

# Optimization, Engineering & Cluster Computing

TAMARA G. KOLDA

Sandia National Labs  
Livermore, CA

*This work was supported by the U.S. Department of Energy.*

“It has often been said that a person does not really understand something until he teaches it to someone else. Actually a person does not really understand something until he can teach it to a *computer*, i.e., expresses it as an algorithm.”

— Donald E. Knuth, 1974,  
Computer Science and Its Relation to Mathematics,  
*American Mathematical Monthly*.

## OPTIMIZATION: NEWTON'S METHOD

1. Construct a *quadratic model* about iterate  $x_k$ :

$$f(x_k + s) \approx f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s$$

where  $g_k = \nabla f(x_k)$  and  $H_k = \nabla^2 f(x_k)$ .

2. Use the model to predict a good *descent direction*:

$$s_k = -H_k^{-1} g_k$$

3. Perform a *line search* to determine the next iterate:

$$x_{k+1} = x_k + \alpha_k s_k$$

*There's 30+ years of theory behind these methods...*

... BUT NEWTON'S METHOD  
DOESN'T ALWAYS WORK

Newton's method depends on access to gradient information, but for many engineering problems, we do not have access to such information! The types of problems we are interested in are characterized as follows:

- The evaluation of  $f$  is time-consuming,
- the gradient cannot be calculated, and
- the function values are not accurate enough for finite-difference gradient approximations.

## EXAMPLE: DETERMINING THE CHARACTERISTICS OF A CIRCUIT

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

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

$x$  = 17 unknown characteristics

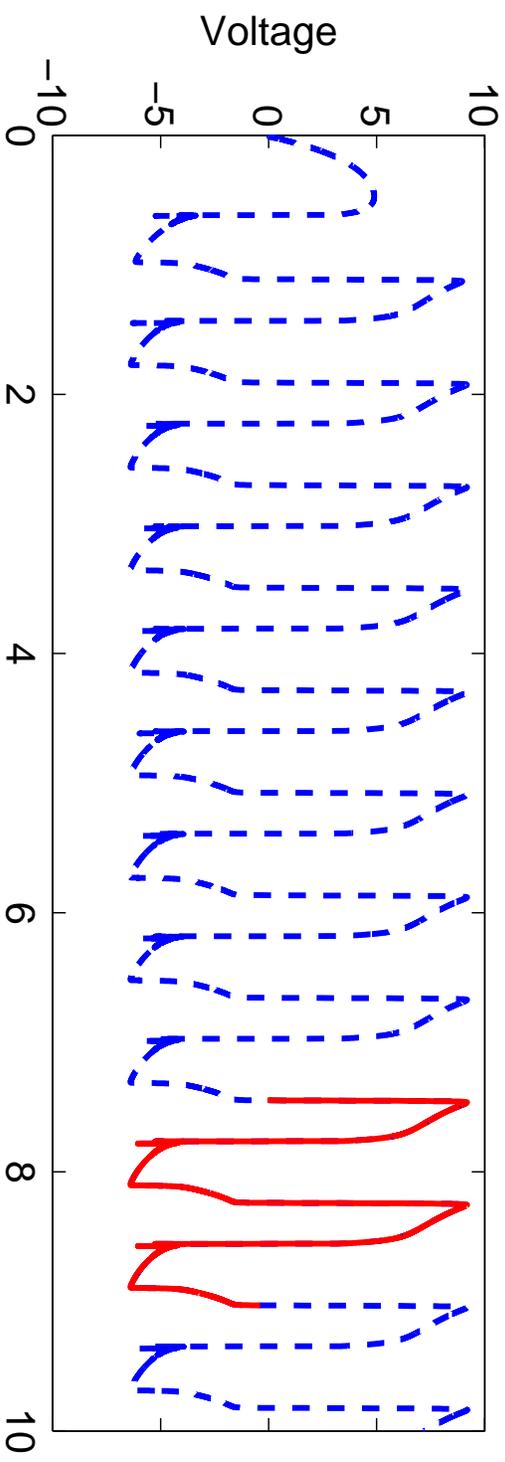
$V_t^{\text{SIM}}(x)$  = Simulation voltage at time  $t$

$V_t^{\text{EXP}}$  = Experimental voltage at time  $t$

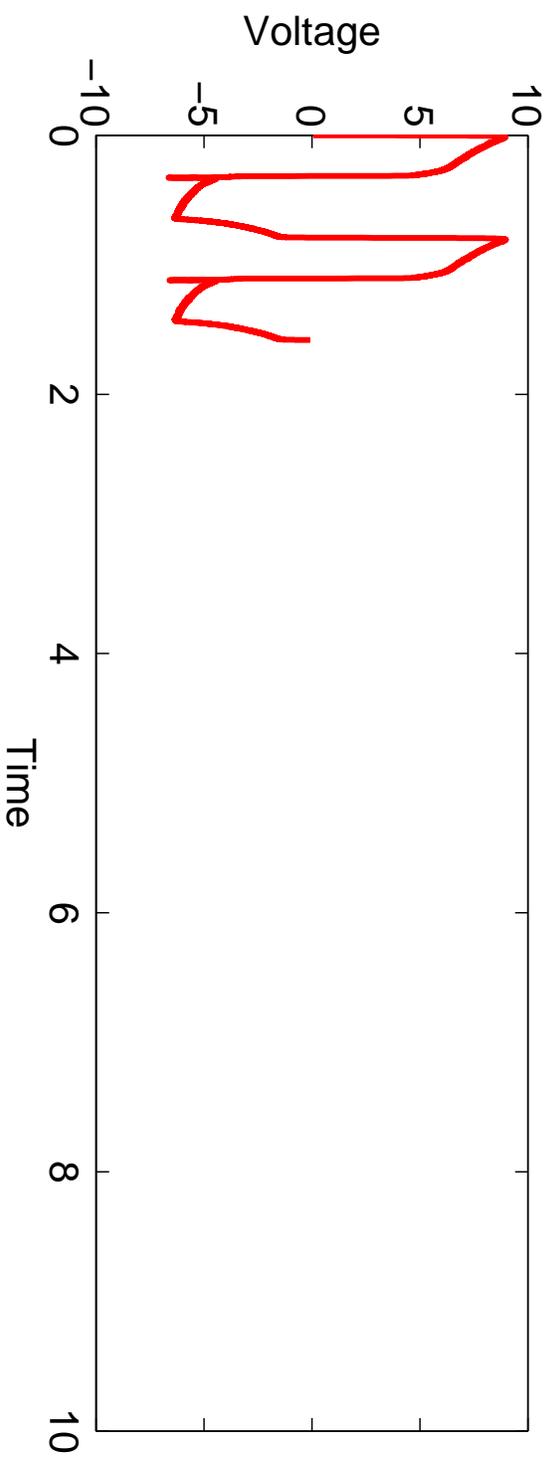
$N$  = Number of timesteps

# CIRCUIT SIMULATION

Simulation Output



Experimental Data



# DIRECT SEARCH

In *Direct Search Methods: Once Scorned, Now Respectable* (1996), Margaret Wright describes direct search methods as follows.

- A direct search uses only function values.
- A direct search method does not “in its heart” develop an approximate gradient.

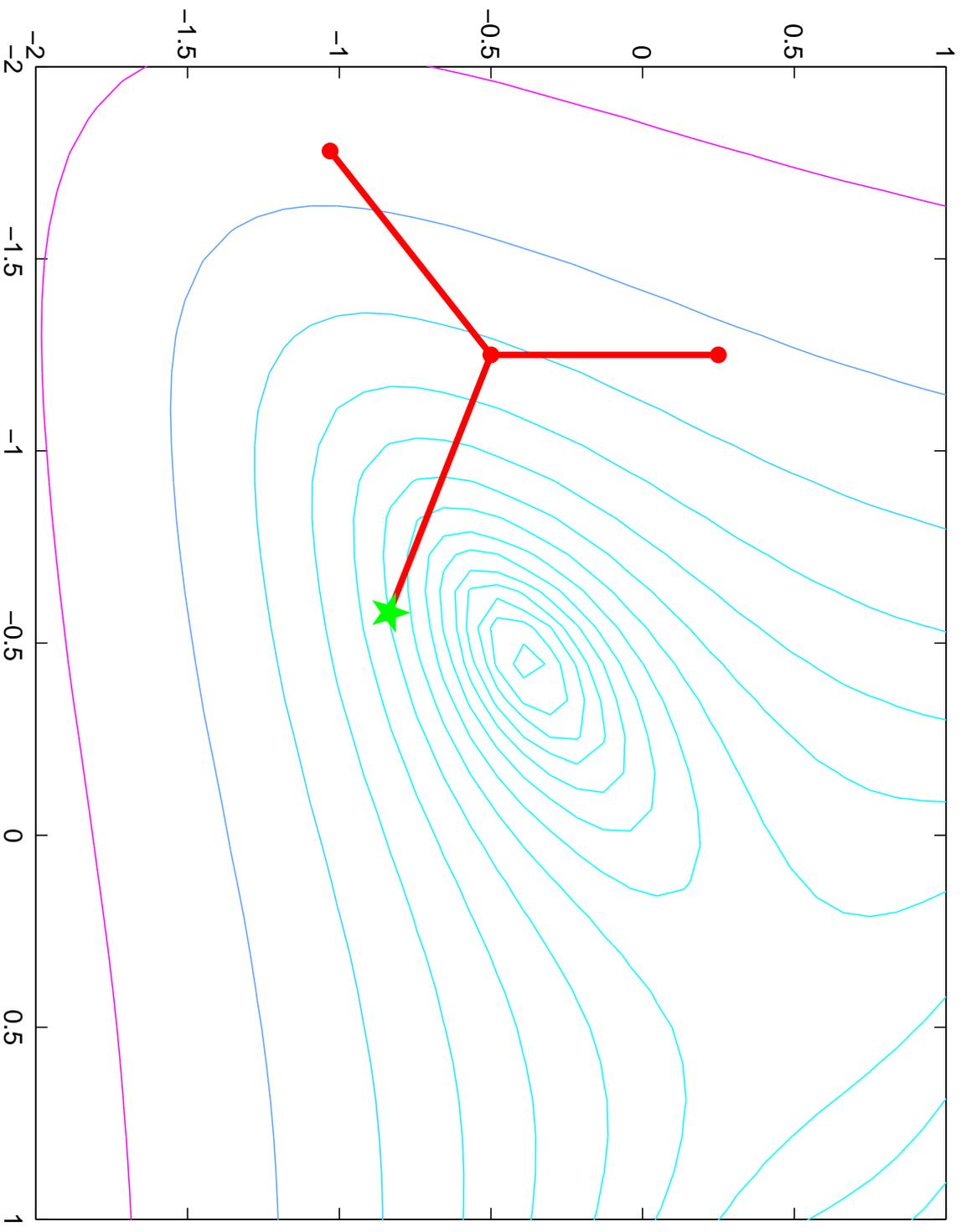
In particular, we are interested in *pattern search methods*, a type of direct search method.

- Popular in 1960s — Box, Hooke & Jeeves, etc.
- Multidimensional Search (MDS), Dennis & Torczon, 1991

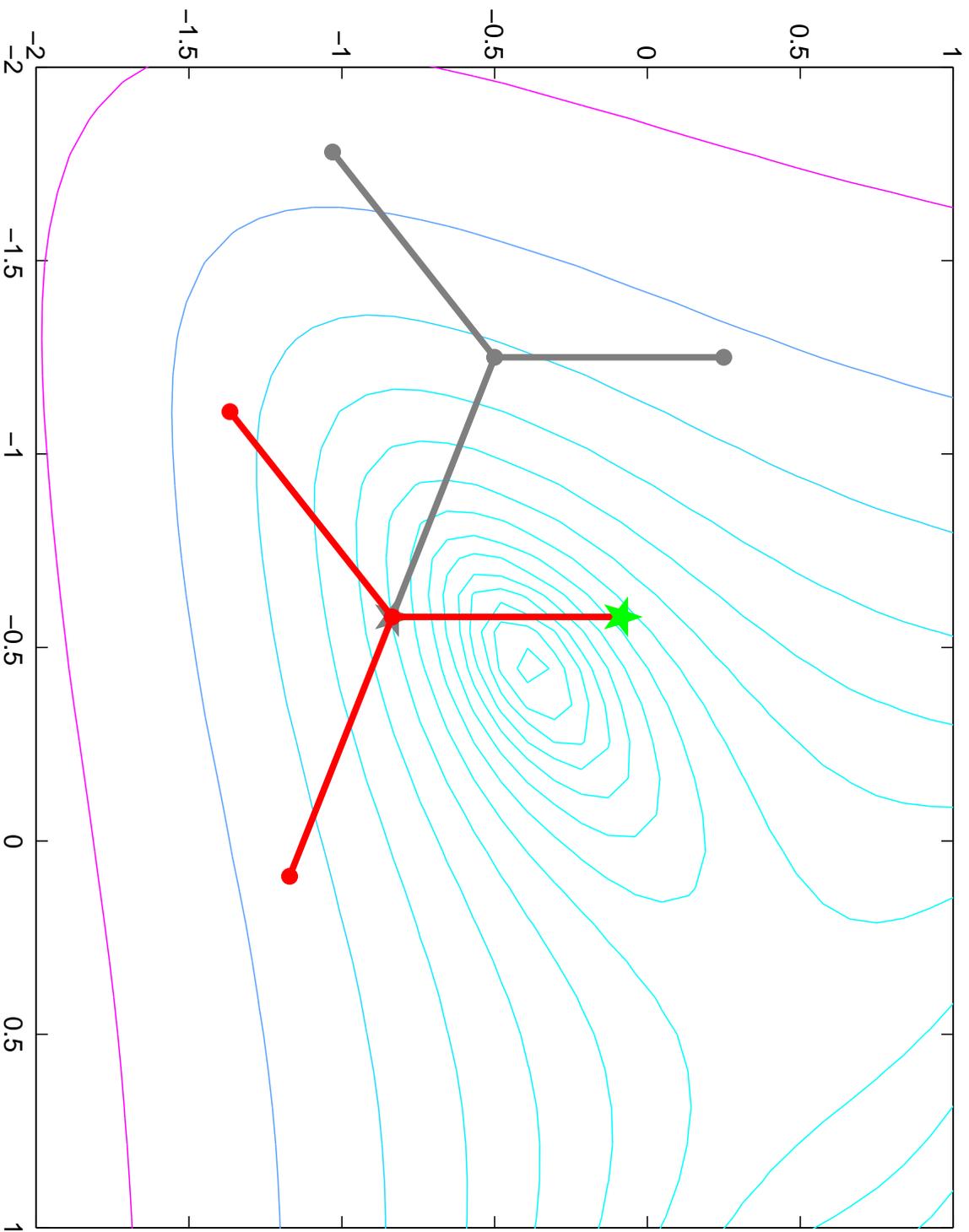
“The *search* for truth is more precious than its possession.”

— Albert Einstein (1879–1955)

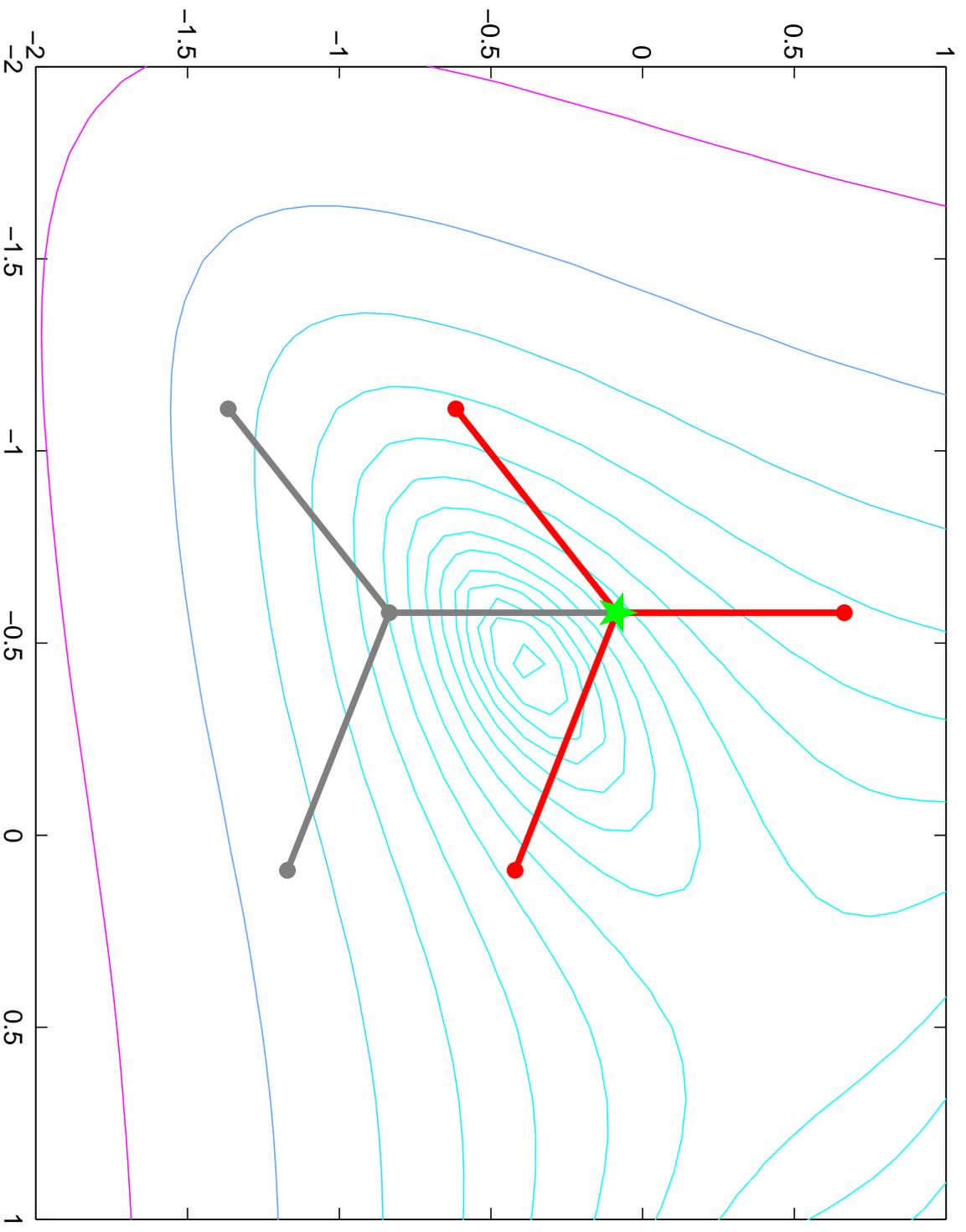
# PATTERN SEARCH



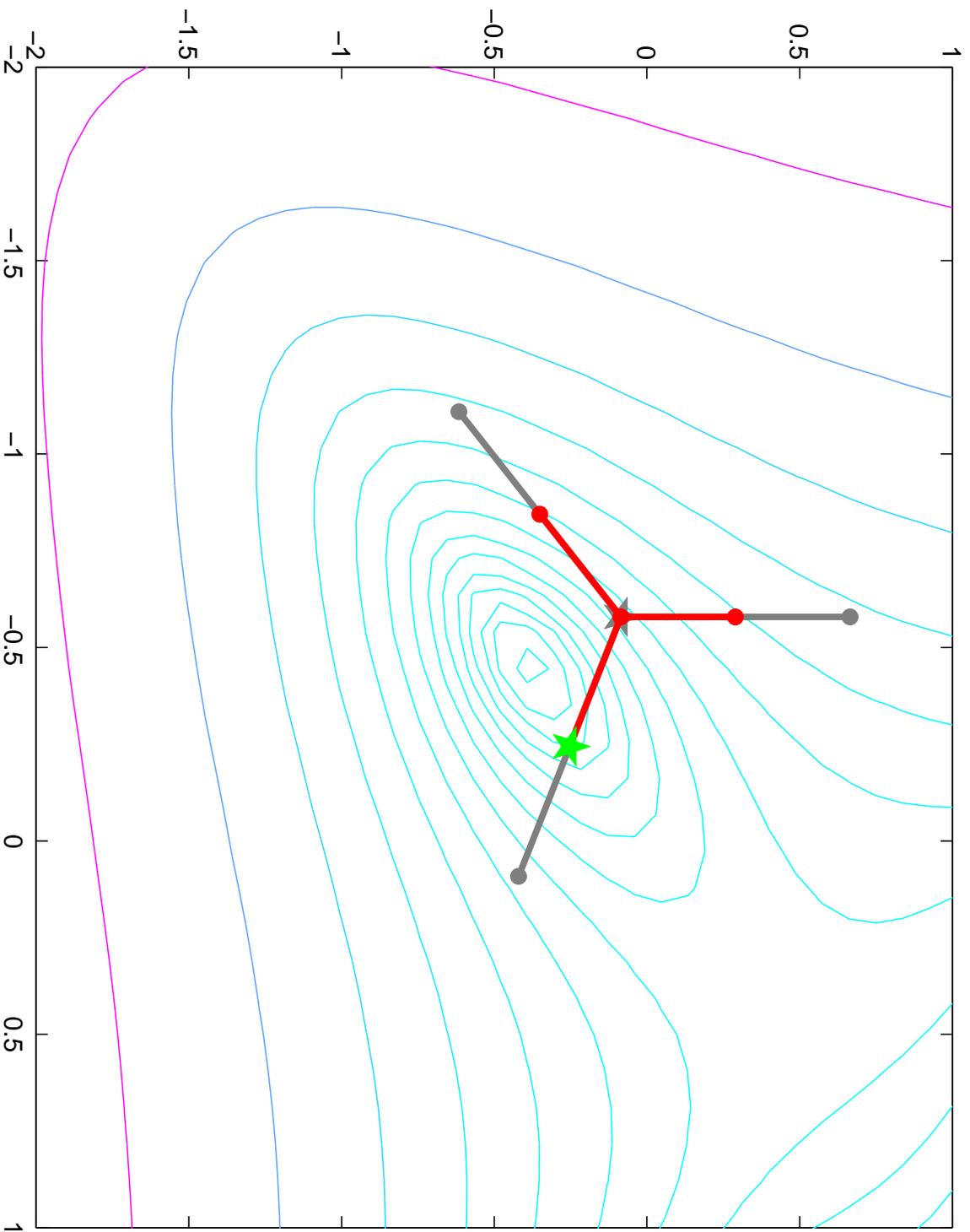
# PATTERN SEARCH



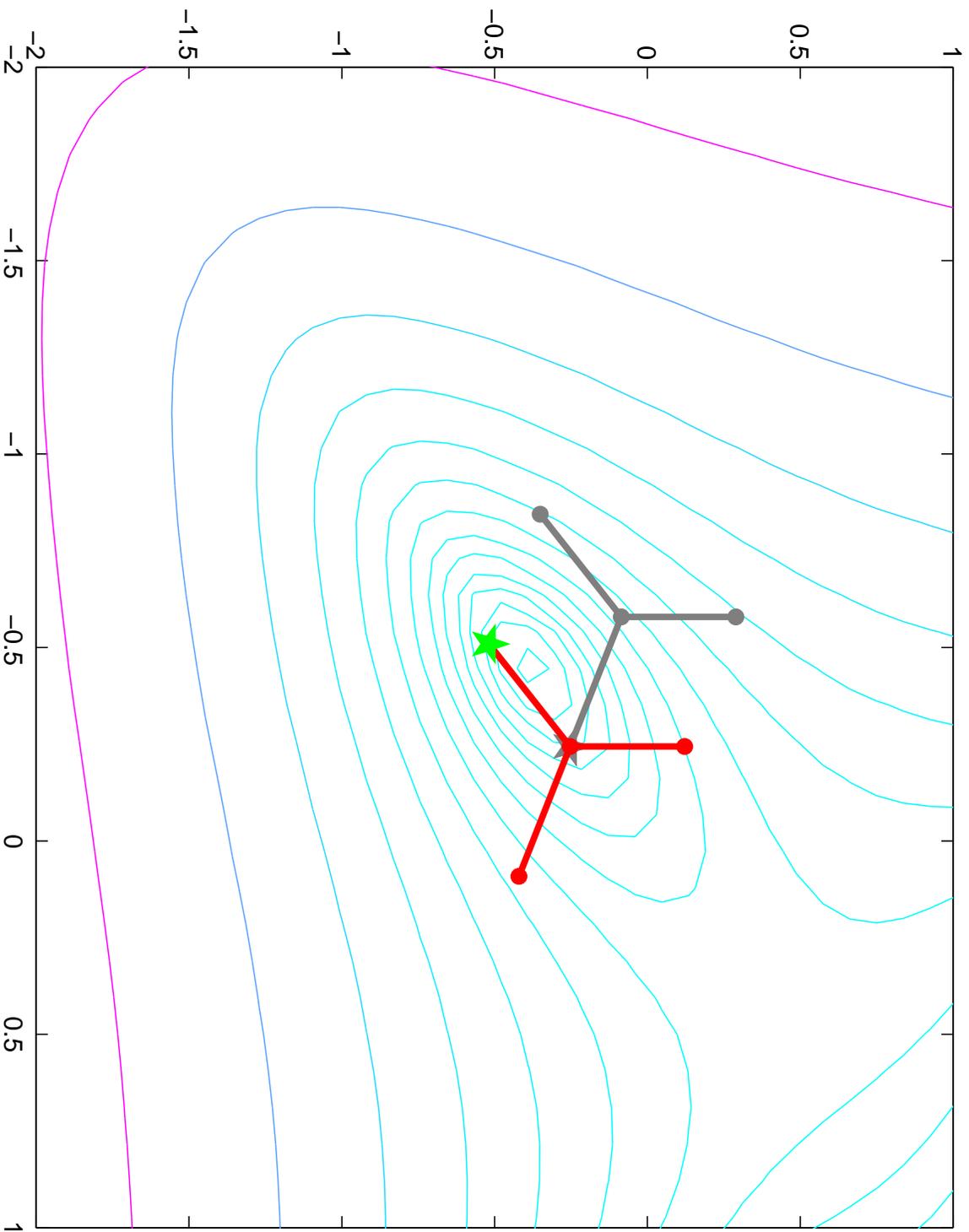
# PATTERN SEARCH



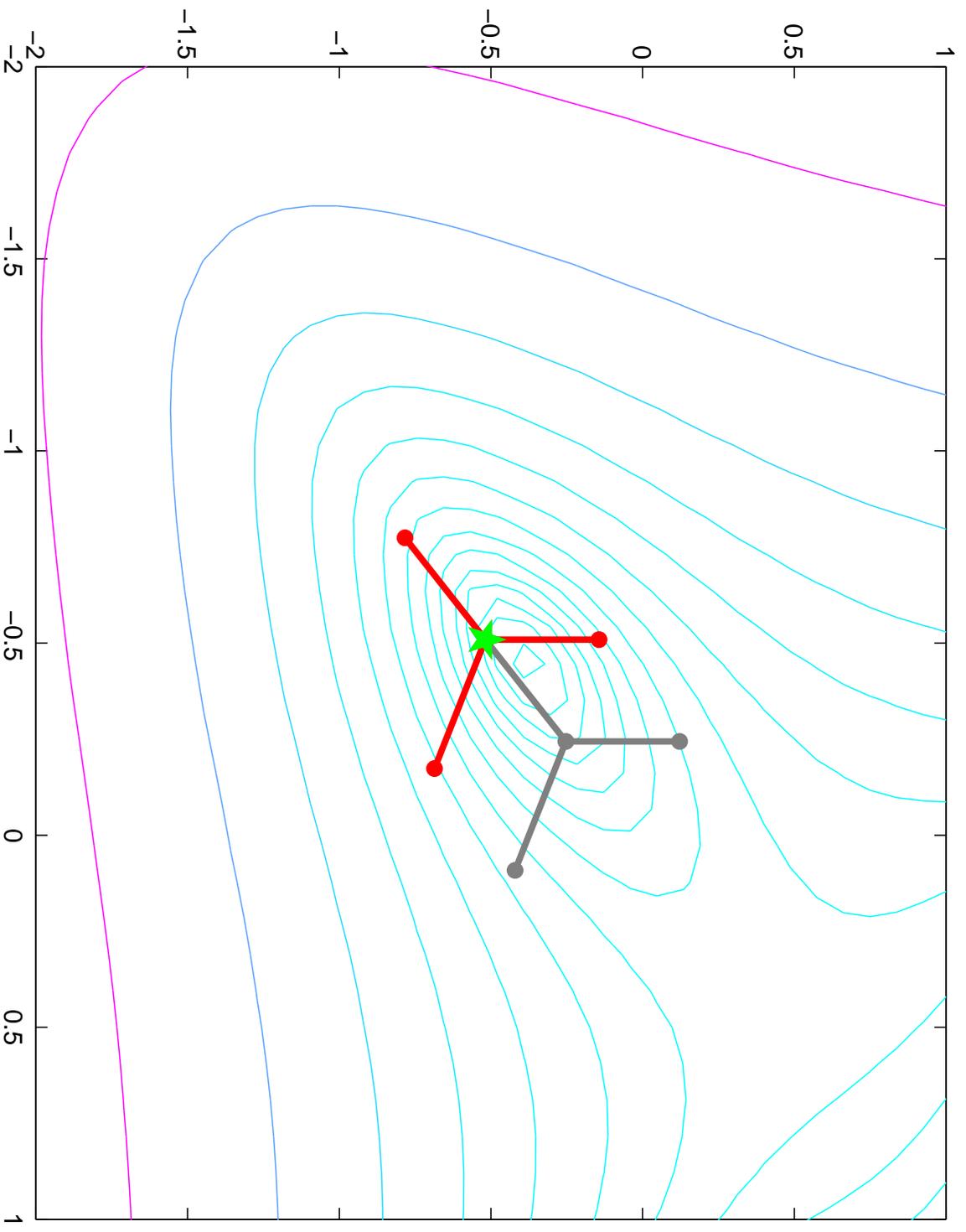
# PATTERN SEARCH



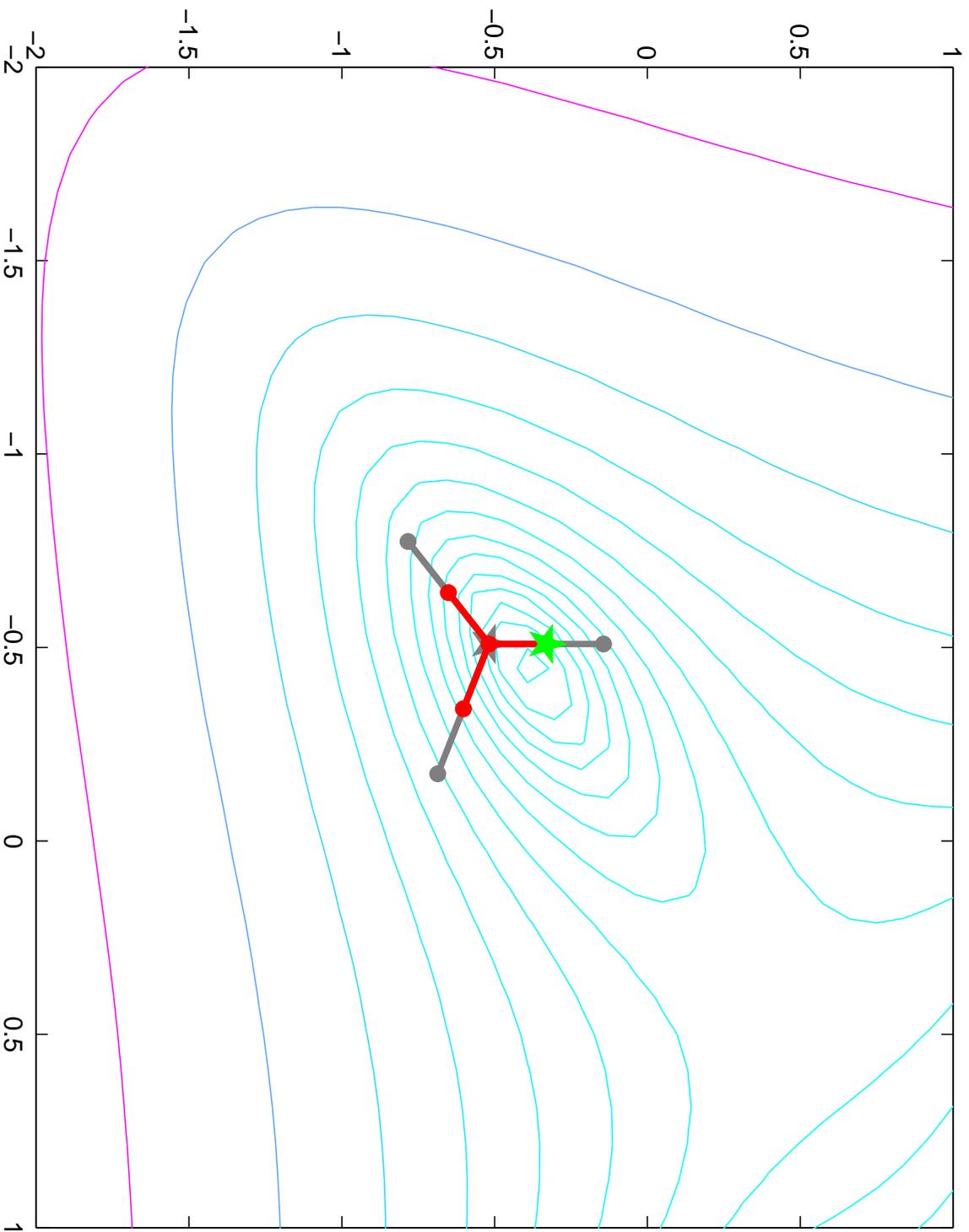
# PATTERN SEARCH



# PATTERN SEARCH



# PATTERN SEARCH



## CHOOSING THE SEARCH DIRECTIONS

The pattern must be chosen so that it positively spans  $\mathfrak{R}^n$ .

**Defn:** A set of vectors  $\{d_1, \dots, d_p\}$  *positively spans*  $\mathfrak{R}^n$  if any vector  $v \in \mathfrak{R}^n$  can be written as

$$v = \alpha_1 d_1 + \dots + \alpha_p d_p, \quad \alpha_i \geq 0 \quad \forall i.$$

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

**Fact:** If  $\{d_1, \dots, d_p\}$  positively spans  $\mathfrak{R}^n$ , then for any  $v \neq 0$ , there exists  $d_i$  such that  $d_i^T v > 0$ .

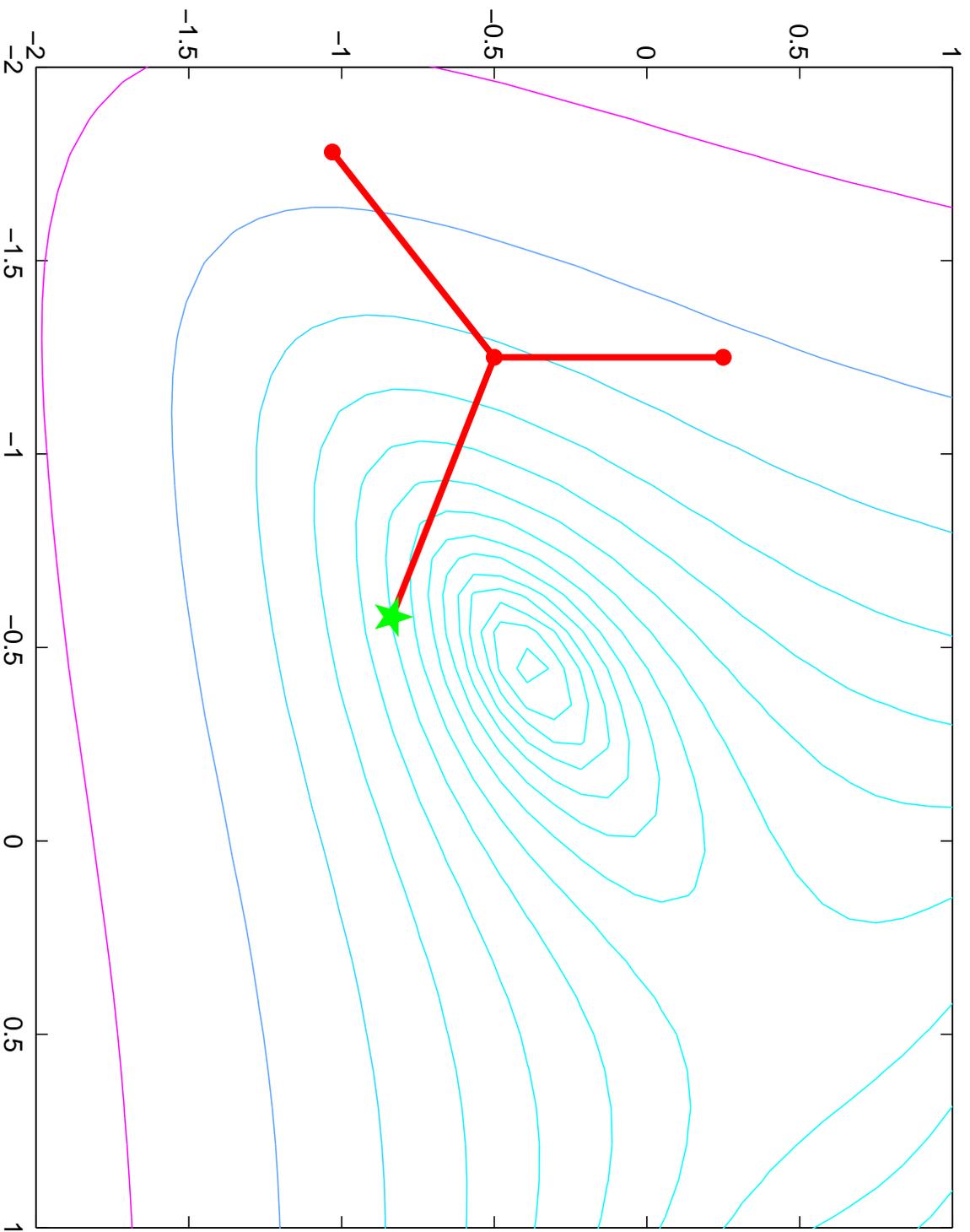
Lewis & Torczon (1996)

## PARALLEL COMPUTING

“In pioneer days they used oxen for heavy pulling, and when one ox couldn’t budge a log, they didn’t try to grow a larger ox. We shouldn’t be trying for bigger computers, but for more systems of computers.”

—Grace Hopper, 1906–1992

# PARALLEL PATTERN SEARCH



# Parallel PATTERN SEARCH ALGORITHM

*This algorithm is from a single processor's point-of-view.*

1. Compute  $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}} d$ , and evaluate  $f_{\text{trial}} \leftarrow f(x_{\text{trial}})$ .
2. Determine  $\{x_{\text{new}}, f_{\text{new}}\}$  via a **global reduction** on all  $\{x_{\text{trial}}, f_{\text{trial}}\}$  values.
3. If  $f_{\text{new}} < f_{\text{best}}$ , replace  $\{x_{\text{best}}, f_{\text{best}}\} \leftarrow \{x_{\text{new}}, f_{\text{new}}\}$ , and reset  $\Delta_{\text{trial}} = 2^\ell \Delta_{\text{trial}}$ ,  $\ell \geq 0$ .  
Else  $\Delta_{\text{trial}} \leftarrow \frac{1}{2} \Delta_{\text{trial}}$ .
4. If  $\Delta_{\text{trial}} > \text{tol}$ , go to Step 1. Else, exit.

## NEW ISSUES

- Simulations are more complicated, and the execution time may vary substantially depending on the inputs. Further, simulations may take minutes, hours, or even days to execute.
- The parallel platform is now a heterogeneous cluster of workstations rather than a tightly coupled network of homogeneous processors.
- Faults — both of the simulation and the hardware — are real issues.

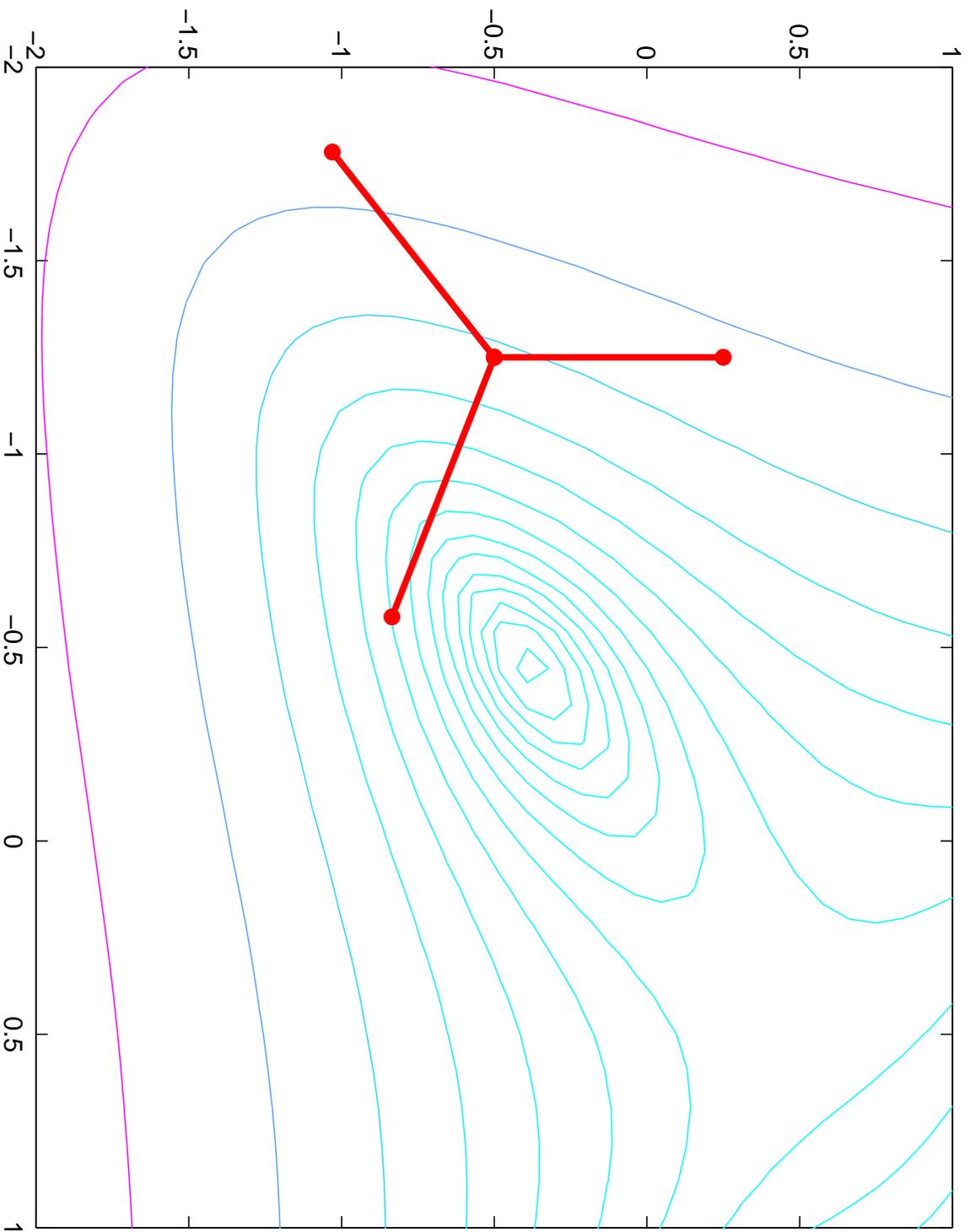
We are motivated to revisit parallel pattern search to take these new issues into account...

*Asynchronous* PARALLEL  
PATTERN SEARCH (APPS)

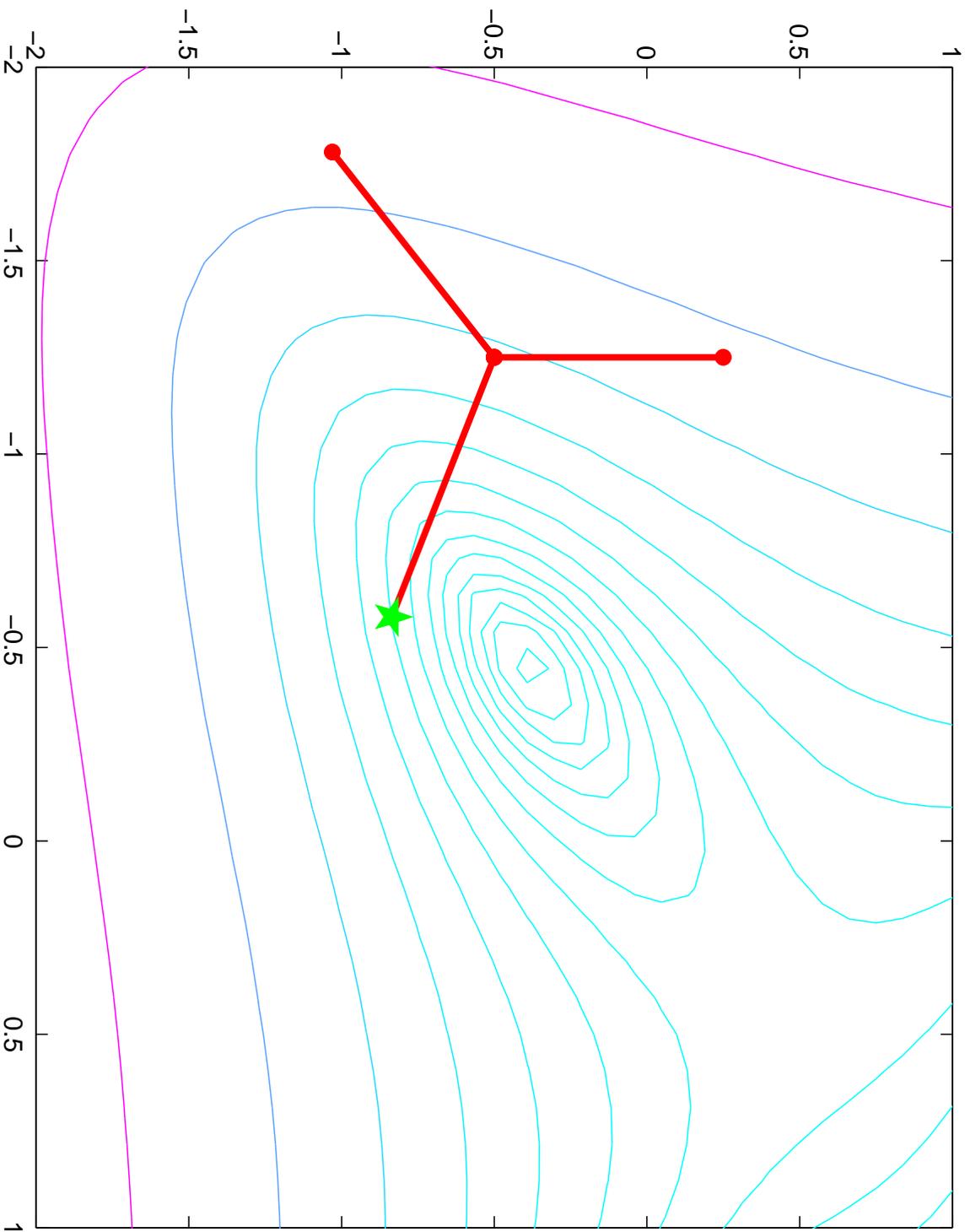
“In great mathematics there is a very high degree of unexpectedness, combined with inevitability and economy.”

— Godfrey Hardy (1877 - 1947)  
A Mathematician’s Apology, 1941.

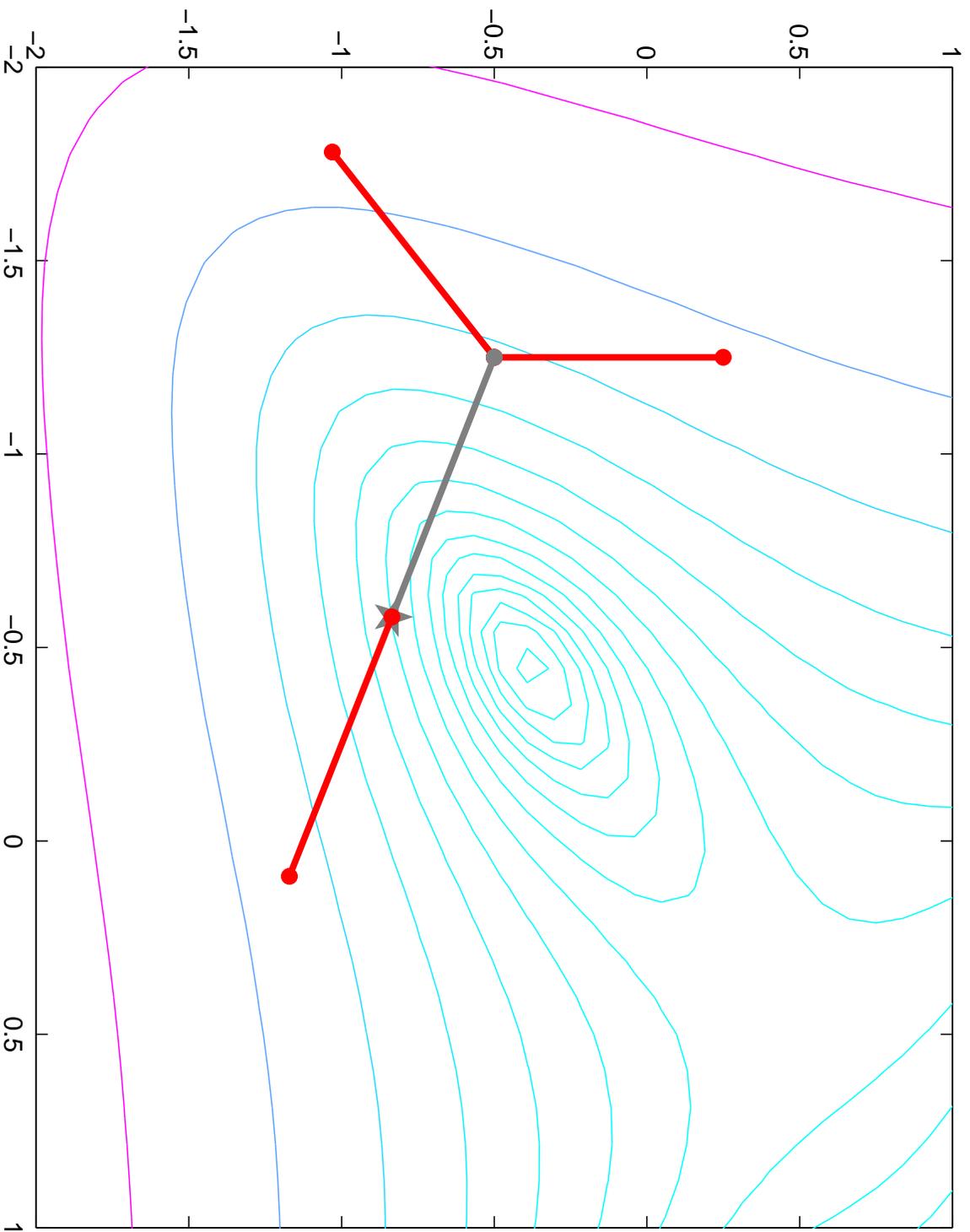
# ASYNCHRONOUS PARALLEL PATTERN SEARCH



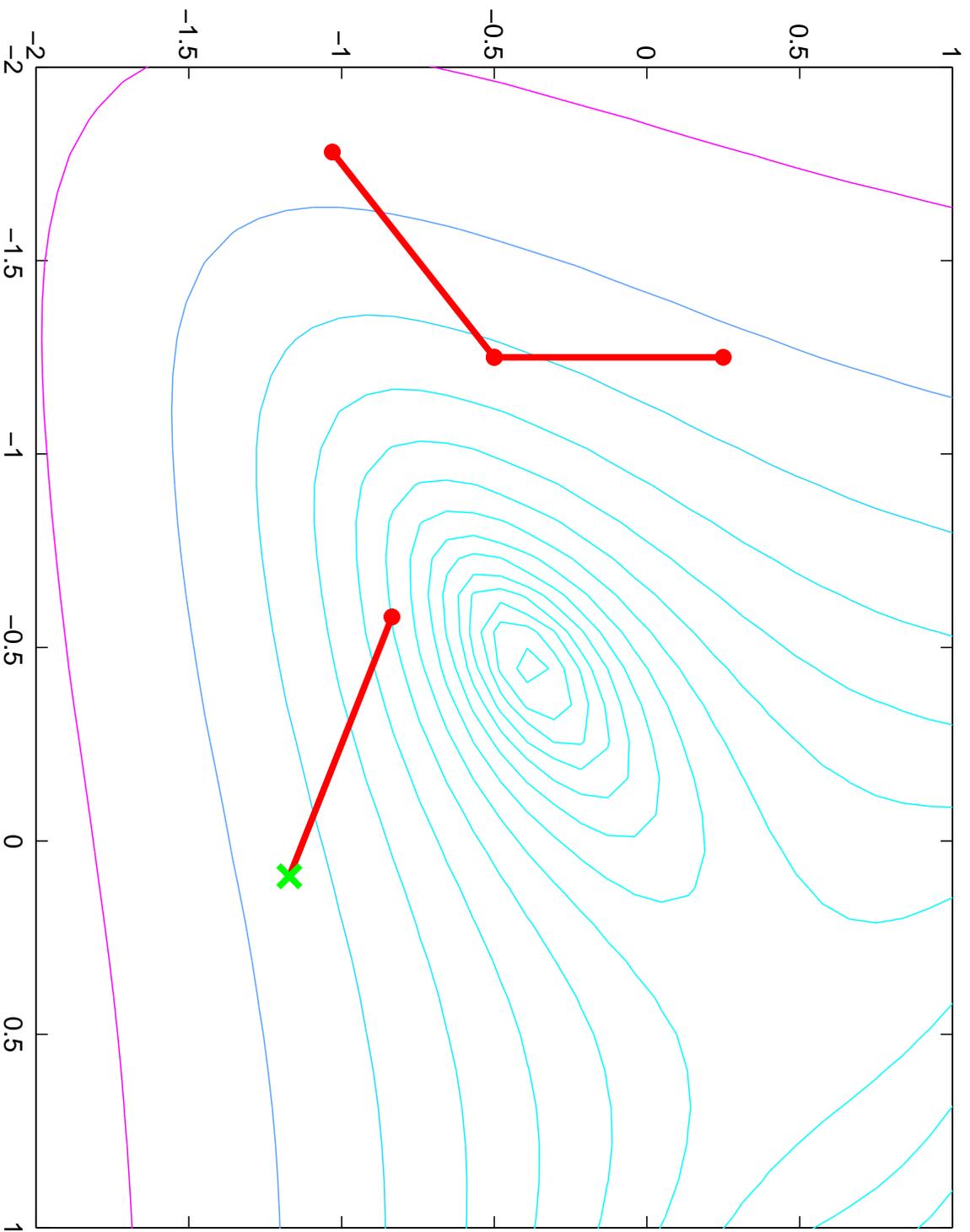
# ASYNCHRONOUS PARALLEL PATTERN SEARCH



# ASYNCHRONOUS PARALLEL PATTERN SEARCH

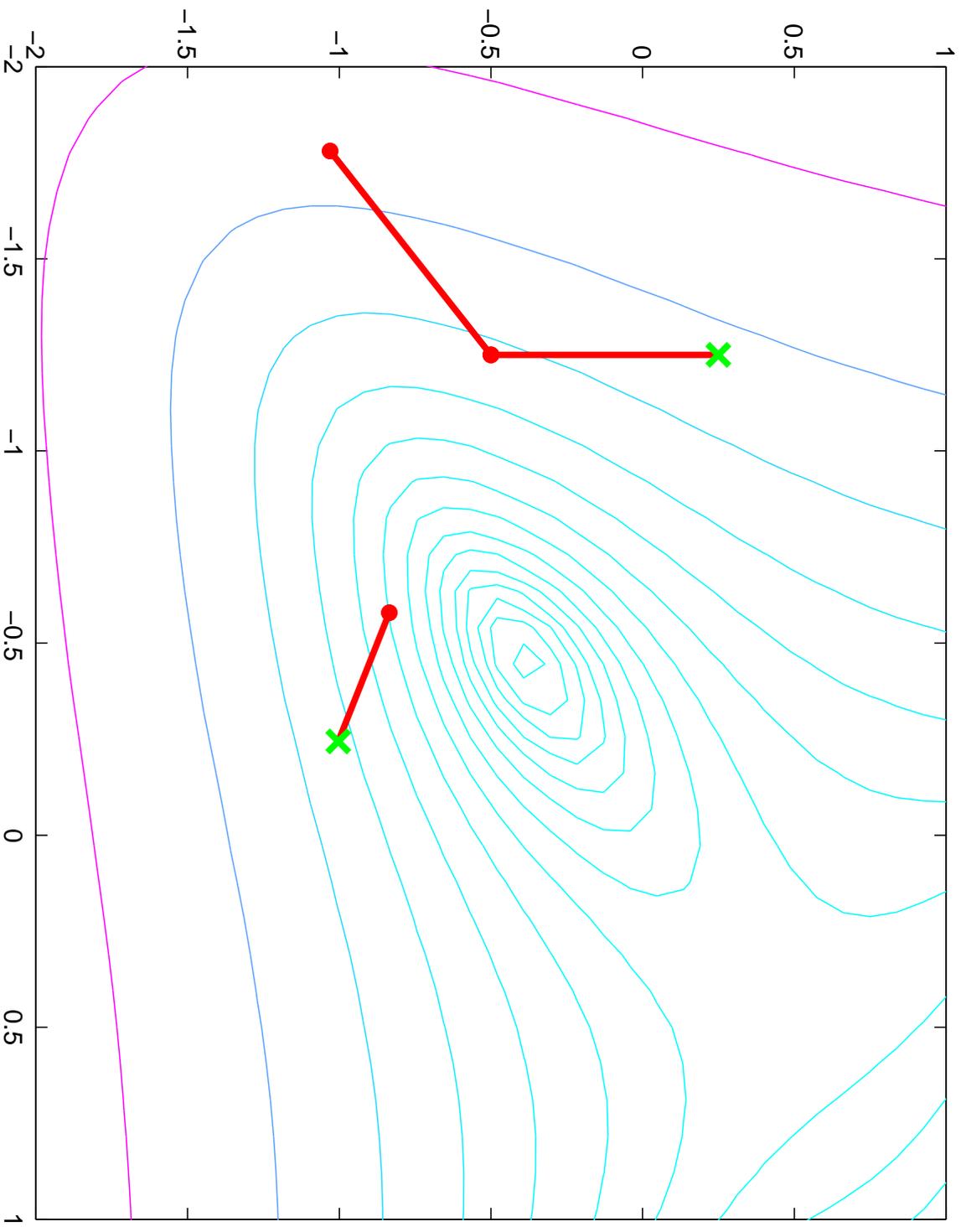


# ASYNCHRONOUS PARALLEL PATTERN SEARCH

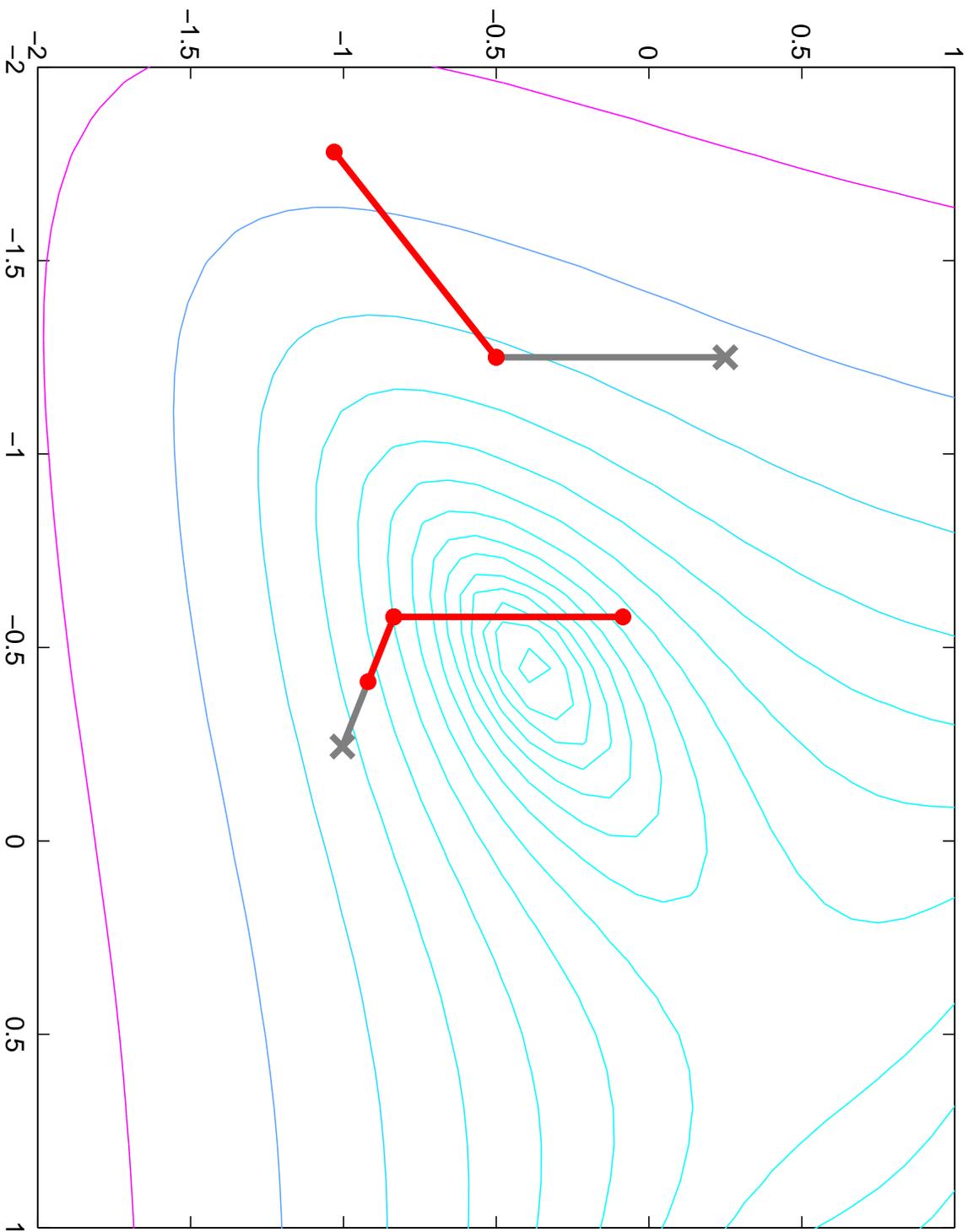




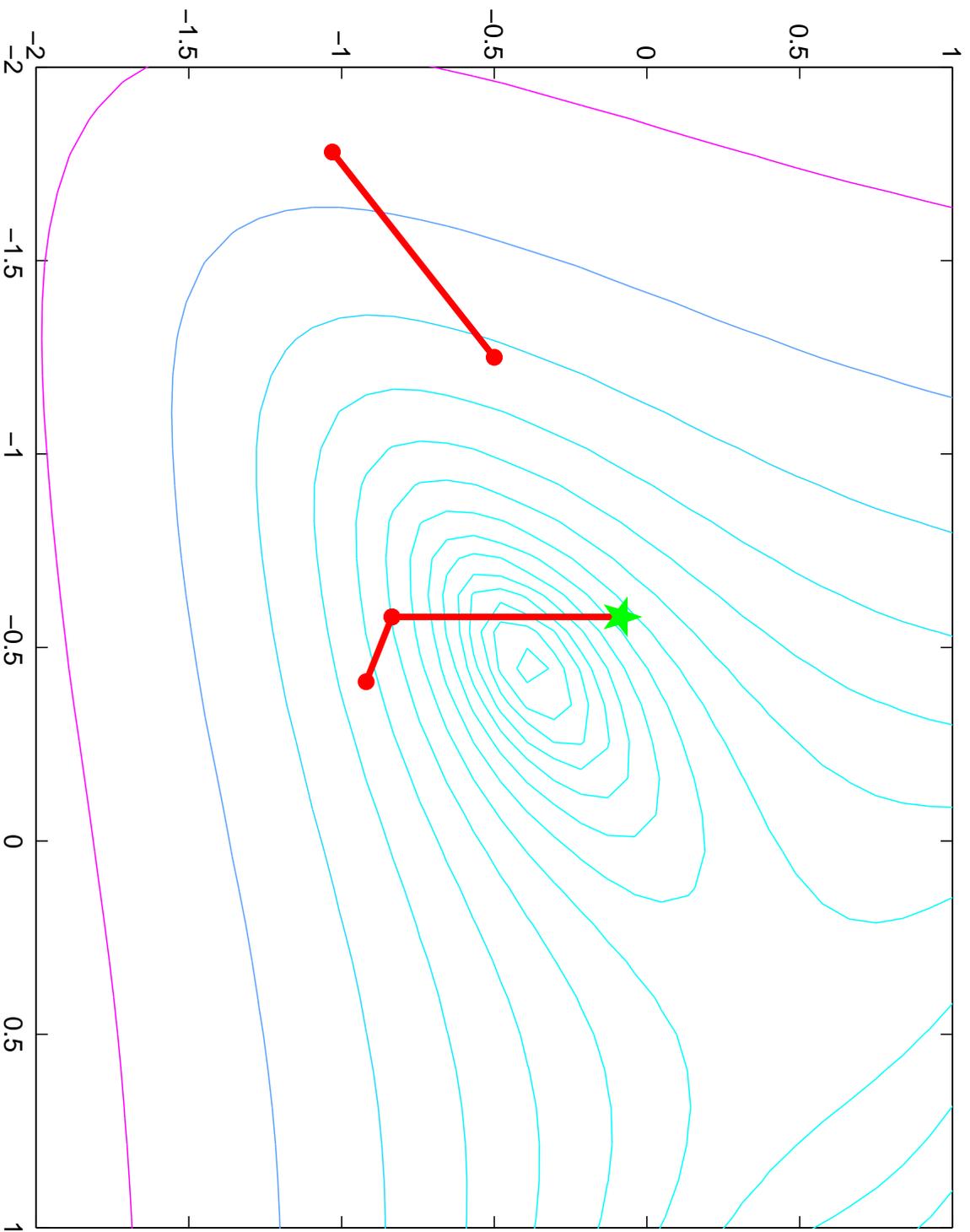
# ASYNCHRONOUS PARALLEL PATTERN SEARCH



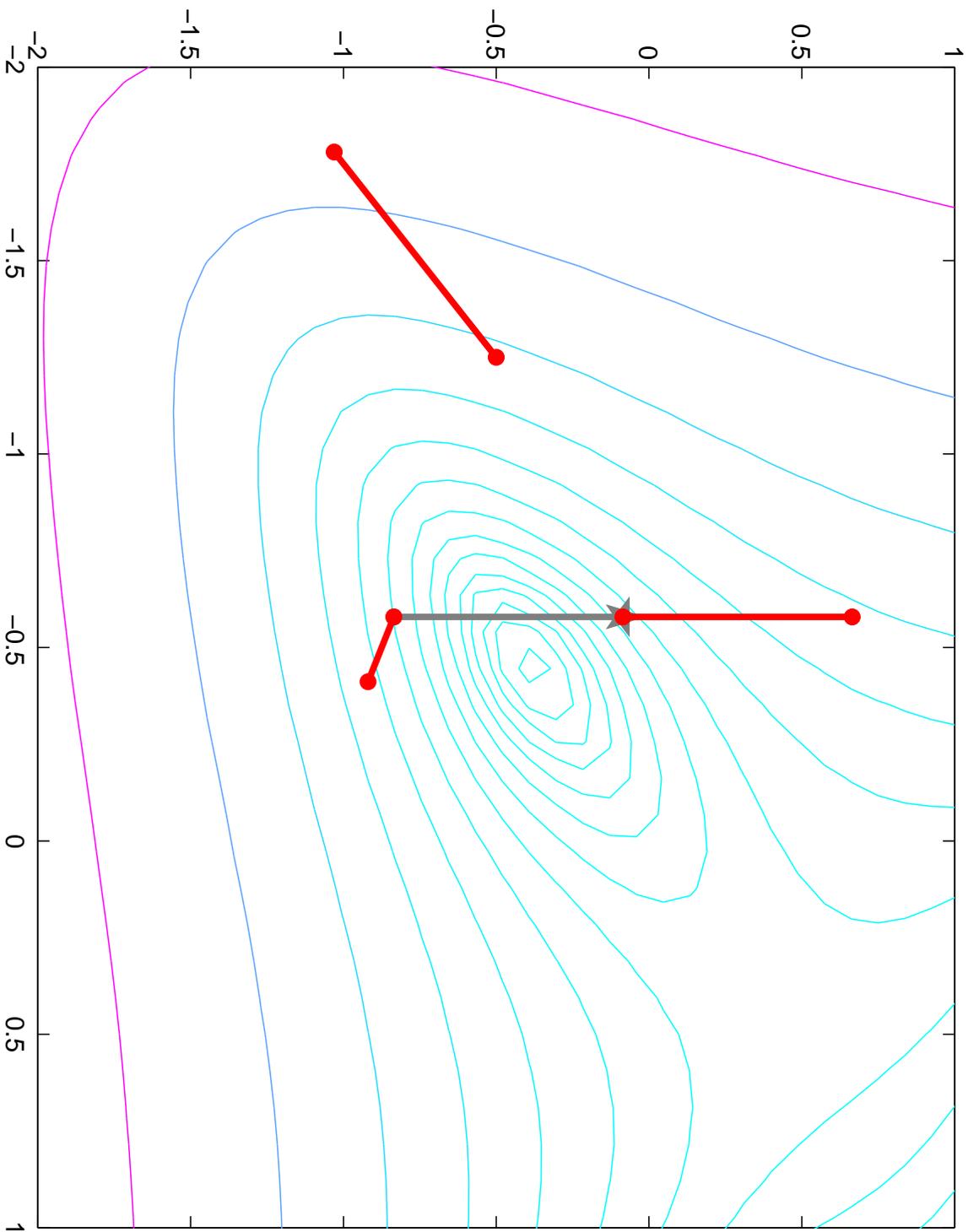
# ASYNCHRONOUS PARALLEL PATTERN SEARCH



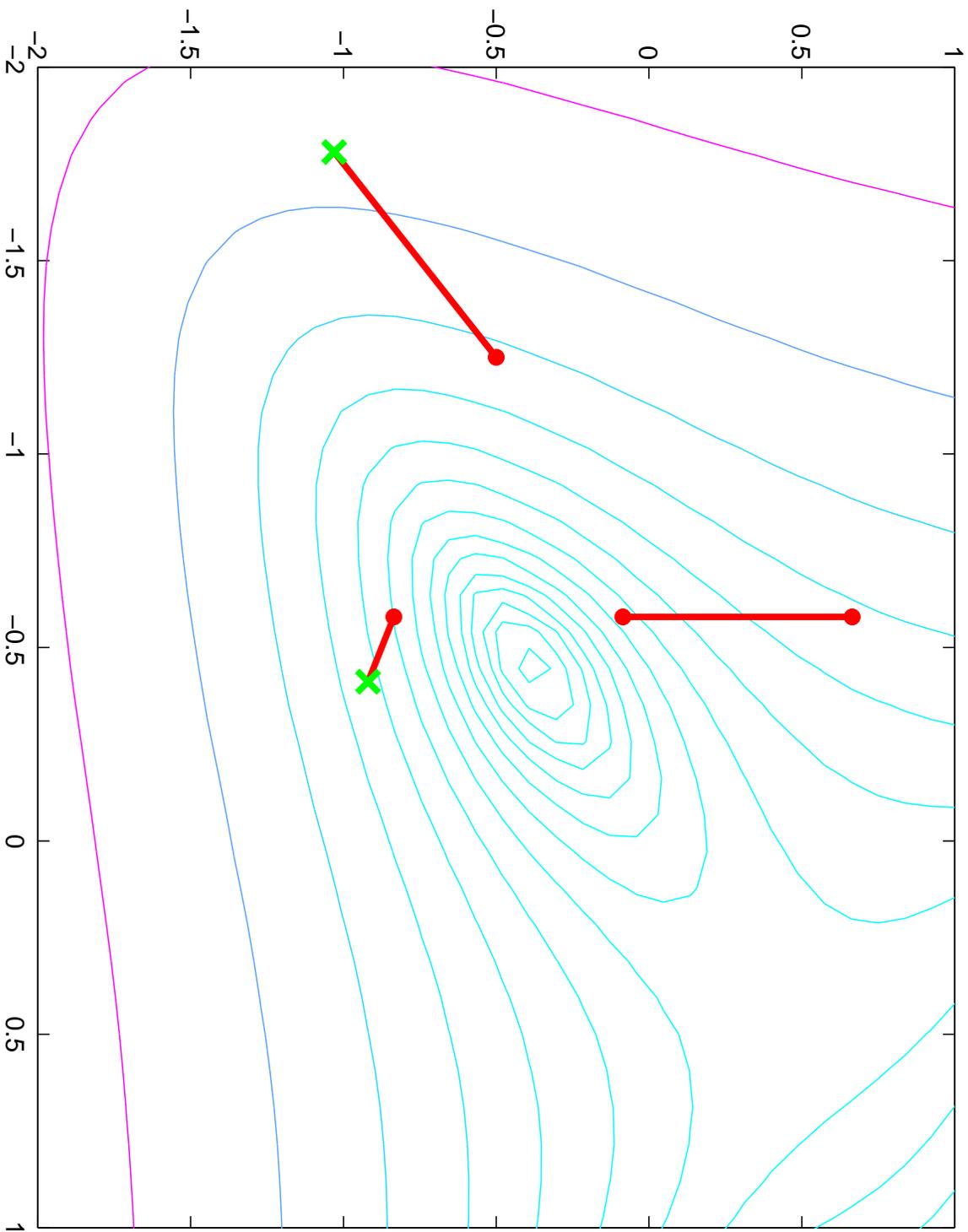
# ASYNCHRONOUS PARALLEL PATTERN SEARCH



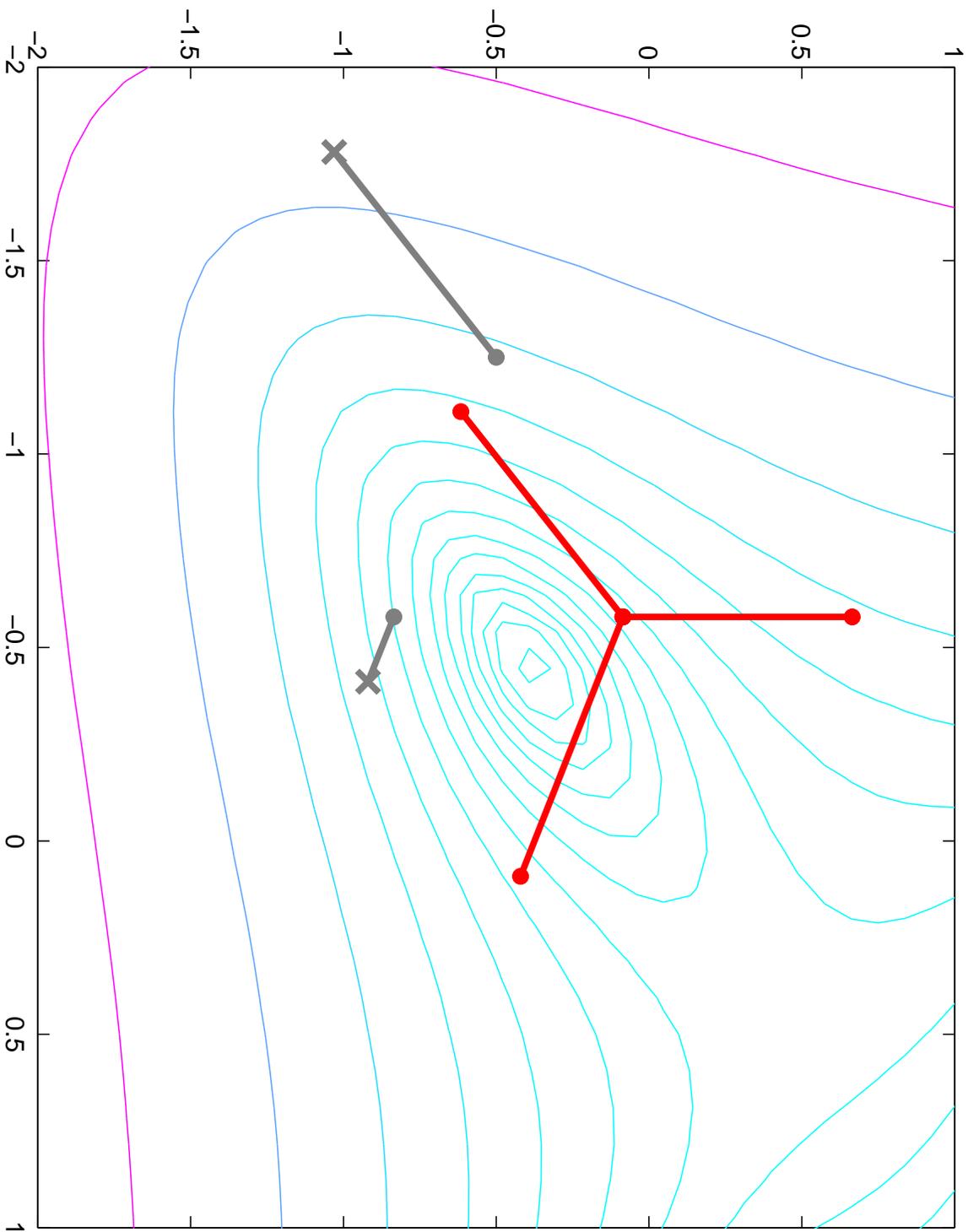
# ASYNCHRONOUS PARALLEL PATTERN SEARCH



# ASYNCHRONOUS PARALLEL PATTERN SEARCH



# ASYNCHRONOUS PARALLEL PATTERN SEARCH



# ASYNCHRONOUS PARALLEL PATTERN SEARCH

0. Consider each incoming triplet  $\{x_{\text{new}}, f_{\text{new}}, \Delta_{\text{new}}\}$  received from another processor. If  $f_{\text{new}} < f_{\text{best}}$ , replace  $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\} \leftarrow \{x_{\text{new}}, f_{\text{new}}, \Delta_{\text{new}}\}$ ,  $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$ .
1. Compute  $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}} d$ , and evaluate  $f_{\text{trial}} \leftarrow f(x_{\text{trial}})$ .
2. Set  $\{x_{\text{new}}, f_{\text{new}}, \Delta_{\text{new}}\} \leftarrow \{x_{\text{trial}}, f_{\text{trial}}, \Delta_{\text{trial}}\}$ .
3. If  $f_{\text{new}} < f_{\text{best}}$ , then  $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\} \leftarrow \{x_{\text{new}}, f_{\text{new}}, 2^{\ell} \Delta_{\text{new}}\}$ ,  $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$ , and *broadcast* the new best triplet  $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\}$  to all other processors. Else  $\Delta_{\text{trial}} \leftarrow \frac{1}{2} \Delta_{\text{trial}}$ .
4. If  $\Delta_{\text{trial}} > \text{tol}$ , goto Step 0. Else *broadcast* a directional convergence message for the pair  $\{x_{\text{best}}, f_{\text{best}}\}$ .
5. Wait until either (a) enough of processes have converged for this point or (b) a better point is received. In case (a), exit. In case (b), goto Step 0.

PLAY APPS MOVIE

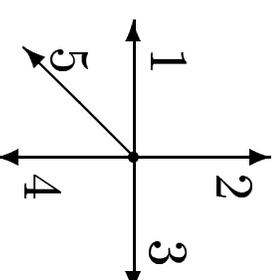
# APPS AGENT

- **New Best from Another Processor**  
If  $f_{\text{new}} < f_{\text{best}}$ , then  $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\} \leftarrow \{x_{\text{new}}, f_{\text{new}}, \Delta_{\text{new}}\}$ .
- **Return from function evaluation**  
If  $f_{\text{trial}} < f_{\text{best}}$ , then  
 $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\} \leftarrow \{x_{\text{trial}}, f_{\text{trial}}, 2^{\ell} \Delta_{\text{trial}}\}$ ,  $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$ ,  
**broadcast** the new best, and **spawn** a new function evaluation  
at  $x_{\text{trial}} = x_{\text{best}} + \Delta_{\text{trial}} d$ .  
Else if  $x_{\text{best}} \neq$  *trial point generator*, then **spawn a new function evaluation** using  $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$ .  
Else  $\Delta_{\text{trial}} = \frac{1}{2} \Delta_{\text{trial}}$ . If  $\Delta_{\text{trial}} < \textit{tol}$ , then **broadcast a local convergence message**, else **spawn a new function evaluation**.
- **Directional Convergence Message**  
Go through steps for new best to verify point. If incoming point is the same as best, then update its *convergence table* and check for *convergence*.

# CHECKING CONVERGENCE

## 2-D Search Pattern

We have *convergence* when enough processes have locally converged to contain a positive basis.



To verify that a given set  $\mathcal{V} = \{v_1, v_2, \dots, v_m\}$  is a positive spanning set, solve  $n + 1$  nonnegative least squares problems of the form

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

for  $b = \{e_1, \dots, e_n, -e\}$ . All have residual zero iff  $V$  is a positive spanning set.

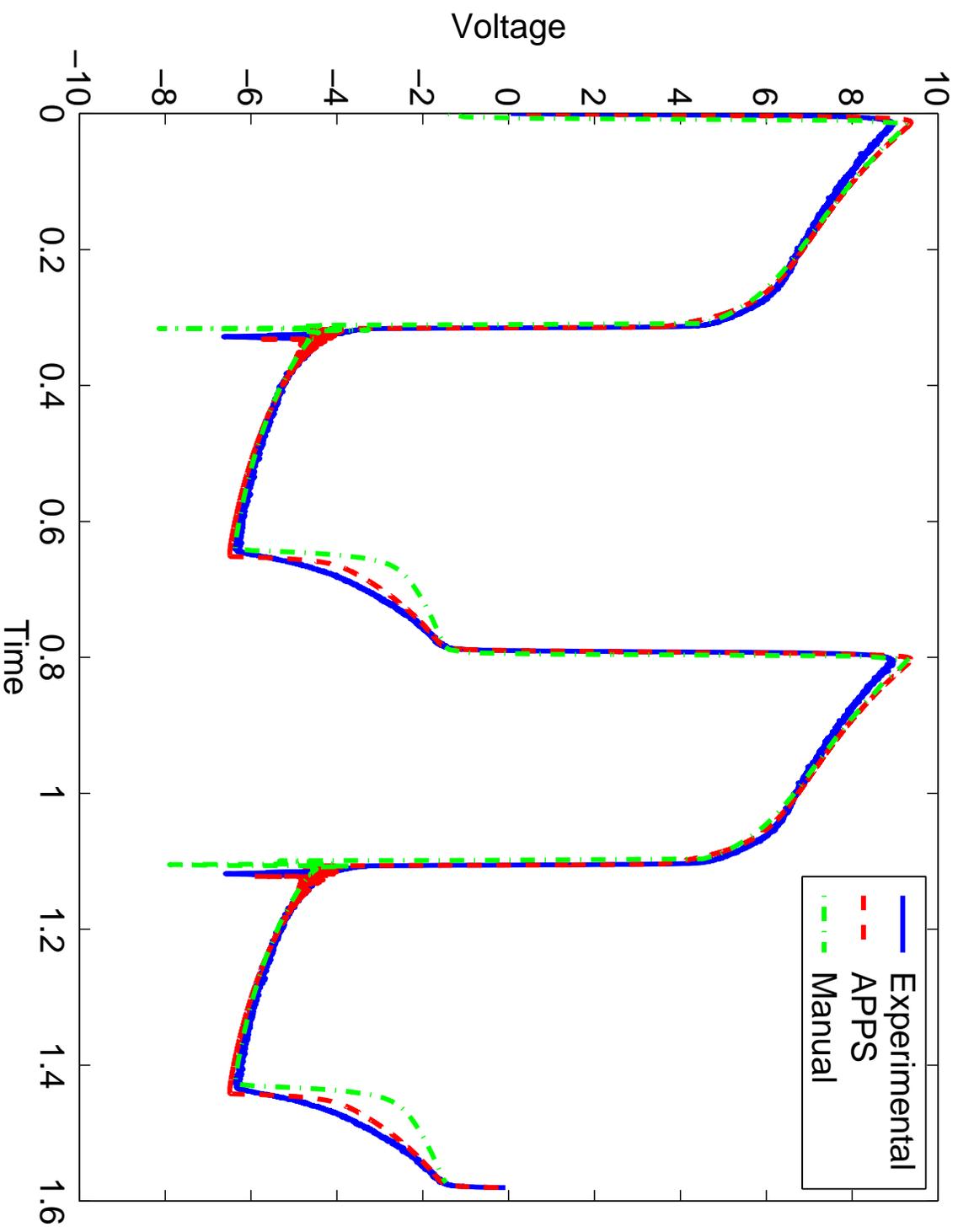
# CIRCUIT SIMULATION PROBLEM NUMERICAL RESULTS

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

- 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

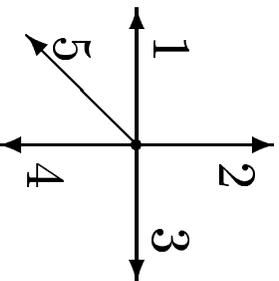
# CIRCUIT SIMULATION PROBLEM SOLUTION



# FAULT TOLERANCE IN APPS

## 2-D Search

Pattern



- We can afford to lose processes as long as we maintain a positive basis.
- Processes can be restarted when we no longer have a positive basis.
- Progression towards a solution continues as long as one or more APPS processes is alive.
- Completion does *not* depend on the survival of any particular APPS agent.

# PLAY APPS WITH FAULTS MOVIE

## CIRCUIT SIMULATION WITH FAULTS

The “fault” versions have a failure in the function evaluation or agent every 30 seconds.

Method	Initial Procs	$f(x^*)$	Total Time
APPS	34	26.2	1330.55
APPS	34-faults	27.8	1618.46
PPS	34	28.8	1712.25

## WHAT I HAVEN'T TALKED ABOUT...

- The Gory Details
- Bound Constraints
- Convergence Theory
- Implementation
- Etc...

“The most valuable acquisitions in a scientific or technical education are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of these tools, and computer science as a third.”

— George Forsythe, 1968,

What to do till the computer scientist comes home,  
*American Mathematical Monthly.*

## THANKS TO...

“It is amazing what you can accomplish if you do not care who gets the credit.” — Harry Truman

- Virginia Torczon, Wm & Mary
- Patty Hough, Sandia
- Alton Patrick, Summer Intern
- Sarah Brown, Summer Intern
- Ken Marx, Sandia Engineer

## FOR MORE INFORMATION...

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

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

[tgkolda@sandia.gov](mailto:tgkolda@sandia.gov)