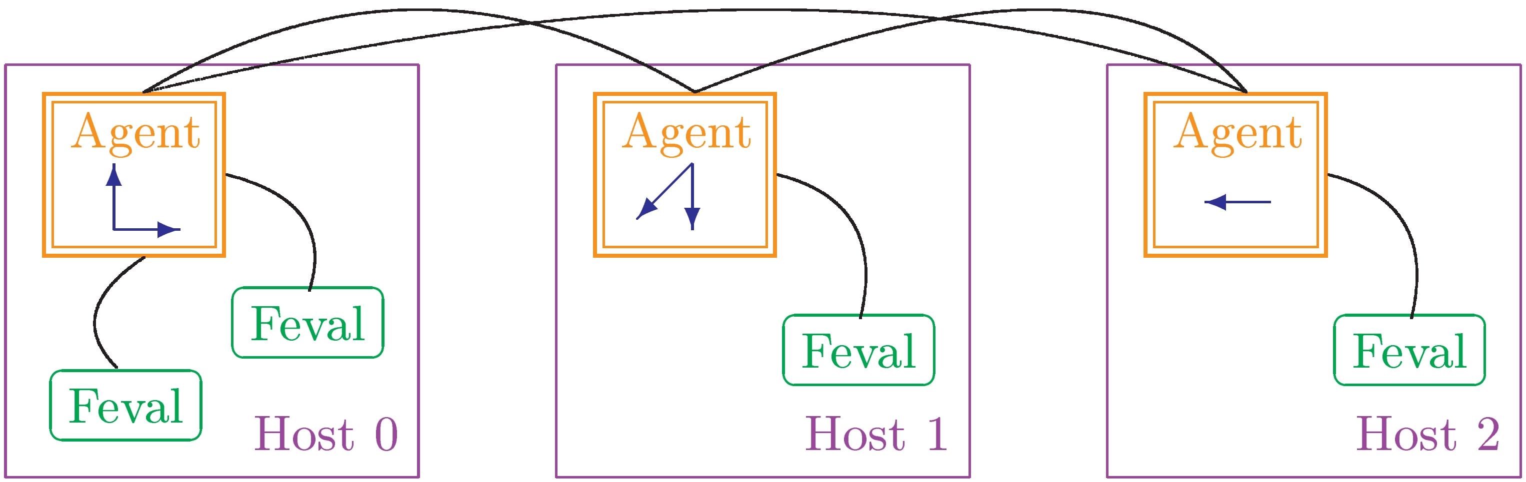
APPS - Extra Directions

There can be more than the minimal number of search directions.

Here, note that we can construct there are at least two distinct positive spanning sets.

2-D Search Pattern



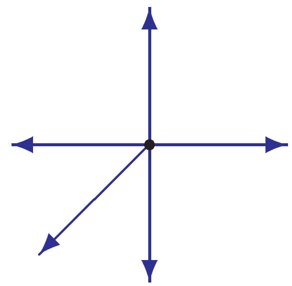


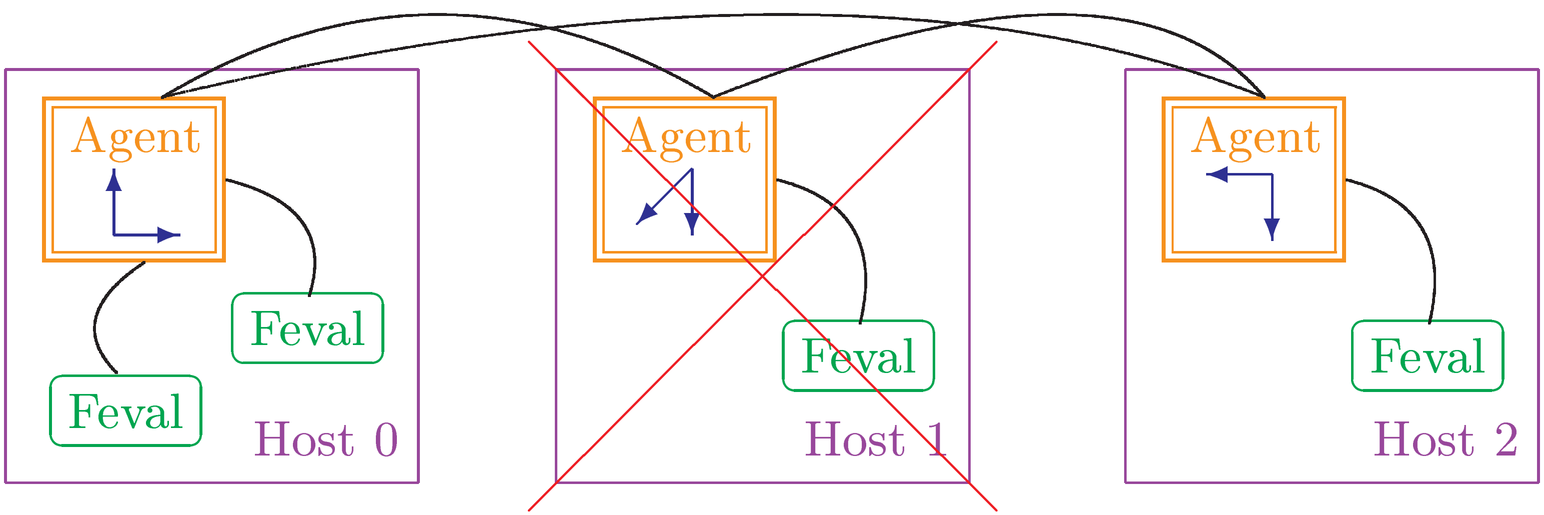
APPS & Fault Tolerance

If there is a fault in a node, then the search directions in the *base pattern* are reassigned.

When Host 1 dies, the (0,-1) direction is reassigned to Host 2, but the (-1,-1) direction is not reassigned at all.

2-D Search Pattern





APPS Agent Algorithm

1. **Q:** Is there a **new best point** reported by another agent?

   Y: Store $\{x_{\text{new}}, f_{\text{new}}, \Delta_{\text{new}}, \Pi_{\text{new}}\}$.

   x_{new} is the point
   f_{new} is the function value
   Δ_{new} is the step length
   Π_{new} is the convergence table

   Q: Is $f_{\text{new}} < f_{\text{best}}$ (using tie-breaking rules)?
   Y: Update best point. Go to Step 1.
   N: Go to Step 1.

   N: Go to Step 2.

APPS Agent Algorithm

2. **Q:** Is there a **completed local function evaluation**?

   Y: Store $\{x_{\text{trial}}, f_{\text{trial}}, \Delta_{\text{trial}}, \Pi_{\text{trial}}\}$ and let i indicate the search direction.

   - **Q:** Is $f_{\text{trial}} < f_{\text{best}}$ (using tie-breaking rules)?
   - **Y:** Update best point and broadcast *new best point* message.
   - **N:** If the best point has not been updated since the trial point was generated, reduce Δ_{trial}; else, $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$.
   - **Q:** Is $\Delta_{\text{trial}} < tol$?
   - **Y:** Update Π_{best}, broadcast a *converged single direction* message for direction i, and check for convergence using positive basis test. Go to Step 2.
   - **N:** Generate a new trial point and begin a new function evaluation. Go to Step 2.

   N: Go to Step 3.

APPS Agent Algorithm

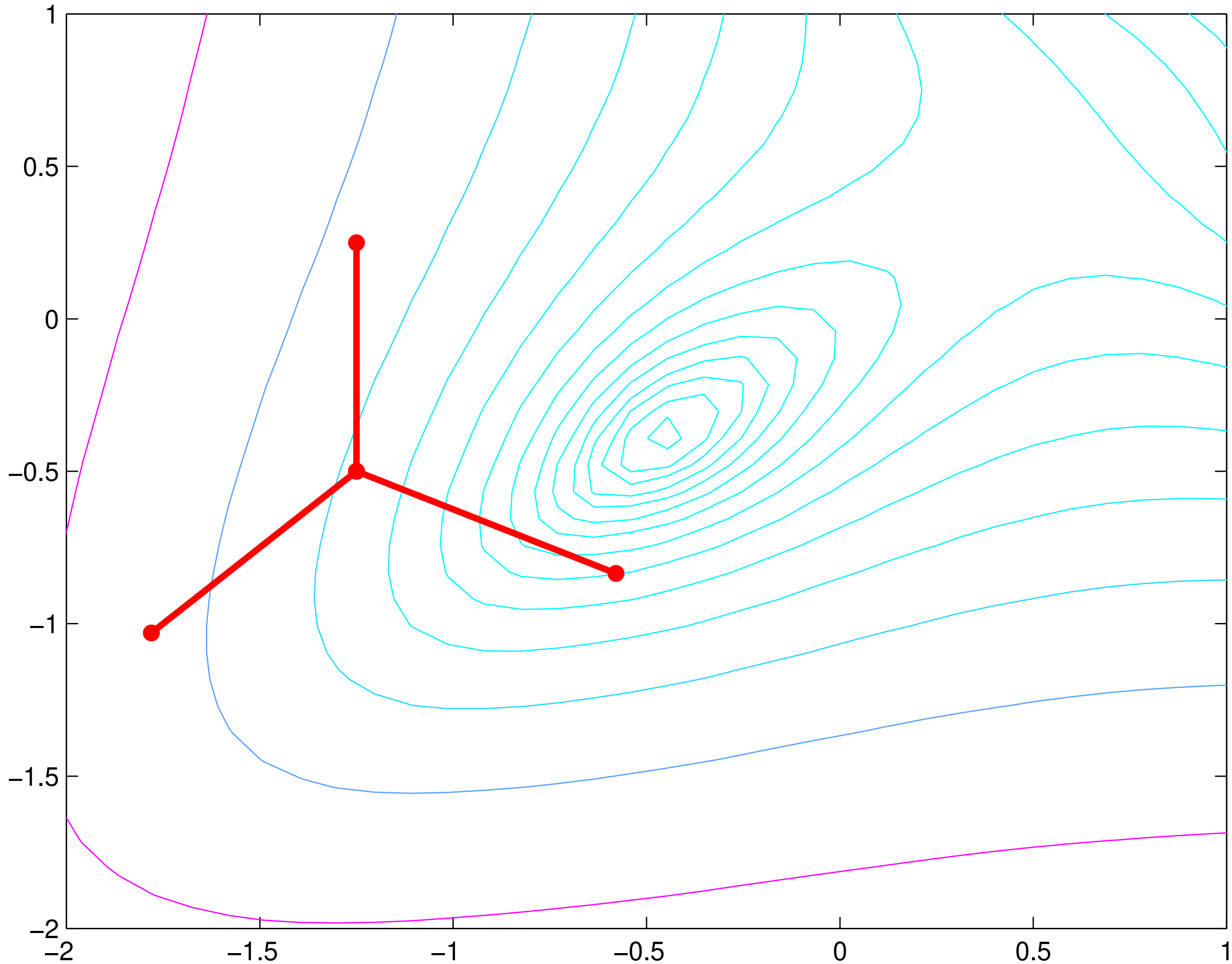
3. Is a **converged search direction** reported by another agent?

   Y: Store $\{x_{\text{new}}, f_{\text{new}}, \Delta_{\text{new}}, \Pi_{\text{new}}\}$.

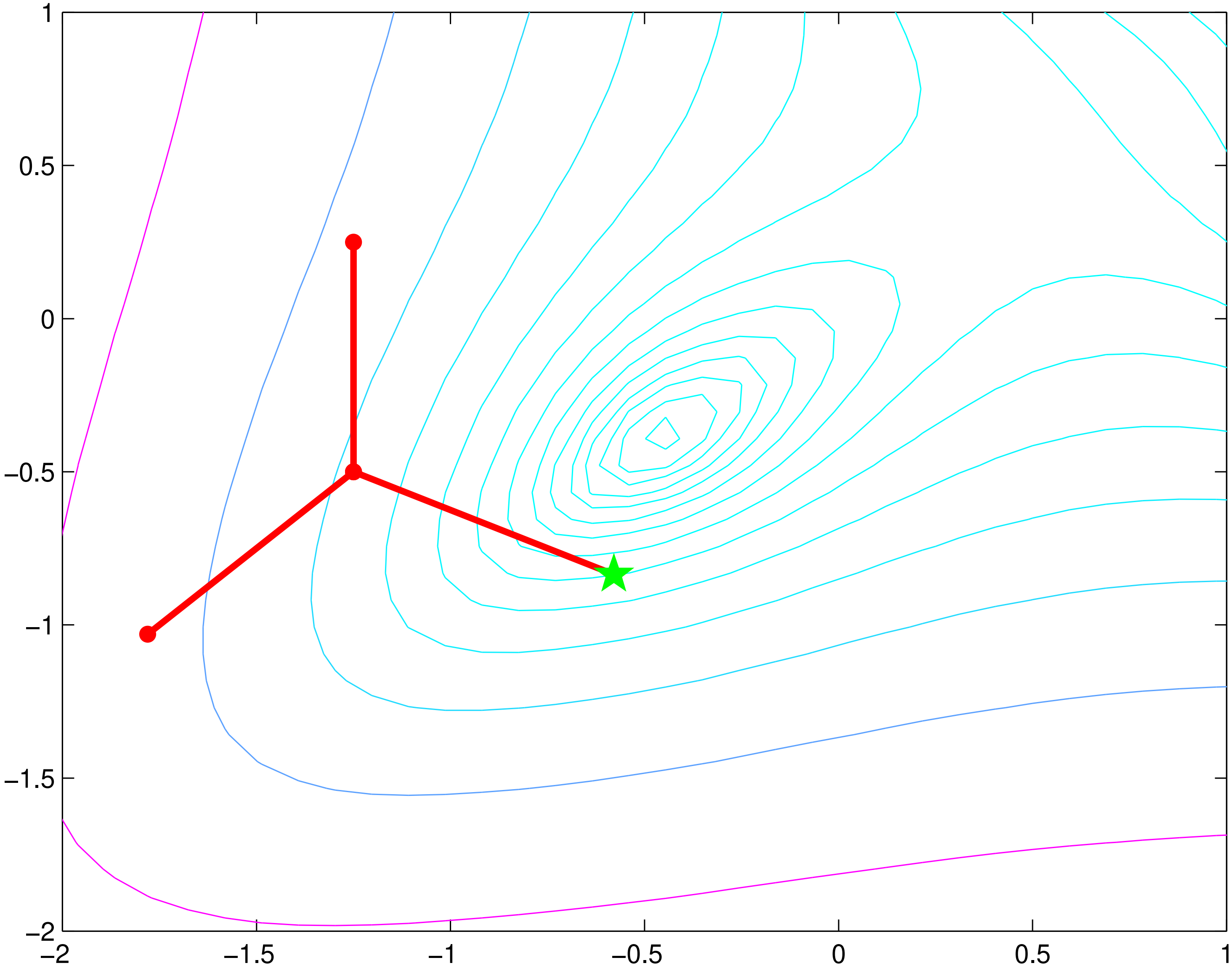
   - **Q:** Is $f_{\text{new}} < f_{\text{best}}$ (using tie-breaking rules)?
   - **Y:** Update best point, and check for convergence using positive basis test. Go to Step 3.
   - **N:** **Q:** Is $x_{\text{new}} = x_{\text{best}}$?
     - **Y:** Merge Π_{new} and Π_{best}, and check for convergence using positive basis test. Go to Step 3.
     - **N:** Go to Step 3.

   N: Go to Step 1.

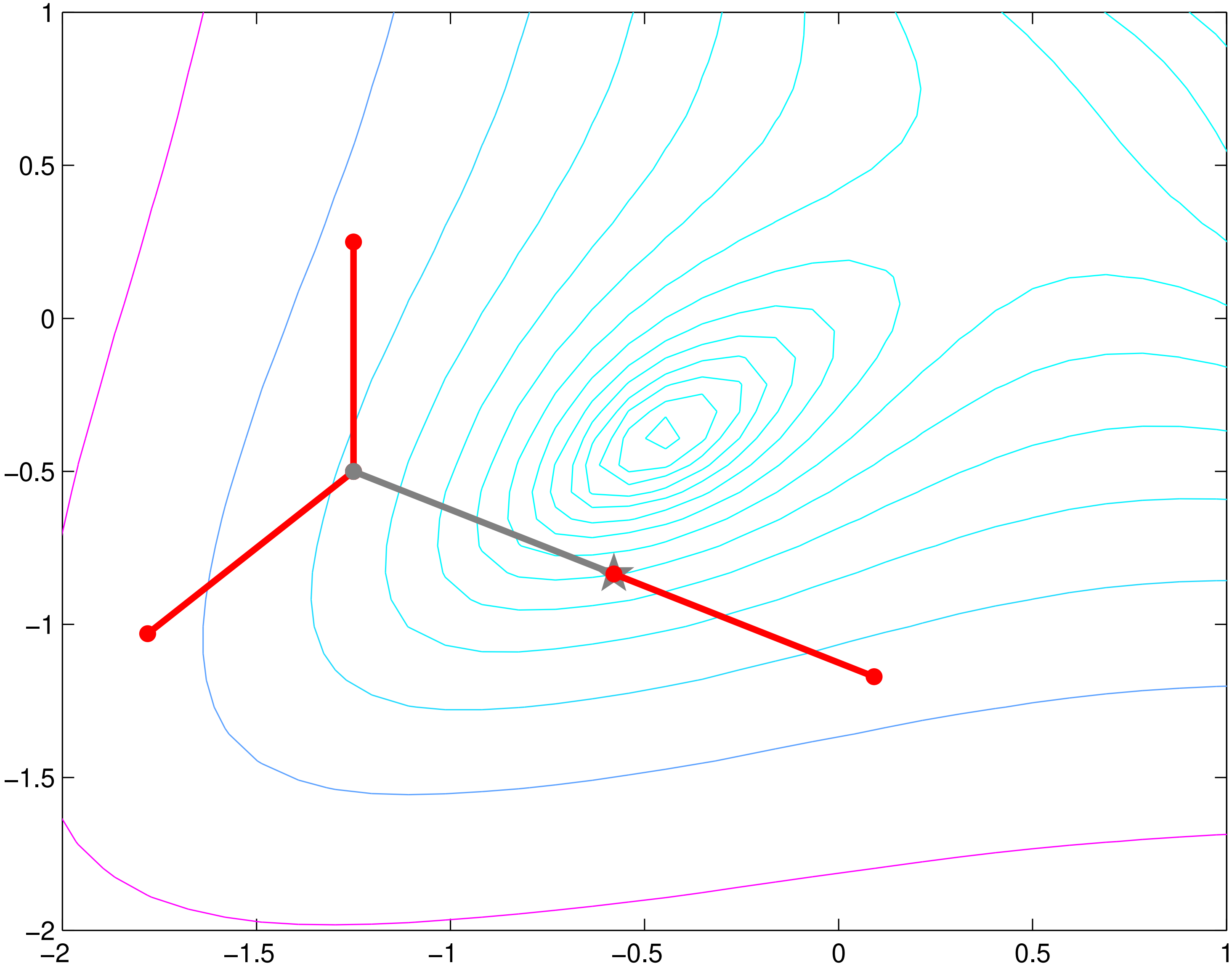
Asynchronous Parallel Pattern Search



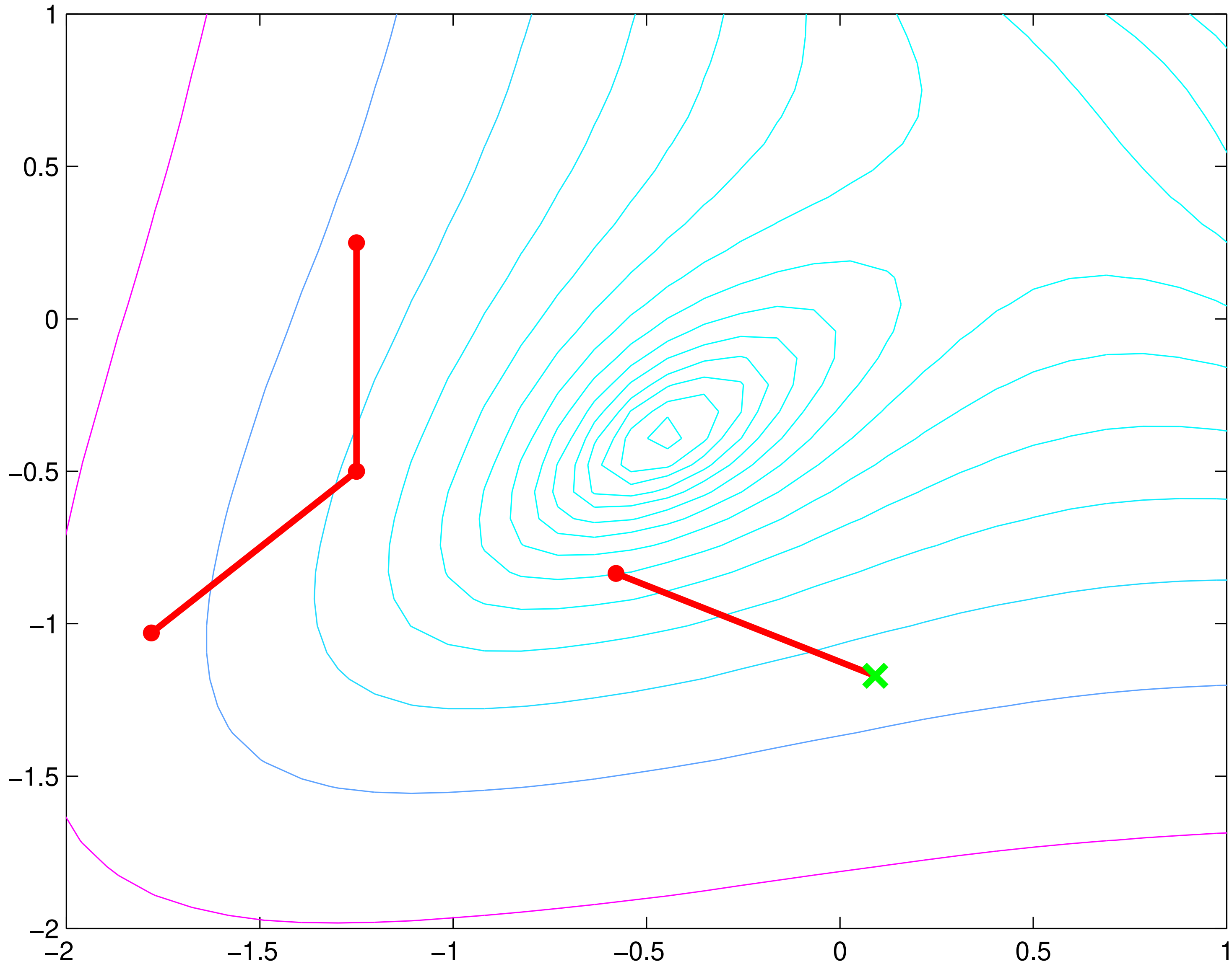
Asynchronous Parallel Pattern Search



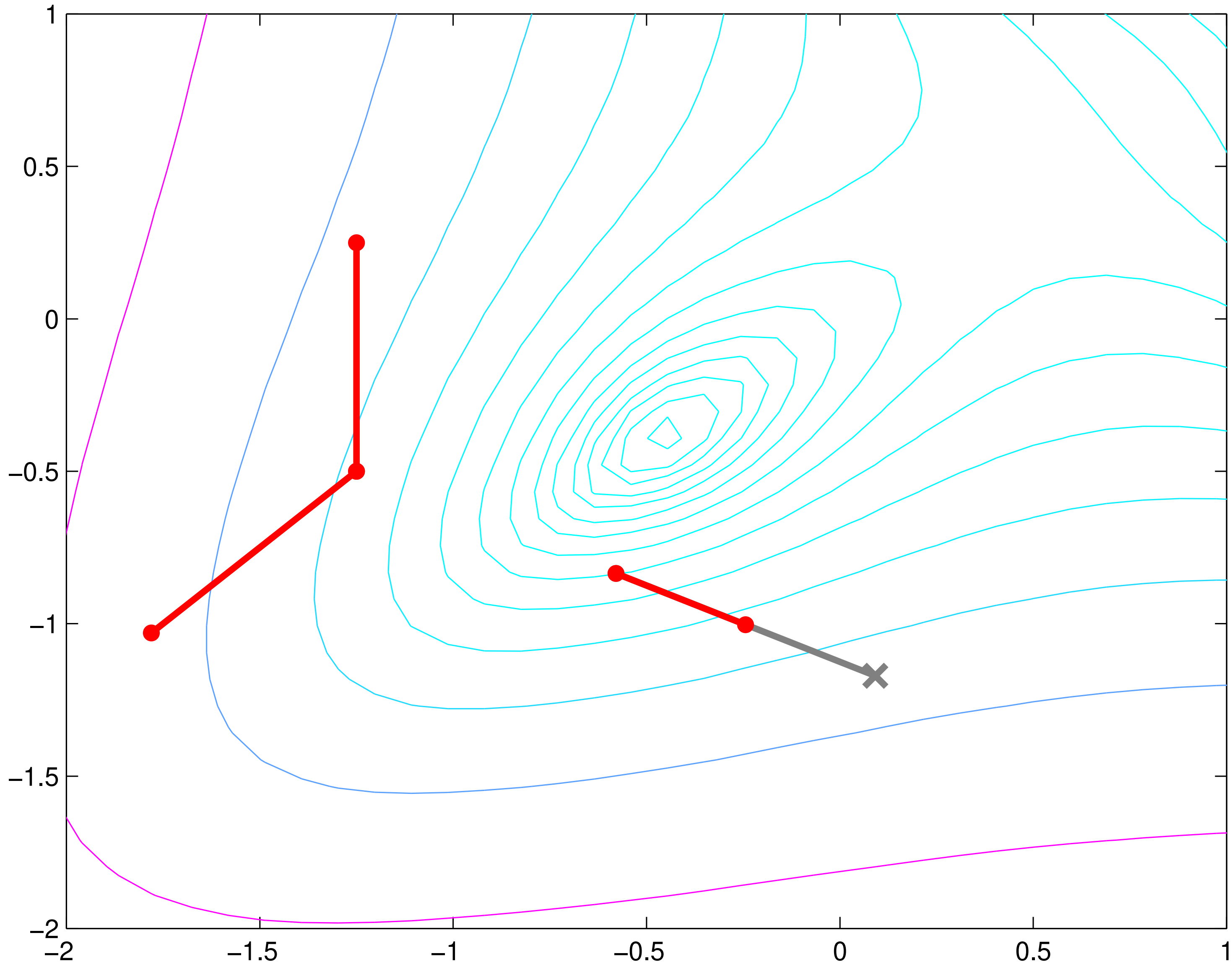
Asynchronous Parallel Pattern Search



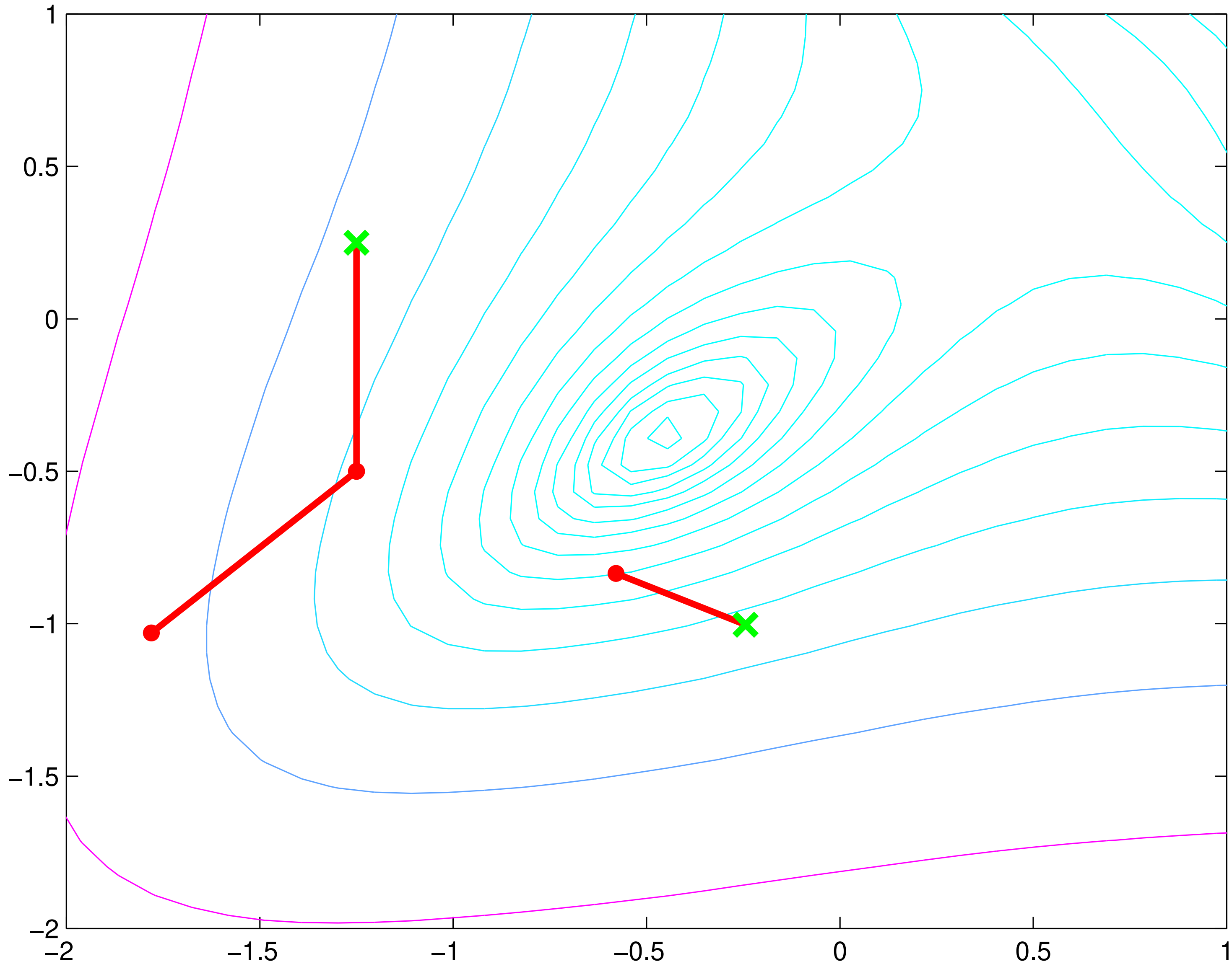
Asynchronous Parallel Pattern Search



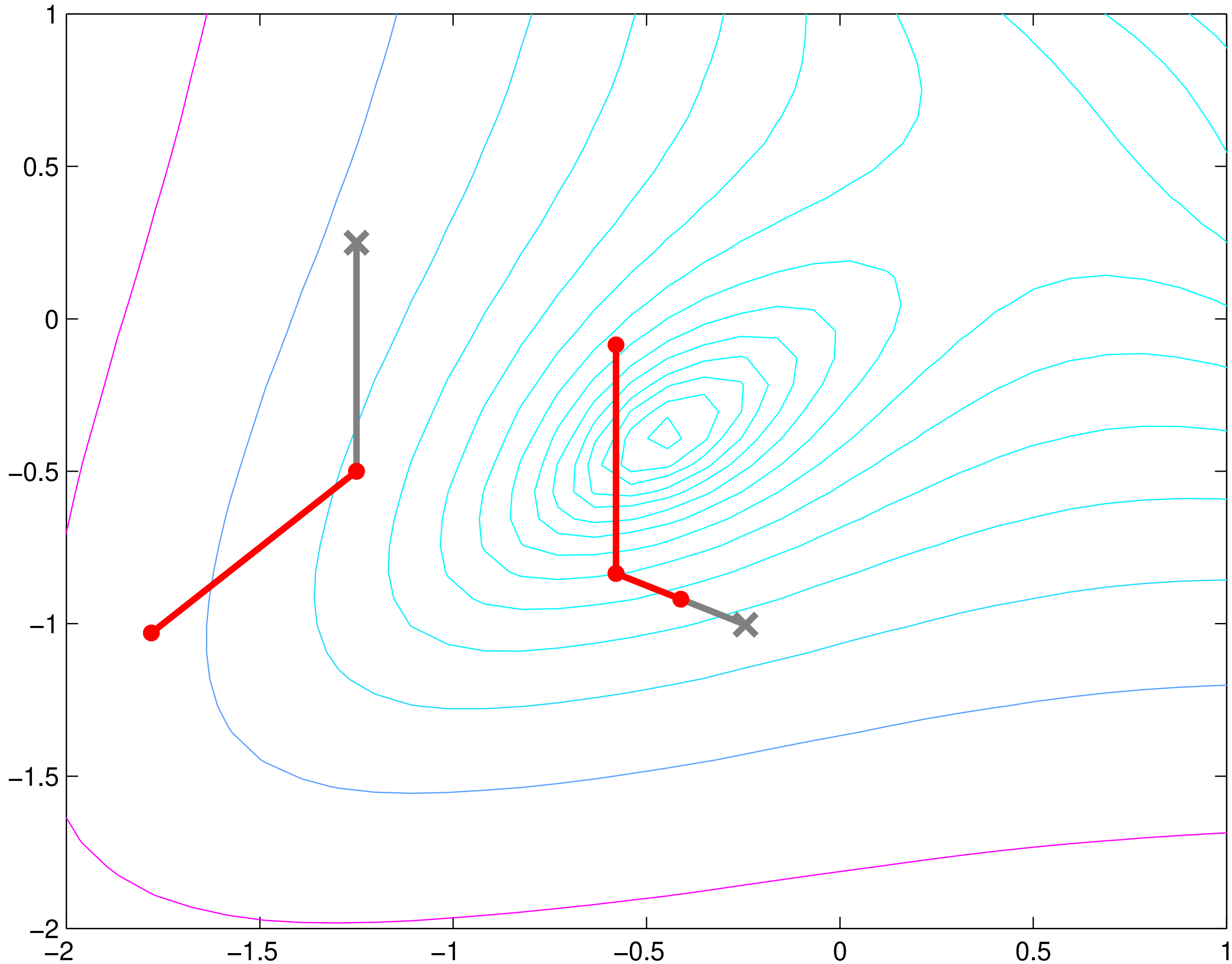
Asynchronous Parallel Pattern Search



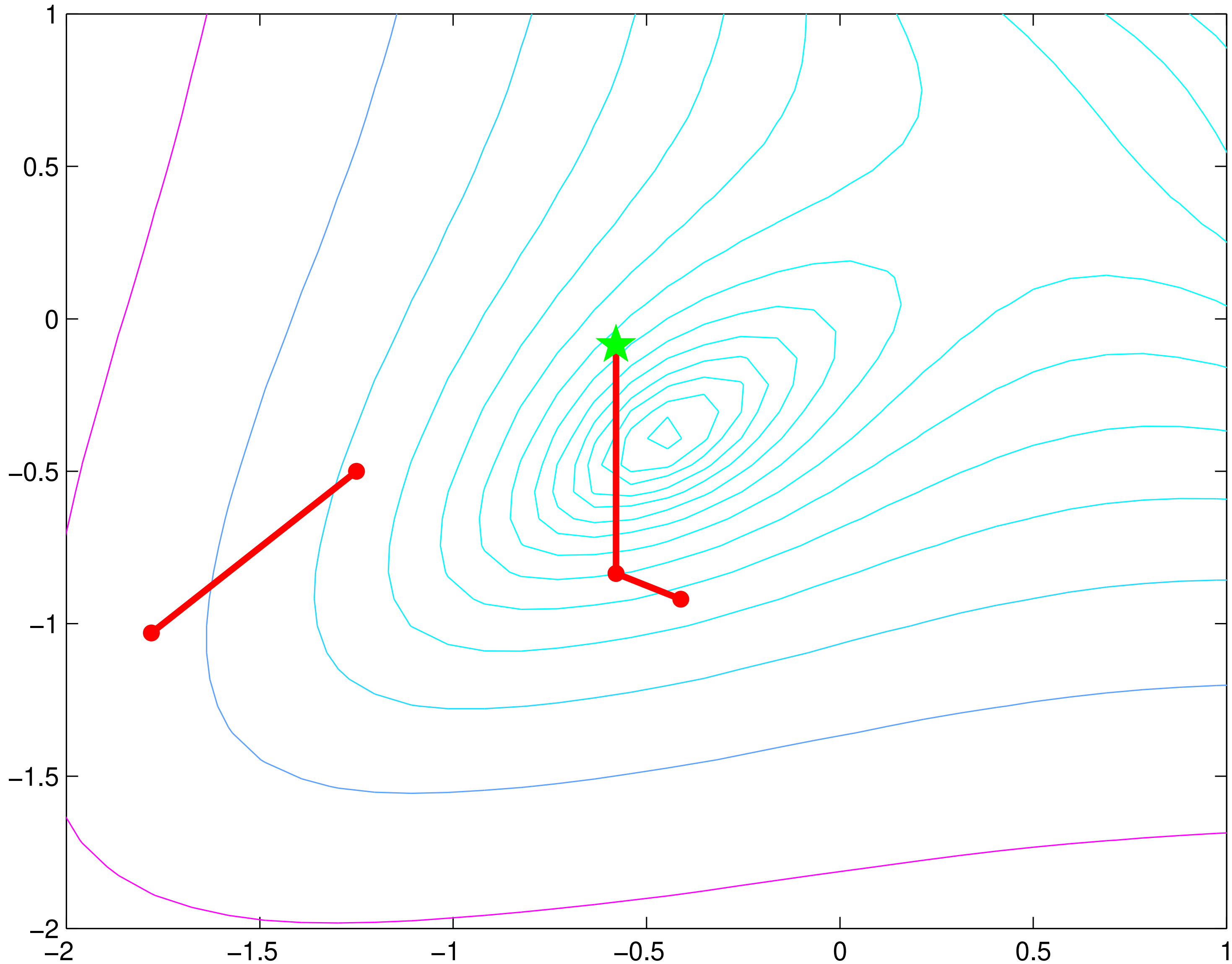
Asynchronous Parallel Pattern Search



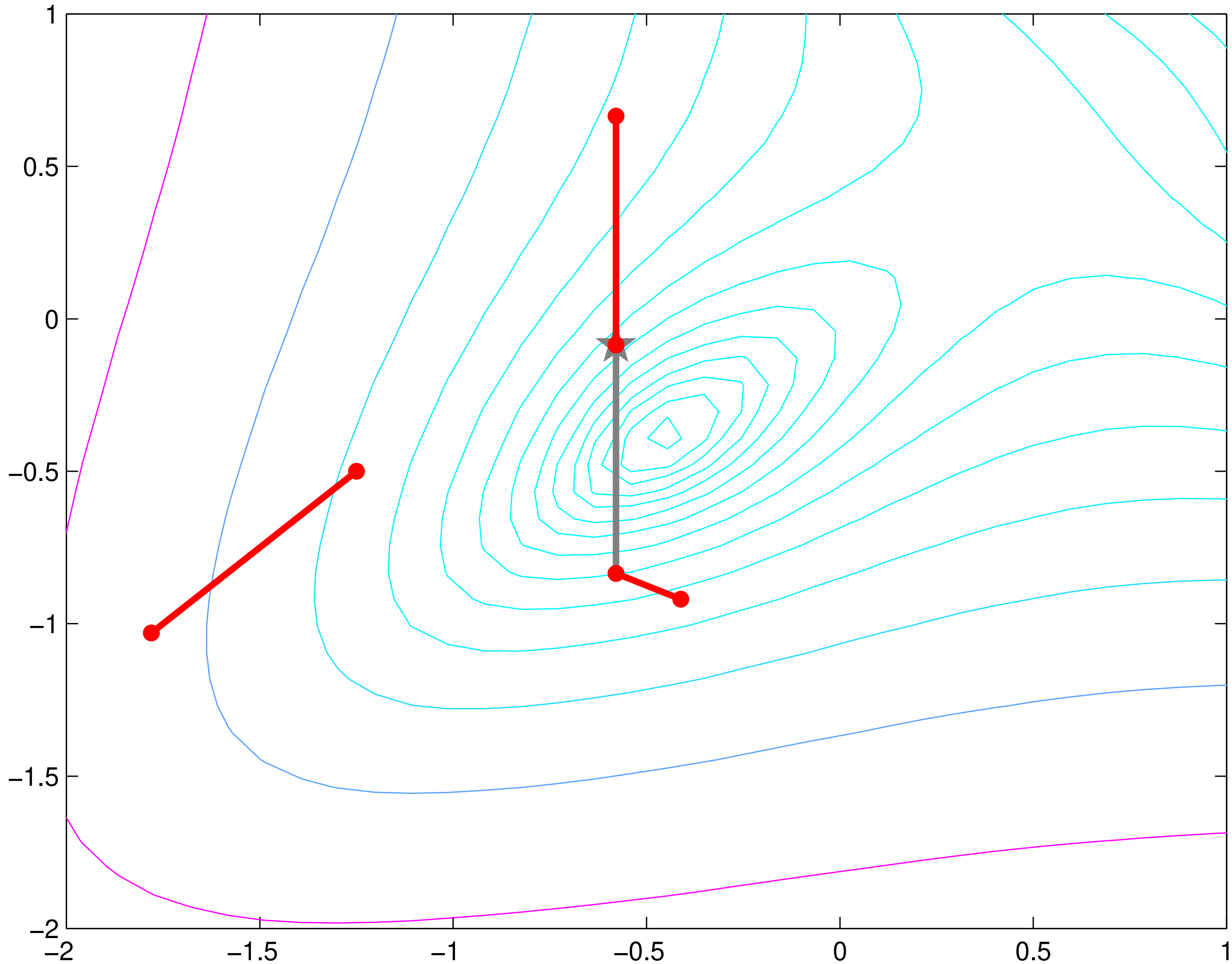
Asynchronous Parallel Pattern Search



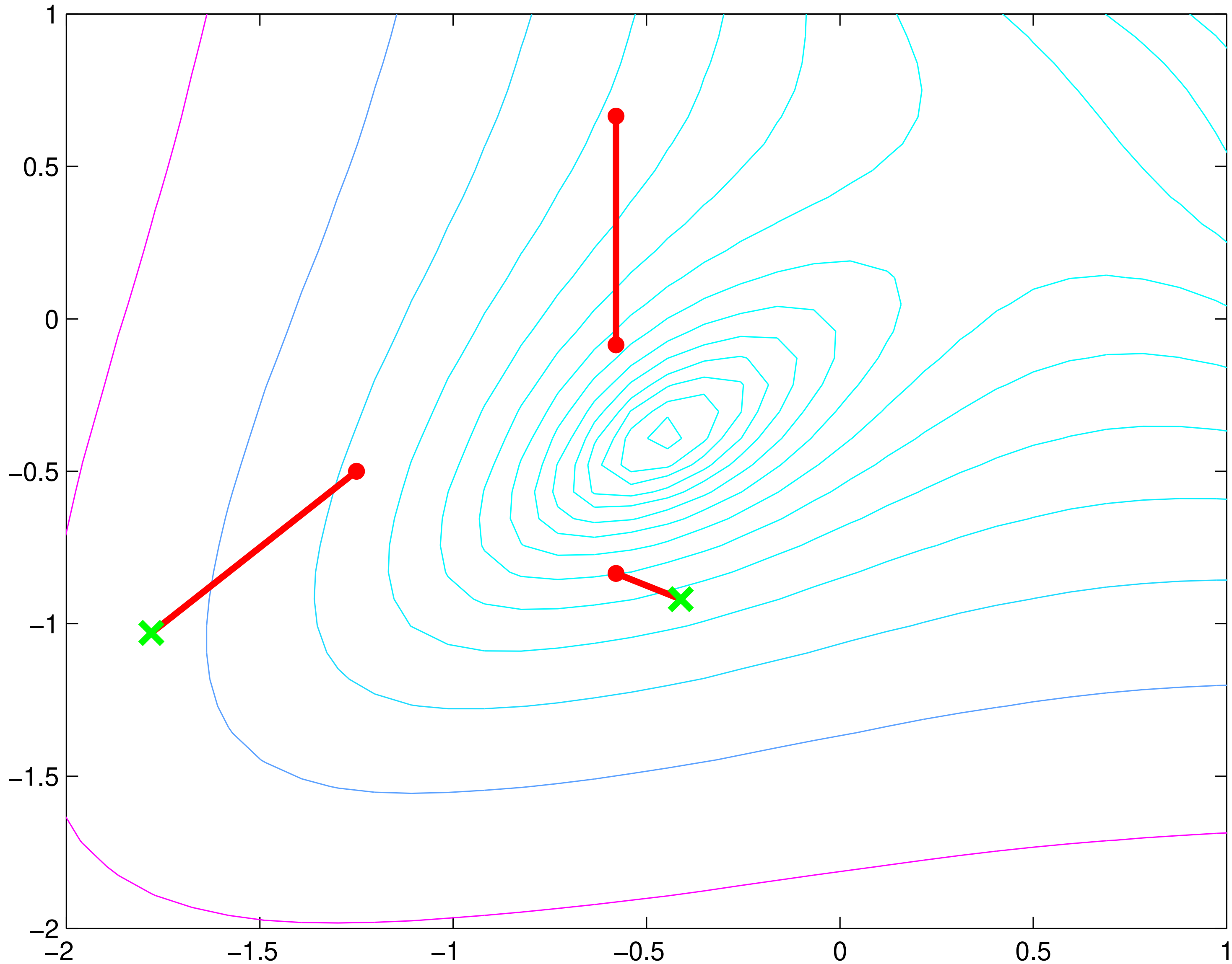
Asynchronous Parallel Pattern Search



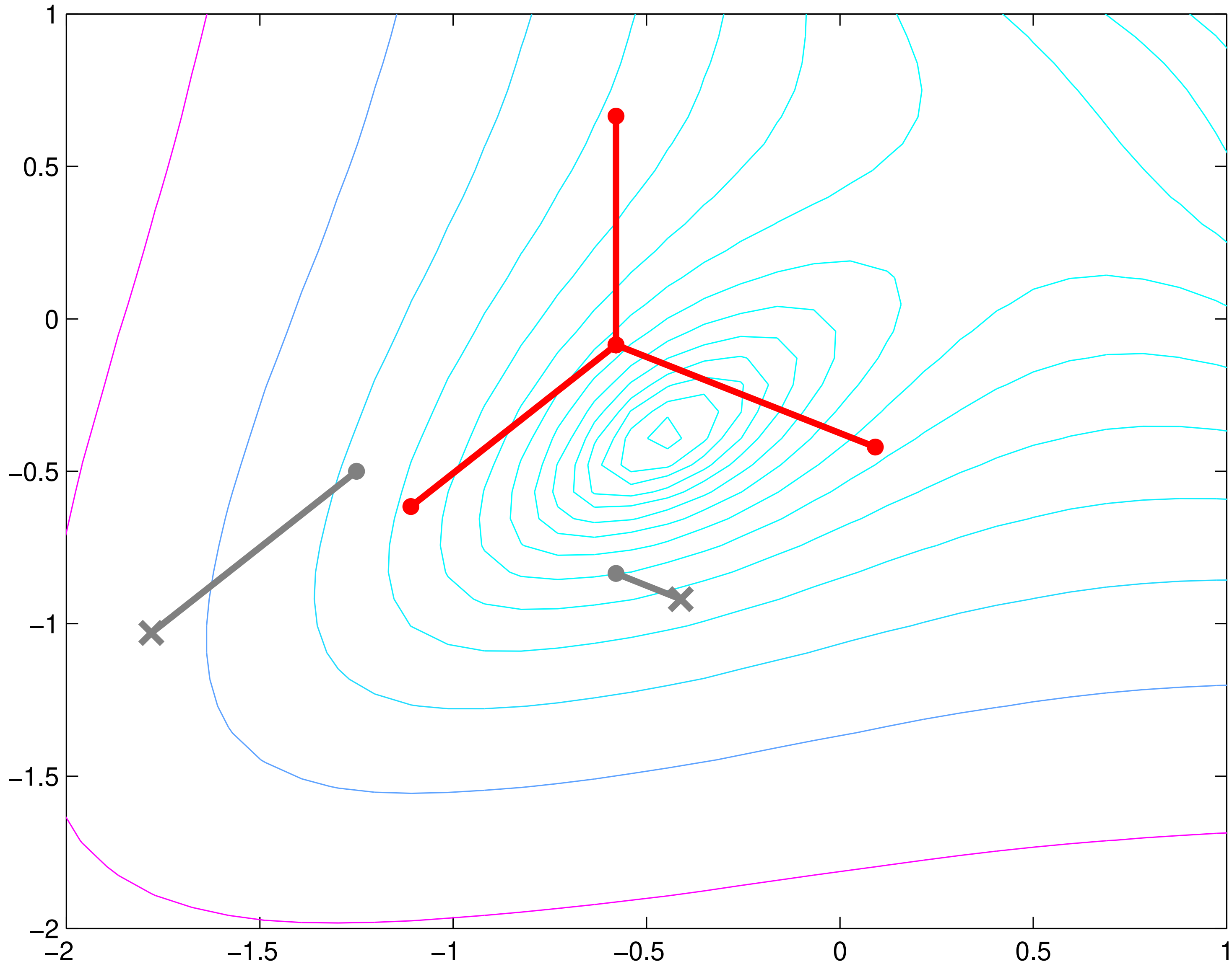
Asynchronous Parallel Pattern Search



Asynchronous Parallel Pattern Search



Asynchronous Parallel Pattern Search



Checking Convergence for Unconstrained APPS

We have *convergence* when enough search directions have converged to contain a positive basis.

Given a set of vectors $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$, we verify that $\mathcal{V}$ is a positive spanning set by solving a series of NNLS problems. Solve $n+1$ nonnegative least squares problems of the form

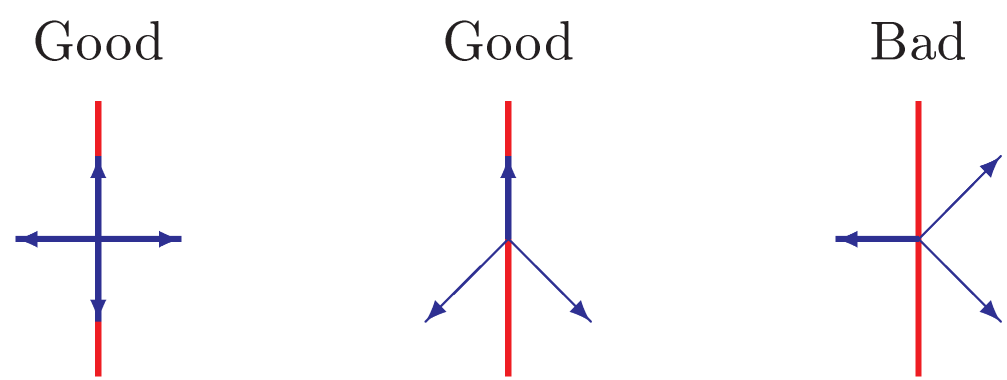
$$\| Vx - b \| \text{ s.t. } x_i \geq 0 \; \forall i$$

for $b = \{e_1, \ldots, e_n, -e\}$ *or any known positive basis.*

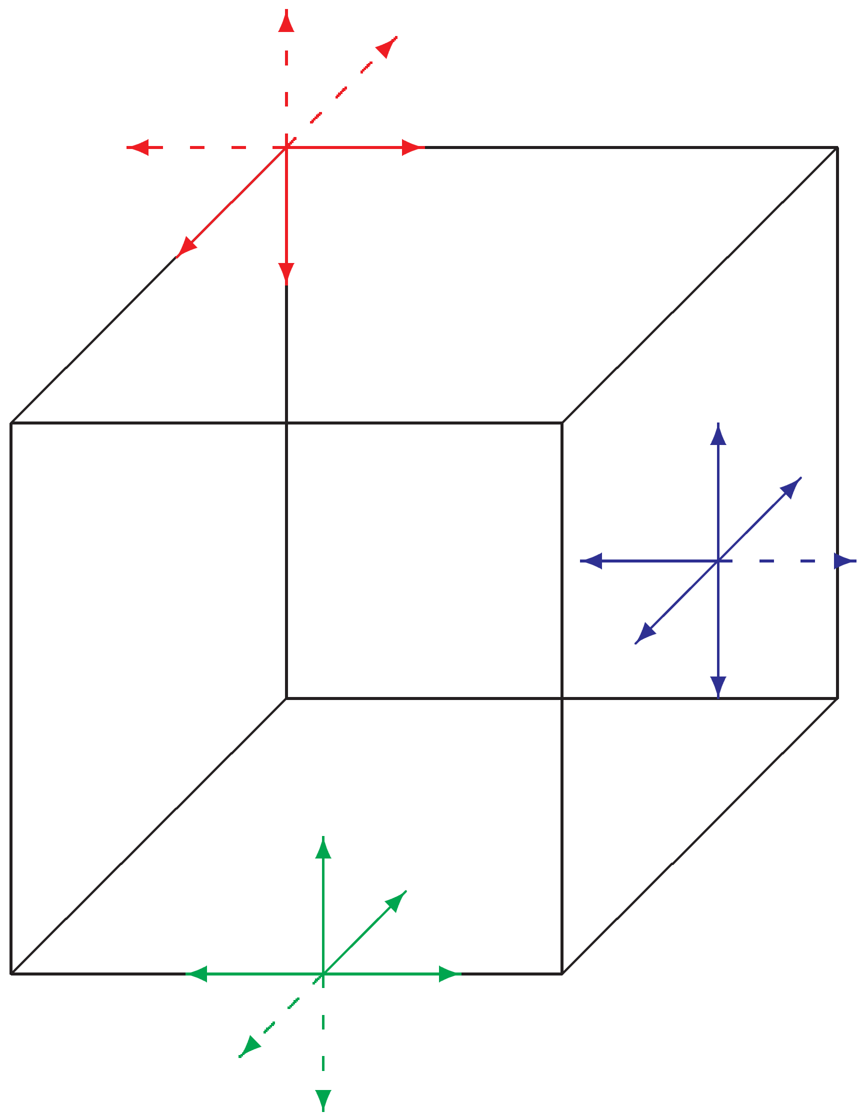
All have residual zero iff V is a positive spanning set.

APPS with Bound Constraints

- Must use plus/minus unit vectors for search.
- Infeasible points are mapped to the boundary.
- Change the positive basis test for convergence.
- Restarting procedure for failed search directions must always keep *base pattern* active.



Active & Inactive Bound Constraints



Checking Convergence for Bound Constrained APPS

We have convergence when enough *projected* search directions have converged to contain a *projected* positive basis.

Given a set of vectors $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$, we verify that $\mathcal{V}$ is a *projected* positive spanning set by solving a series of NNLS problems of the form

$$\| P(V)x - b \| \text{ s.t. } x_i > 0 \, \forall i$$

for each $b \in \mathcal{B}$, where

$$\begin{aligned} \mathcal{B} &= \{ e_i \mid \text{upper bound } i \text{ is inactive } \} \cup \\ &\quad \{ -e_i \mid \text{lower bound } i \text{ is inactive } \} \end{aligned}$$

All have residual zero iff V is a positive spanning set.

Example: An Electrical Circuit Simulation

- **Variables:** inductances, capacitances, diode saturation currents, transistor gains, leakage inductances, and transformer core parameters
- **Simulation Code:** SPICE3

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

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

The "Unconstrained" Problem

- Scaling of the variables is handled in the same way as it is for the bound constrained problem

$$s_i = \frac{u_i - l_i}{100}$$

The search directions are stretched rather than scaling the x-vectors.

- When an out-of-bounds x-vector is sent to the function evaluation, a value of $+\infty$ is returned.

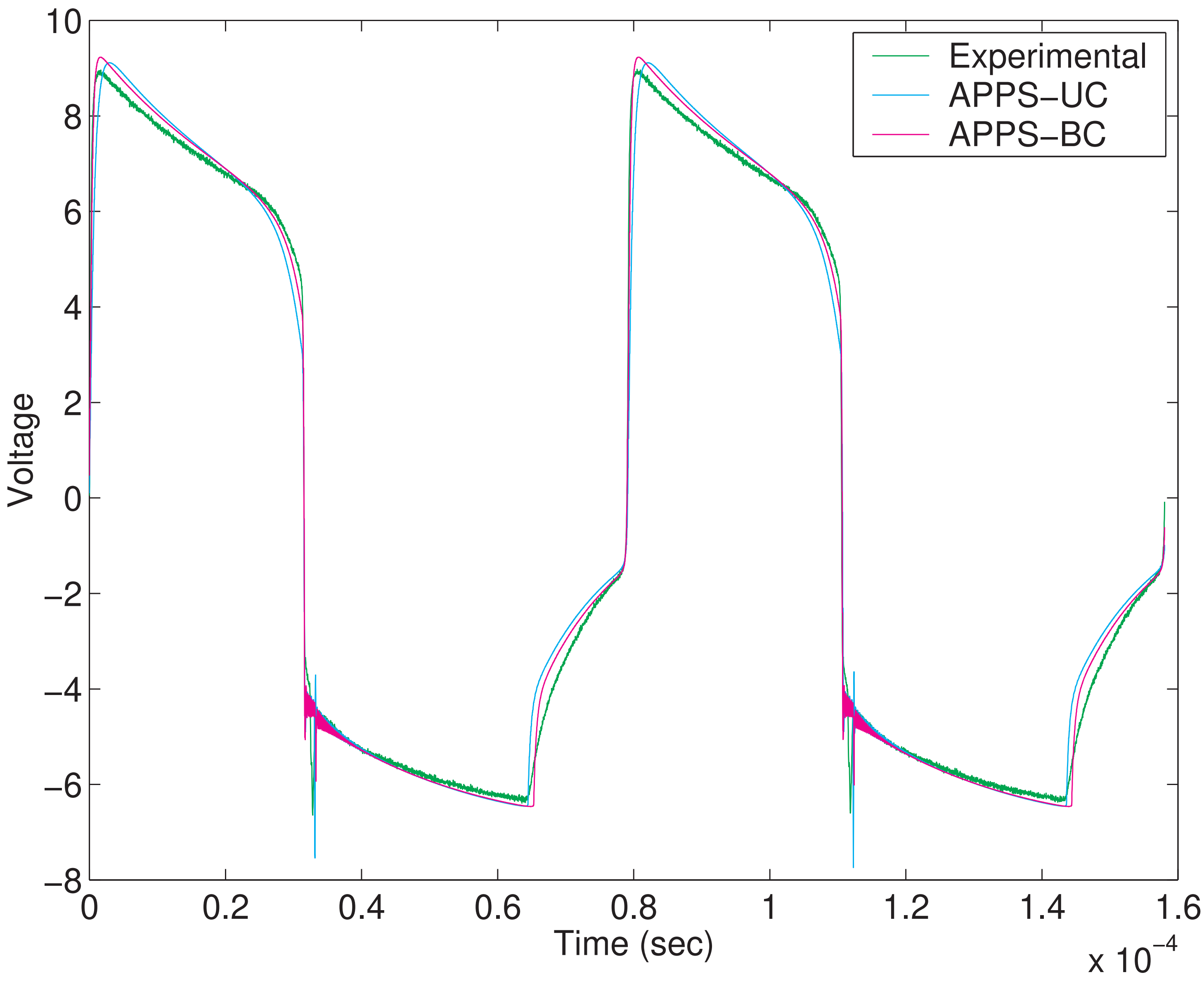
Numerical Results for Circuit Problem

Type	Search Dirs	Final f(x)	Time (sec)	Idle (sec)	Func. Evals	No-Gos
BC	34	25.3	2750	448	36	3
UC	34	45.1	2440	414	39	4

BC = Bound Constrained

UC = Unconstrained

Circuit Problem Results



Conclusions

- The ability to handle bound constraints is incorporated into APPS with minor changes.
- The most significant change is in the positive basis test, where we now incorporate information about active constraints.
- Numerical results indicate that using bound constraints in APPS may improve the final result.

Future Work

- Adding linear and nonlinear constraints.
- Convergence for bound constrained APPS.

APPSPACK

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

Recent Highlights

- Added support for bound constraints
- Added MPI support (Alton Patrick)

Coming Soon

- Function Value Cache (Alton Patrick)
- Model-Assisted Pattern Search (Sarah Brown)