Fault Tolerance in APPSPACK

Asynchronous Parallel Pattern Search for Derivative-Free Optimization

TAMARA G. KOLDA
Sandia National Labs

Using Pattern Search

Our goal is to solve the unconstrained optimization problem

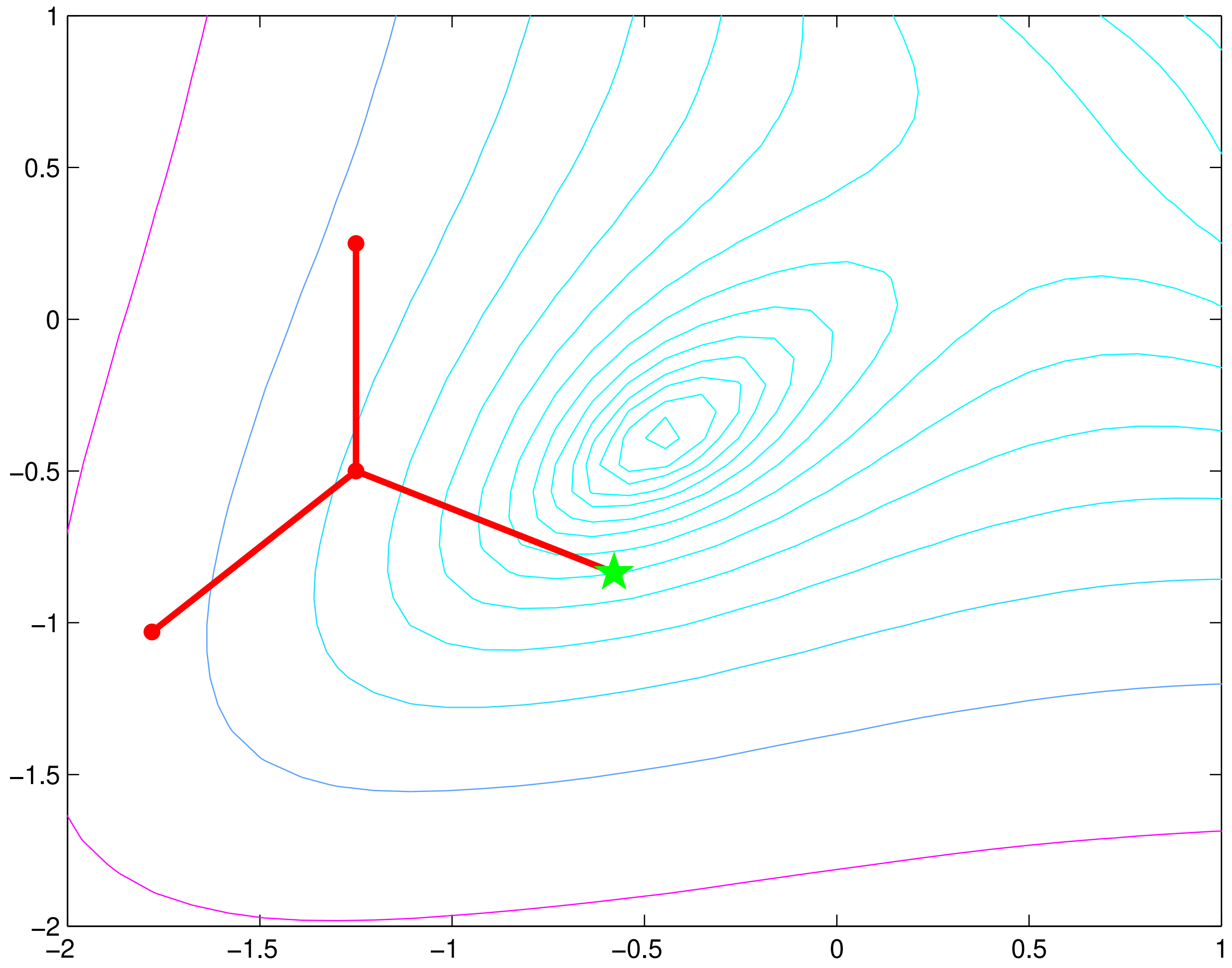
$$\min f(x) \quad x \in \Re^n$$

without derivative information. Pattern search is a popular *derivative-free* method which is most useful when...

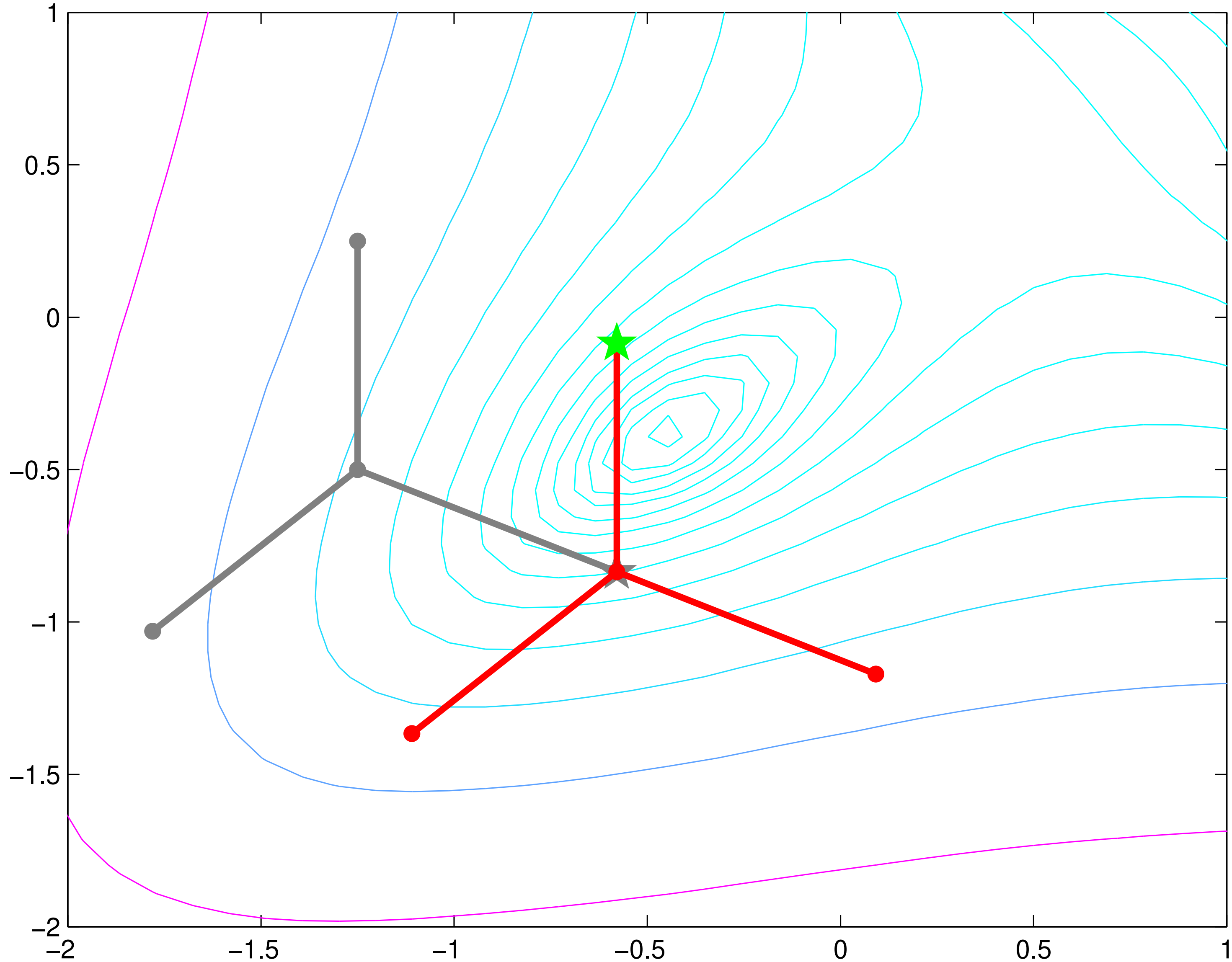
- The function is expensive to calculate.
- The gradient cannot be calculated.
- Numerical approximation of the gradient is too slow, or the function values are too noisy to yield reliable gradient approximations.

We are motivated by engineering optimization applications.

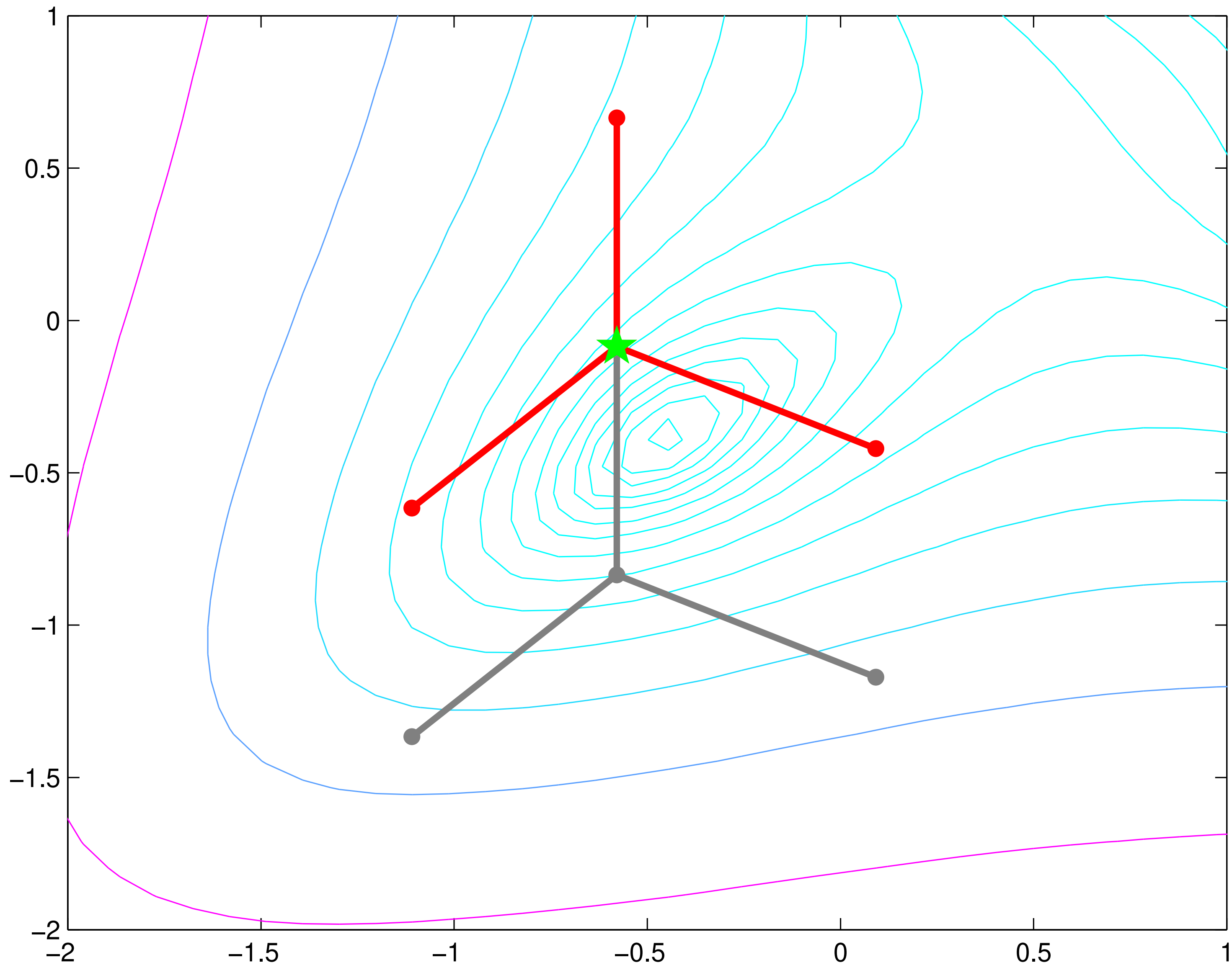
Parallel Pattern Search



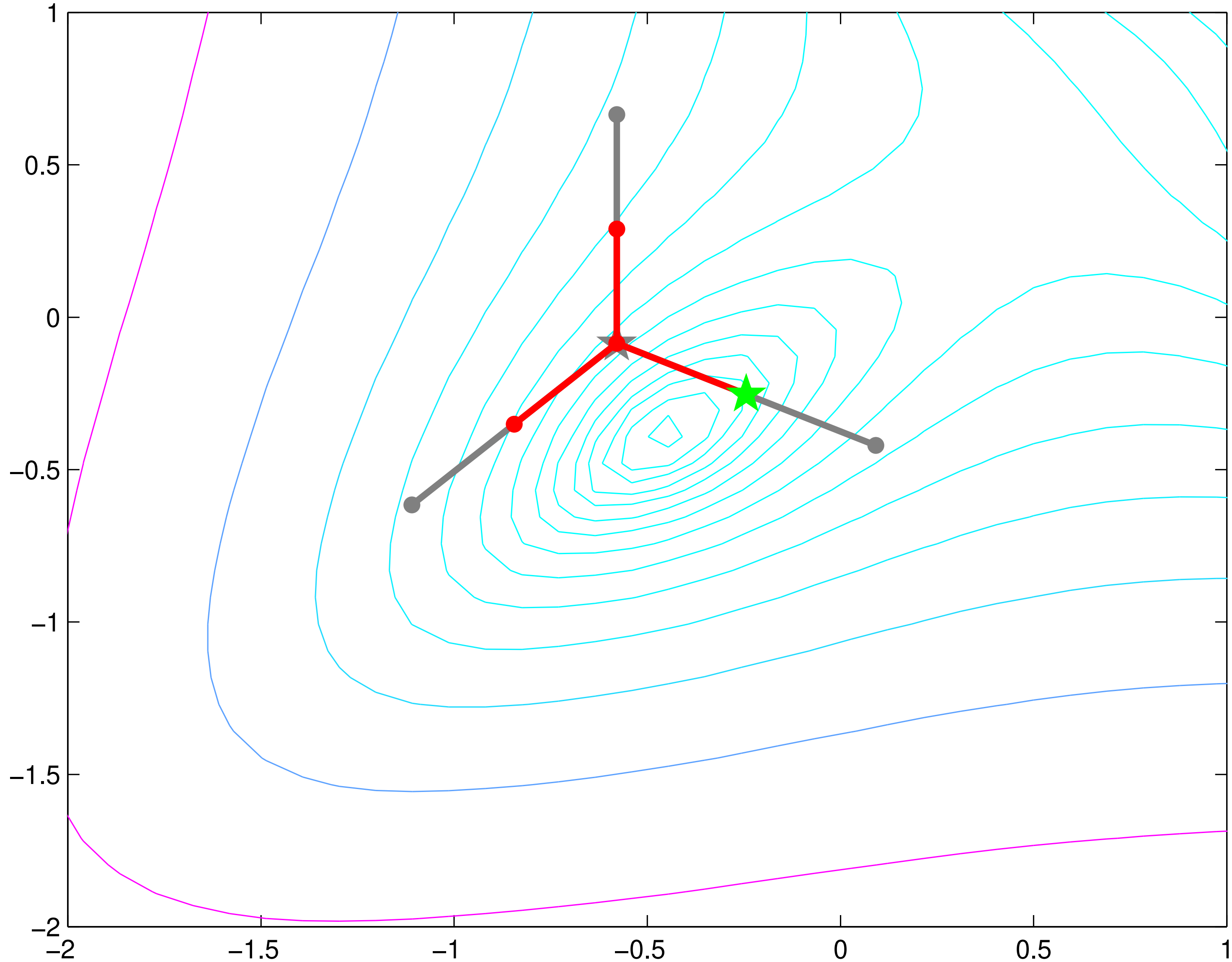
Parallel Pattern Search



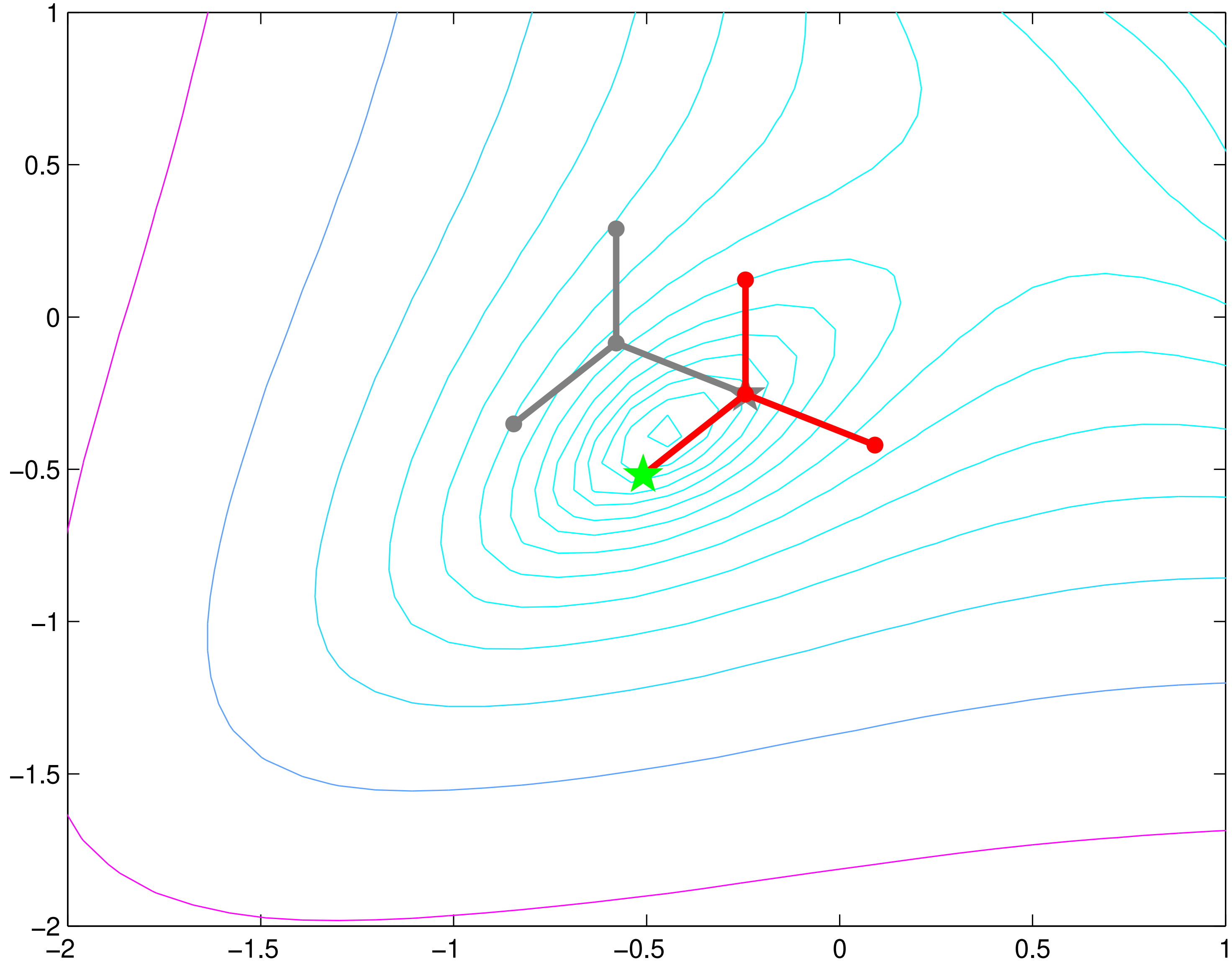
Parallel Pattern Search



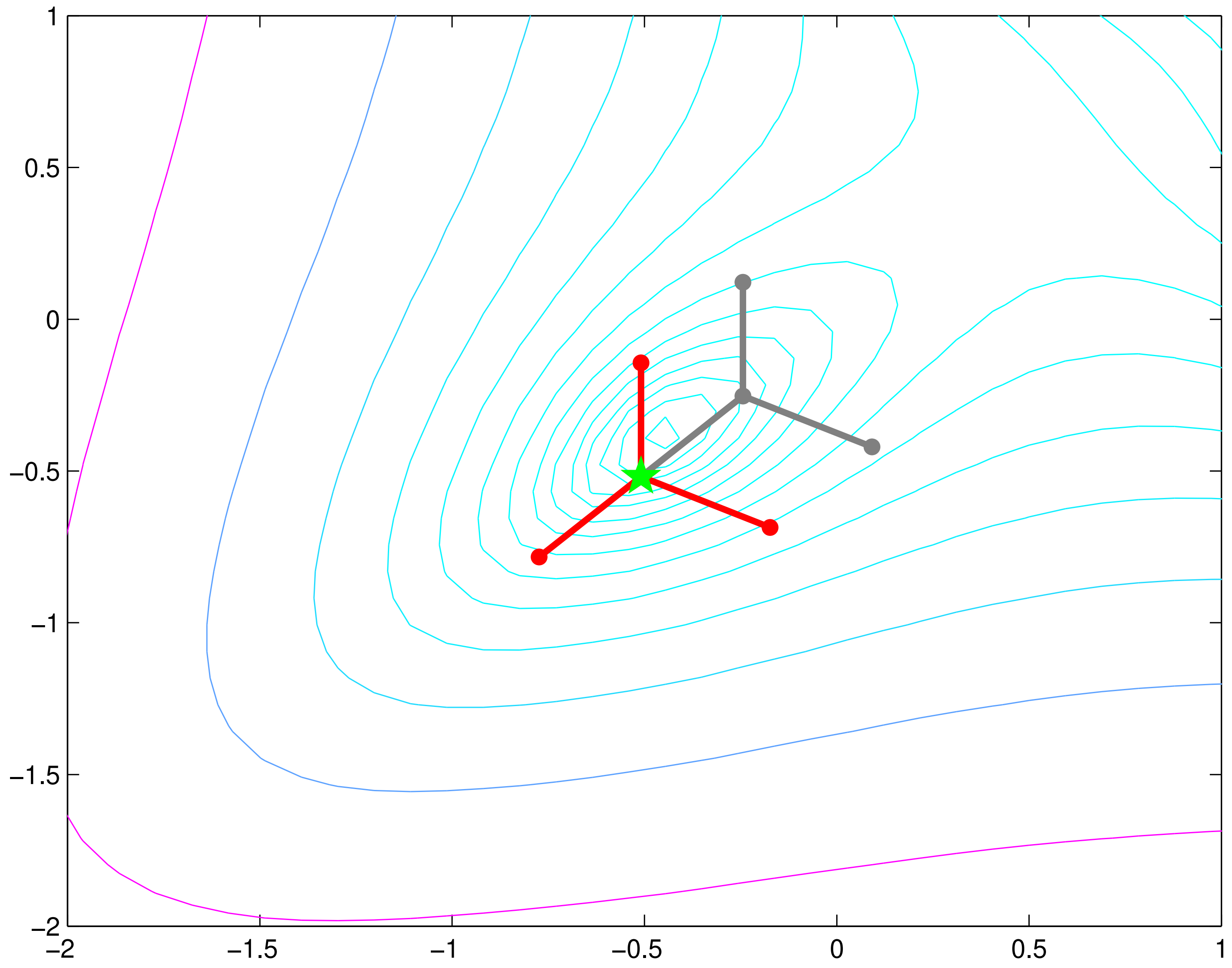
Parallel Pattern Search



Parallel Pattern Search



Parallel Pattern Search



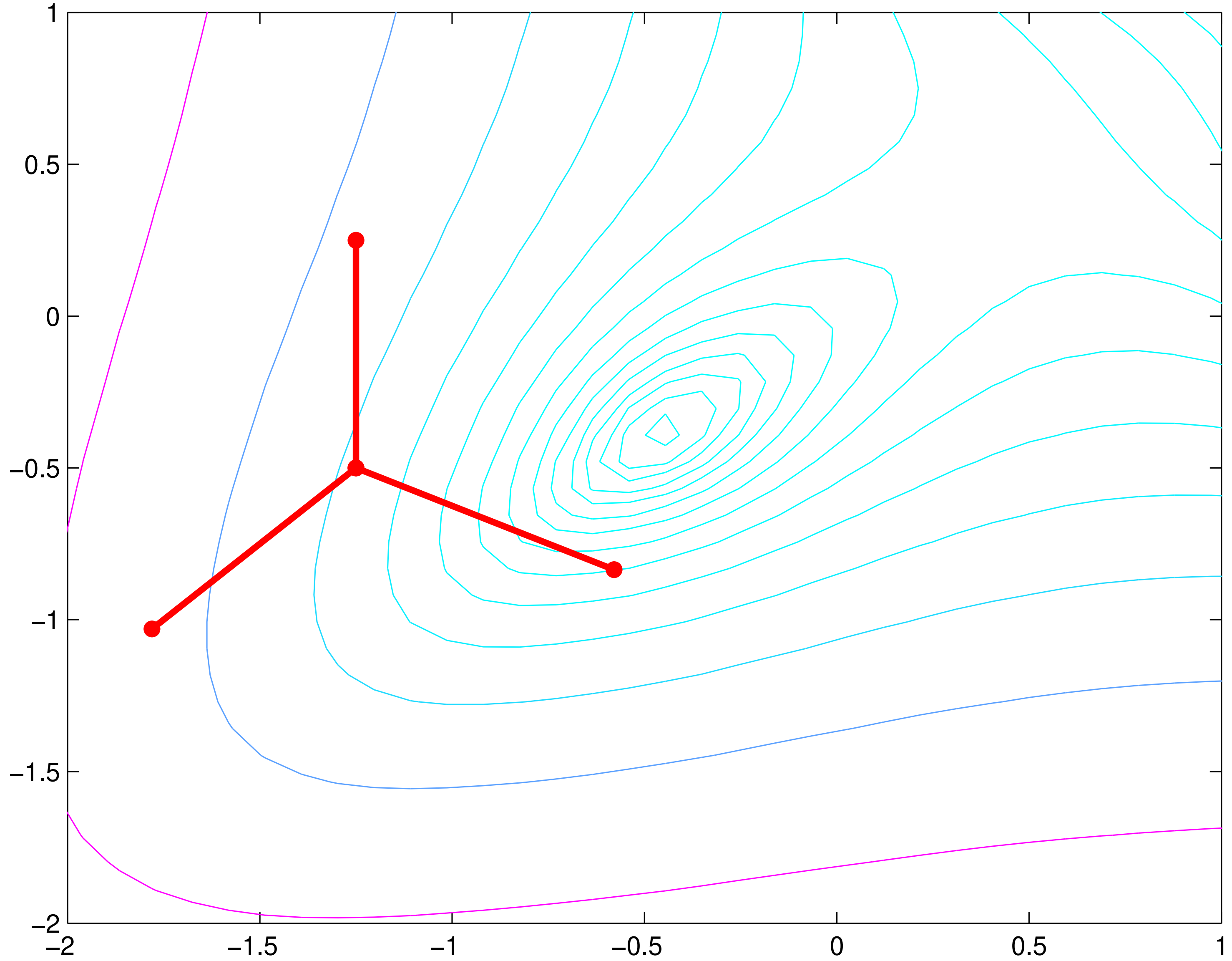
Asynchronous PPS Reduces Idle Time

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

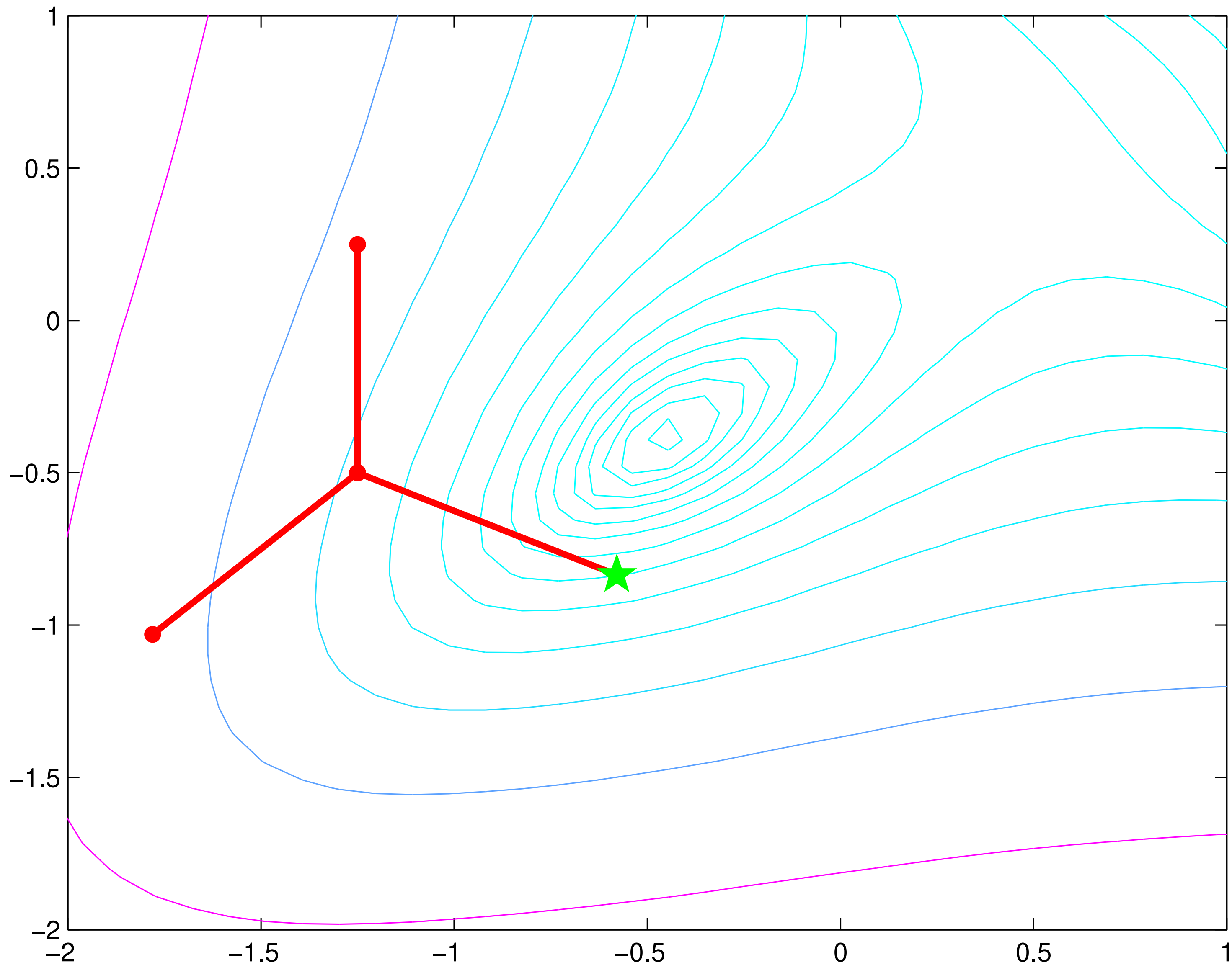
APPS/APPSPACK Collaborators

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

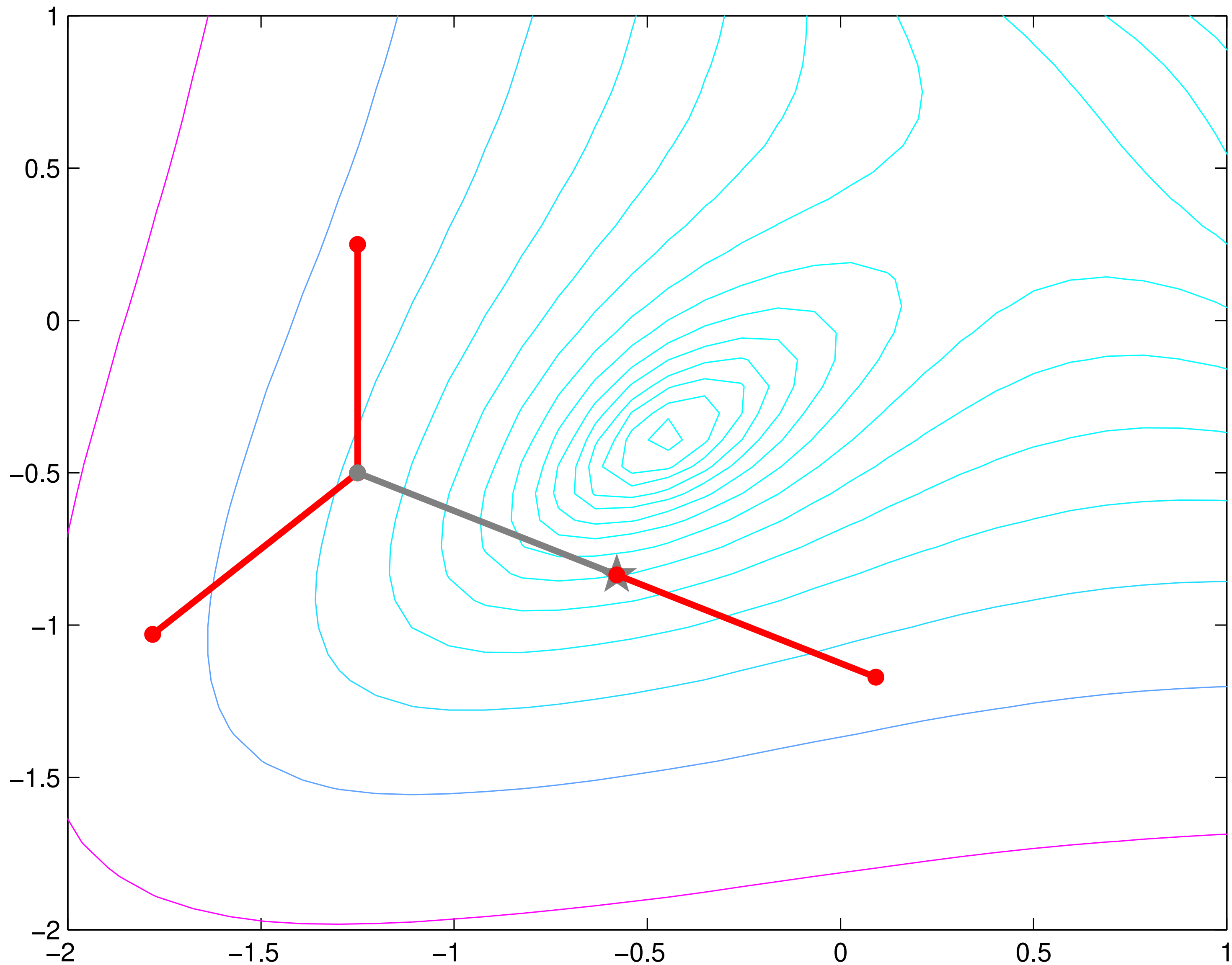
Asynchronous Parallel Pattern Search



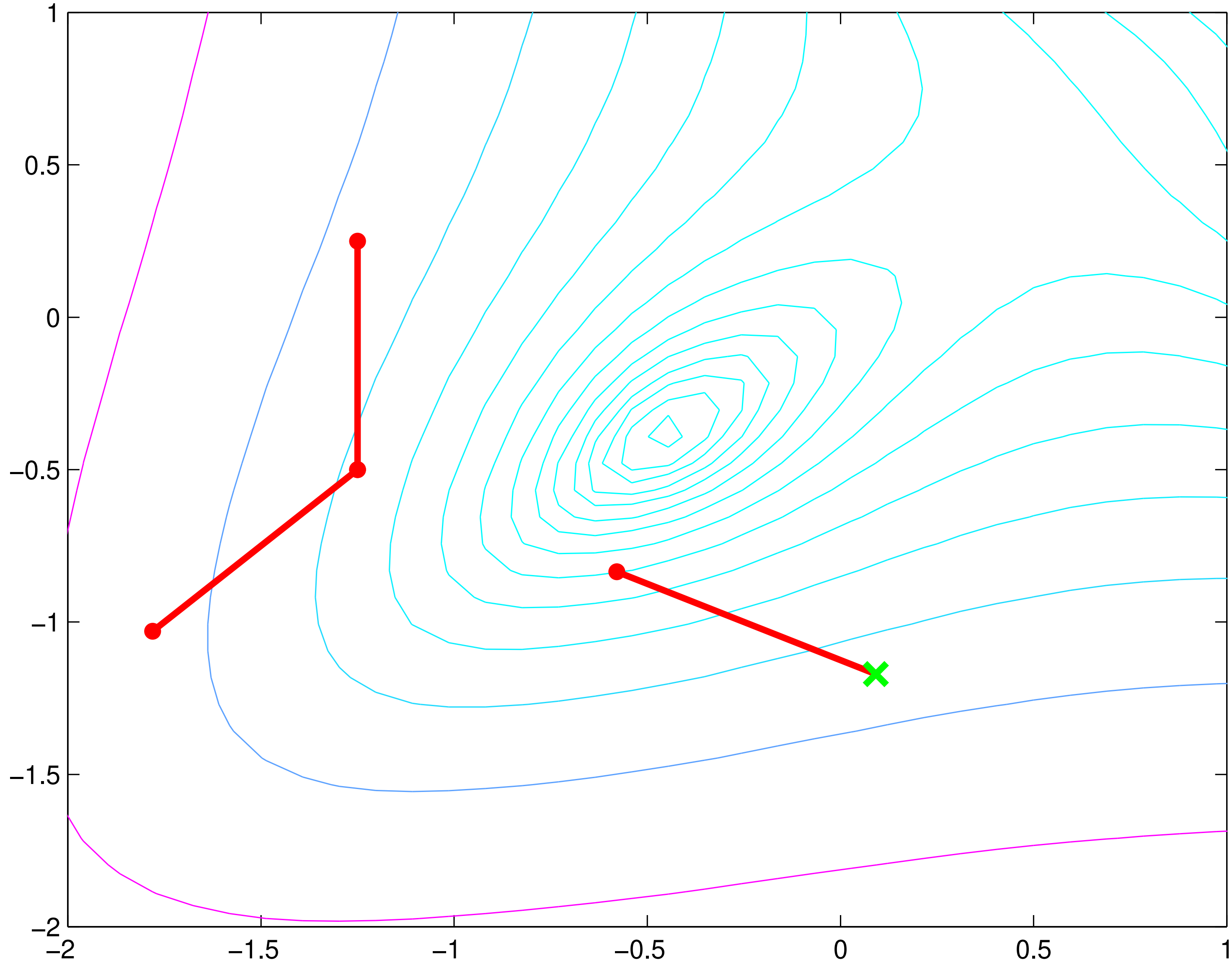
Asynchronous Parallel Pattern Search



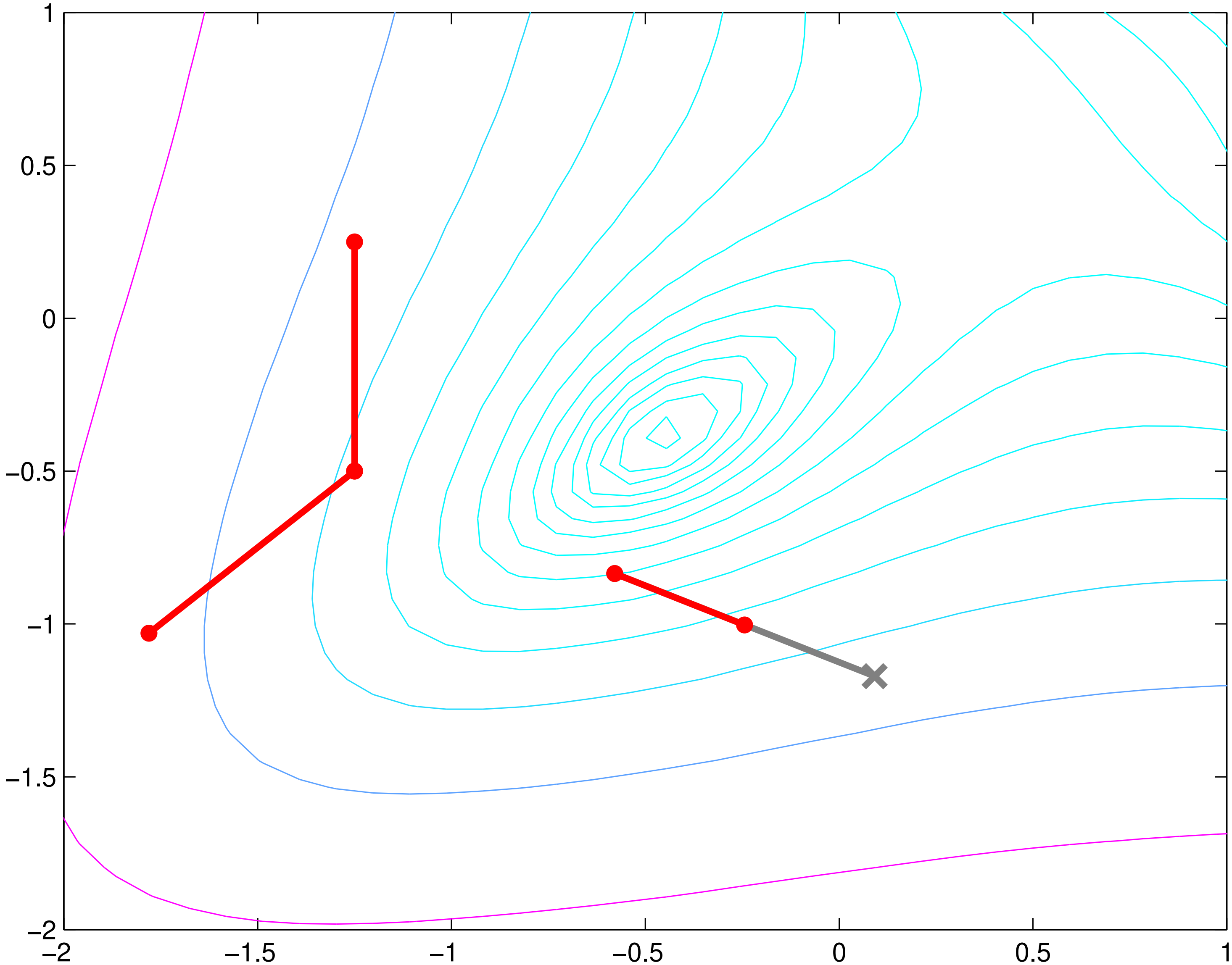
Asynchronous Parallel Pattern Search



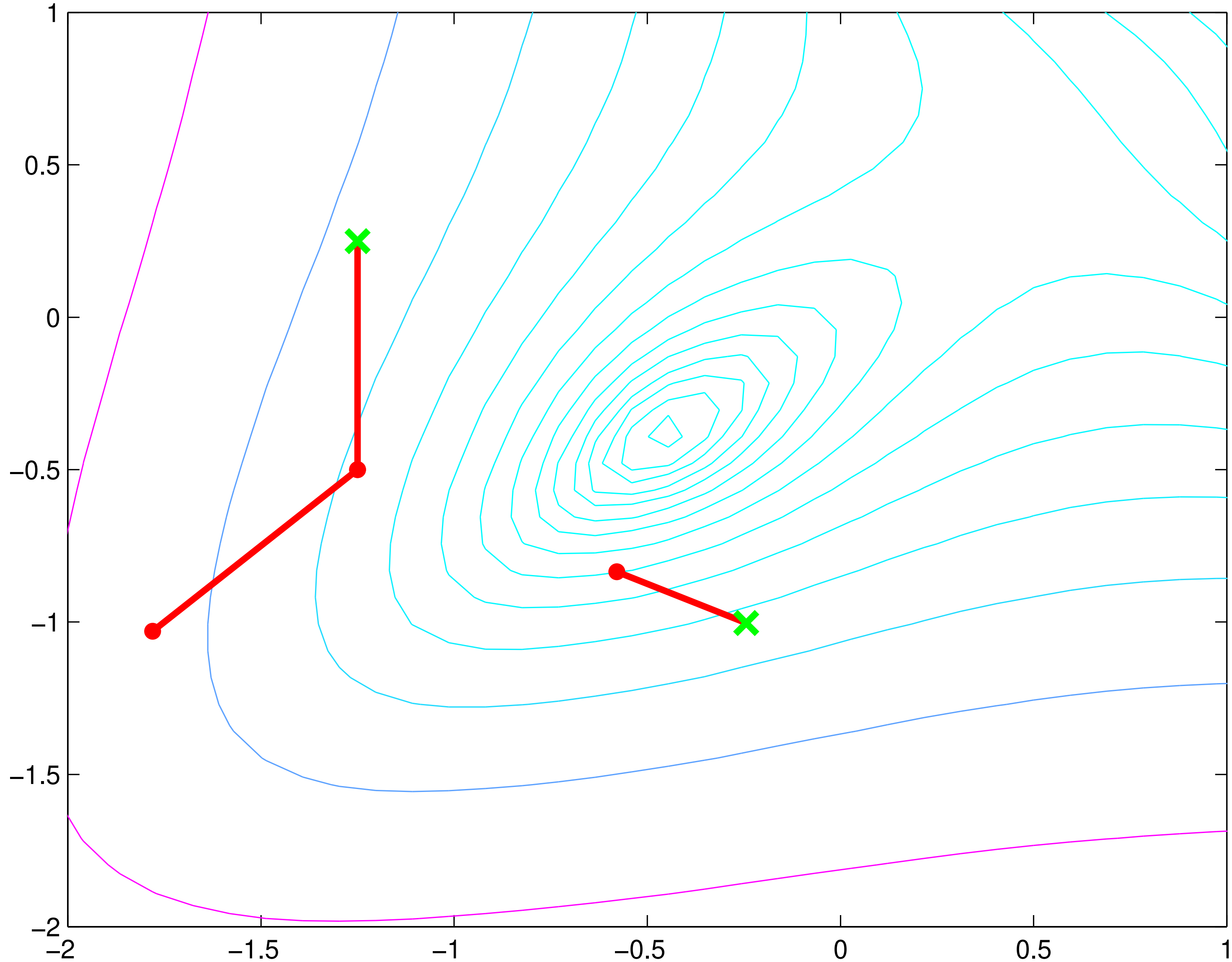
Asynchronous Parallel Pattern Search



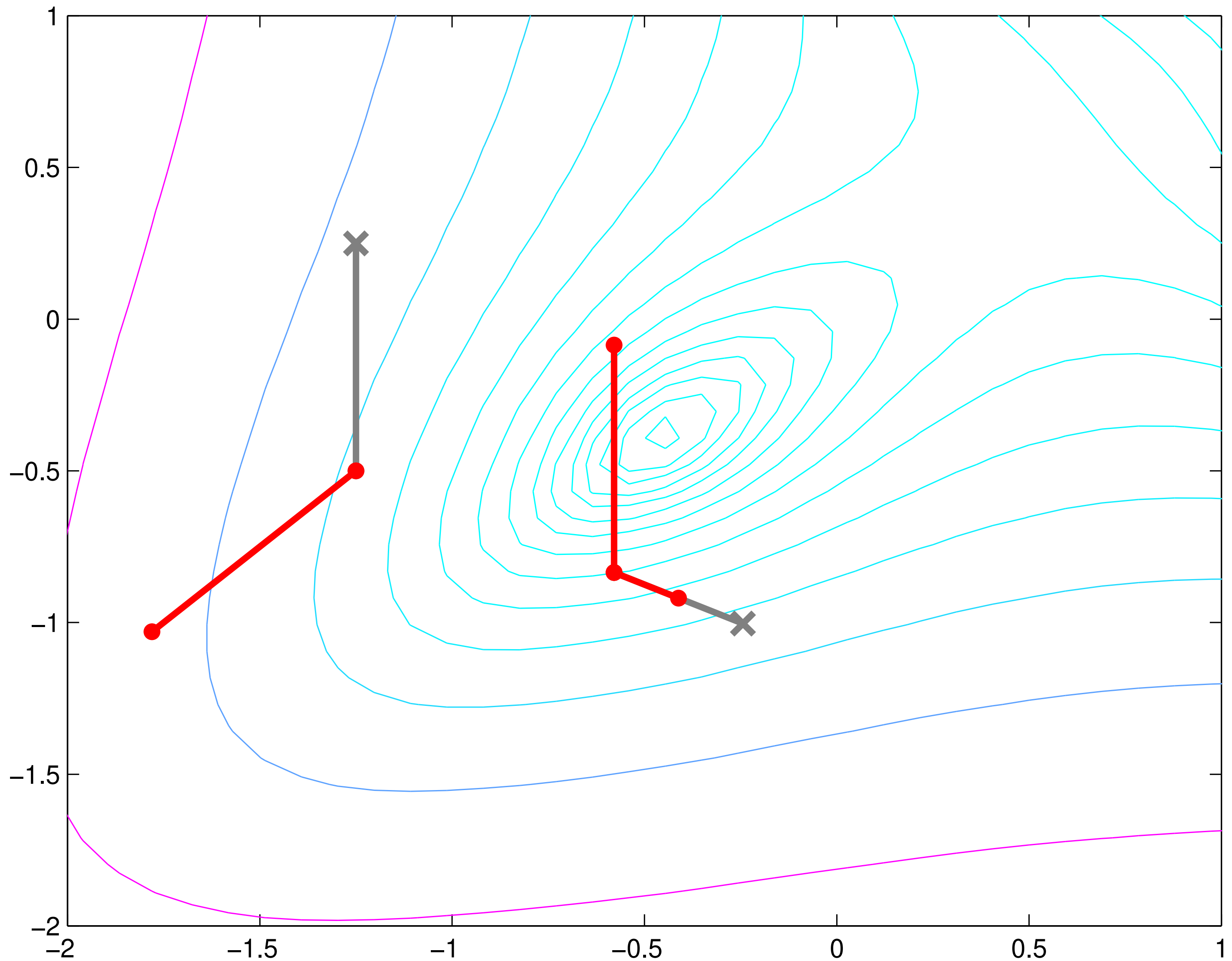
Asynchronous Parallel Pattern Search



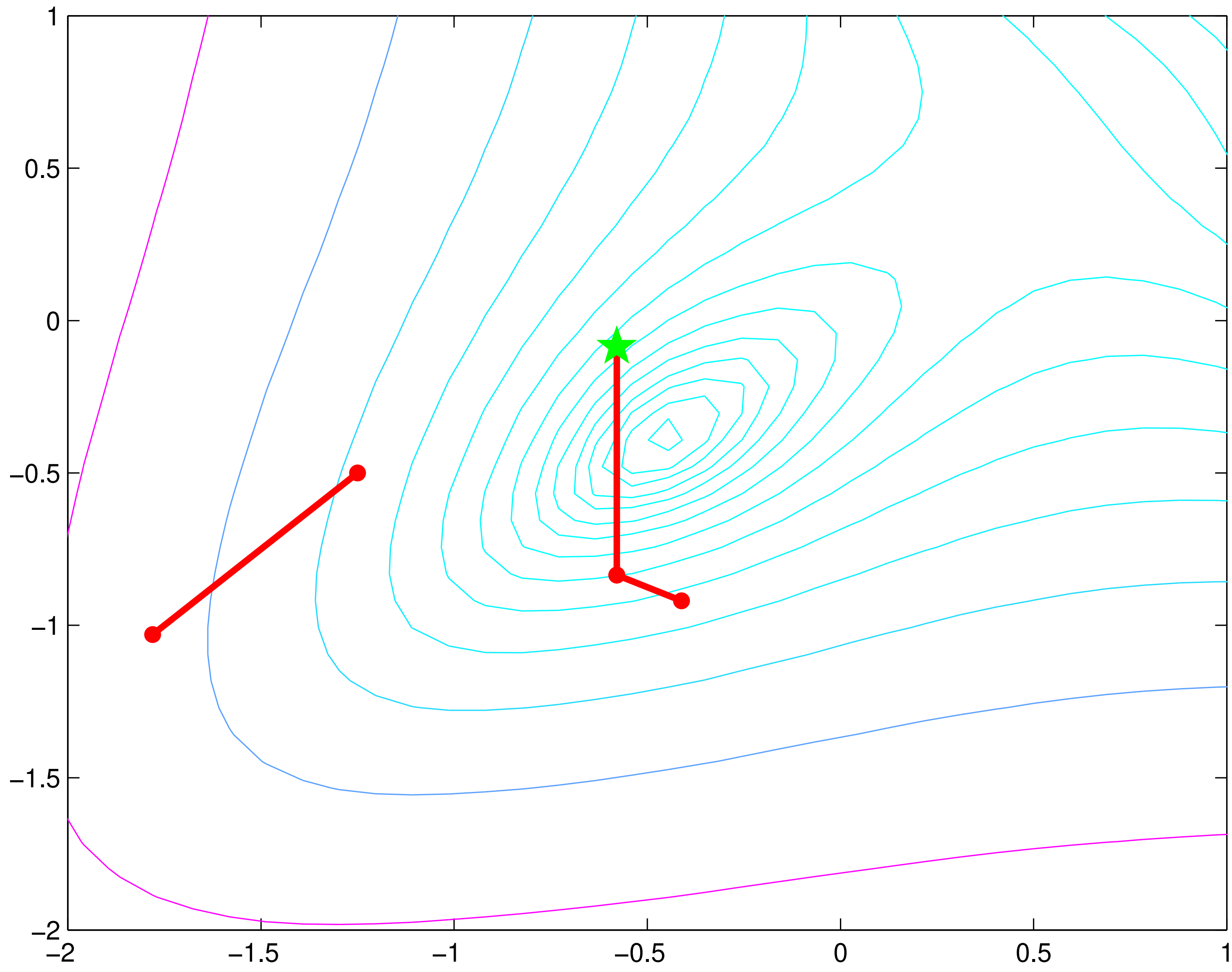
Asynchronous Parallel Pattern Search



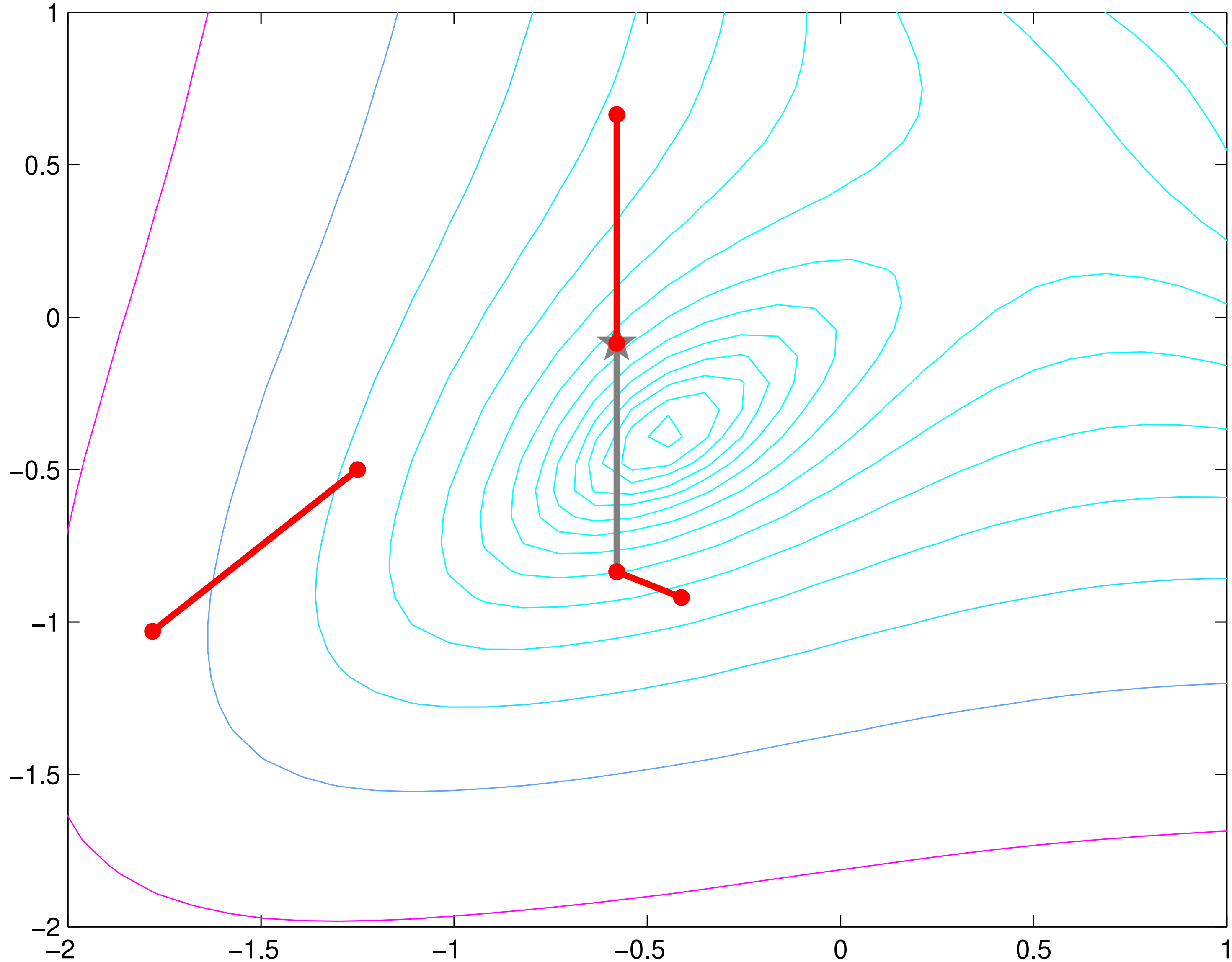
Asynchronous Parallel Pattern Search



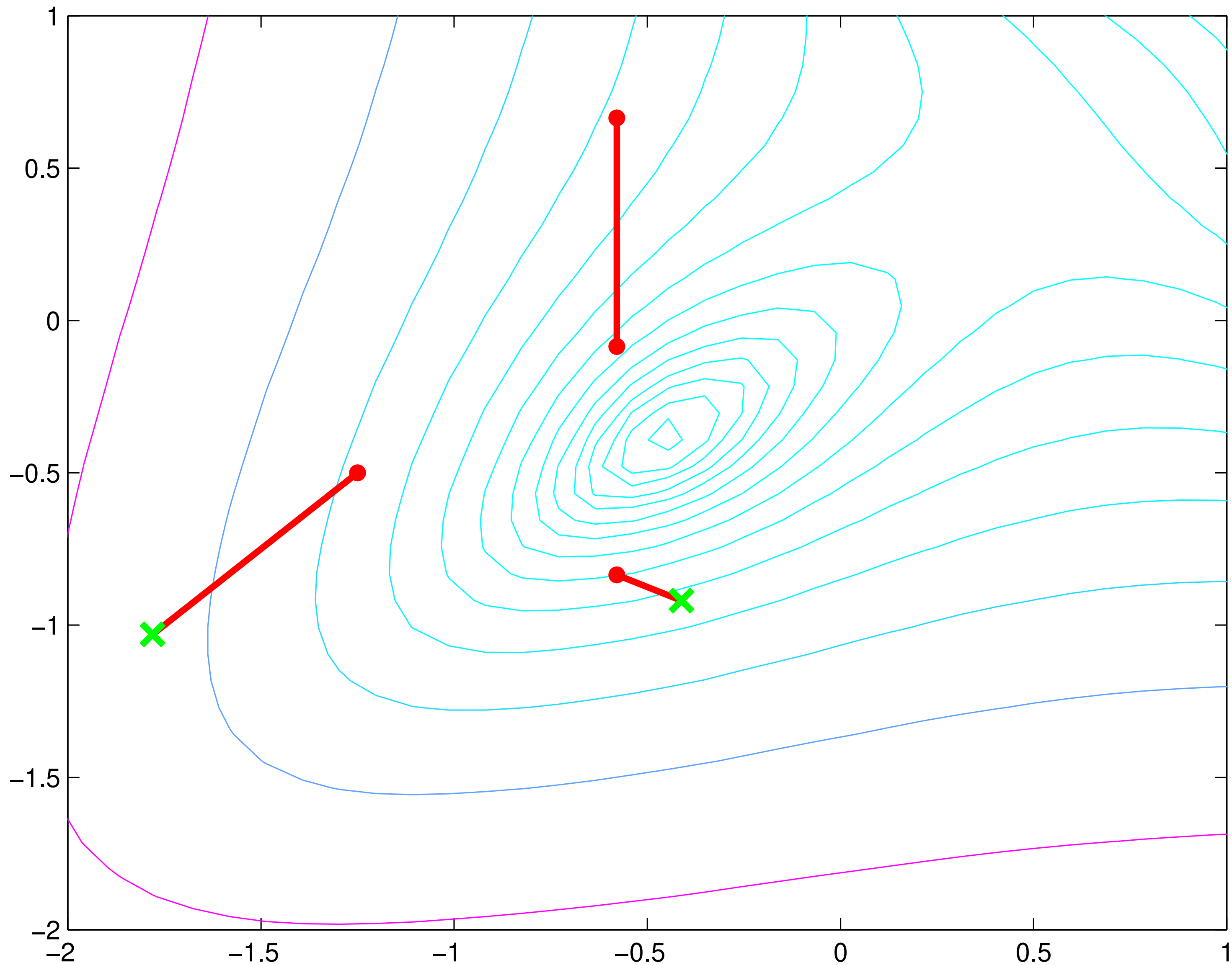
Asynchronous Parallel Pattern Search



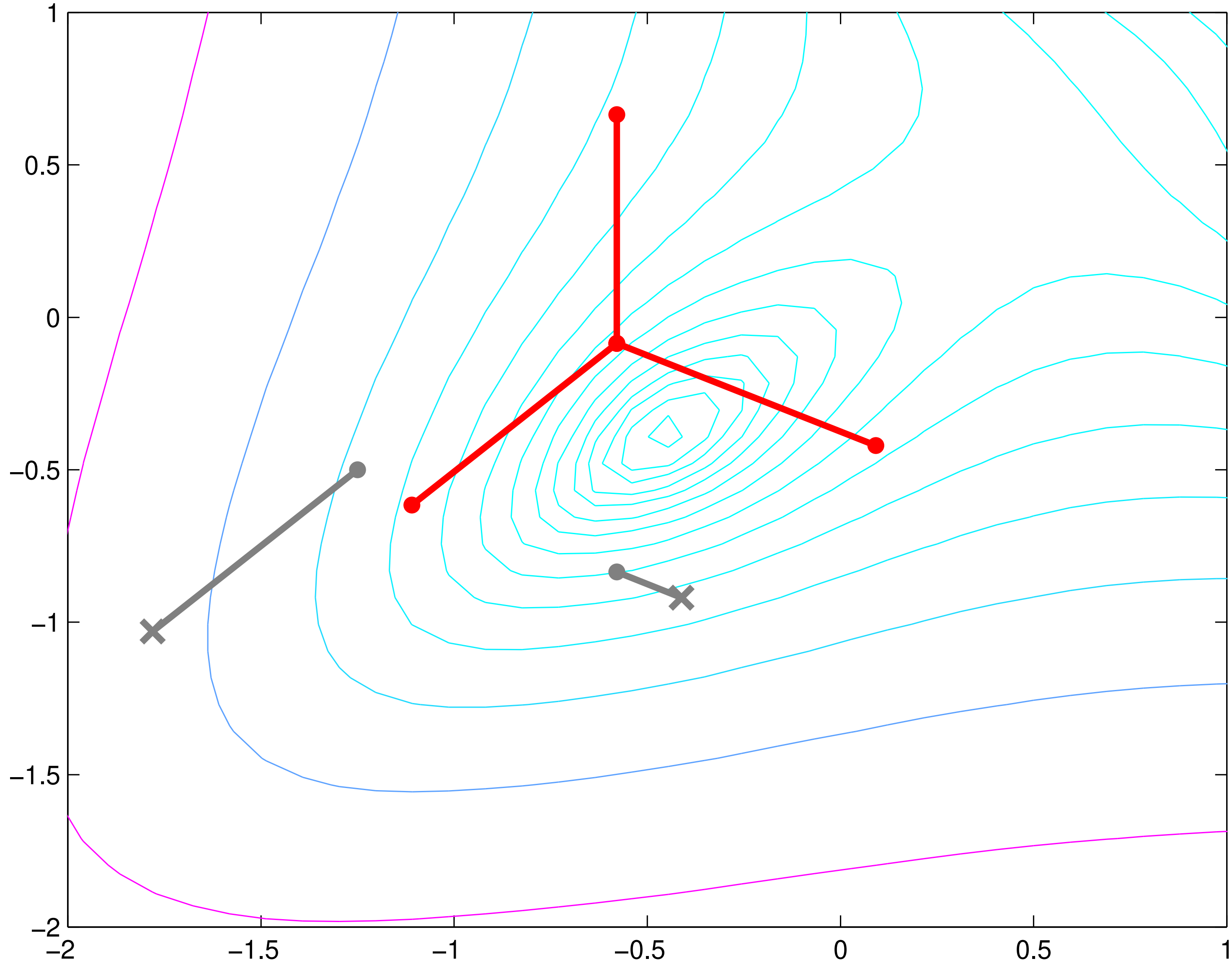
Asynchronous Parallel Pattern Search



Asynchronous Parallel Pattern Search



Asynchronous Parallel Pattern Search



Engineering Example: Determining the Characteristics of a Circuit

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

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

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

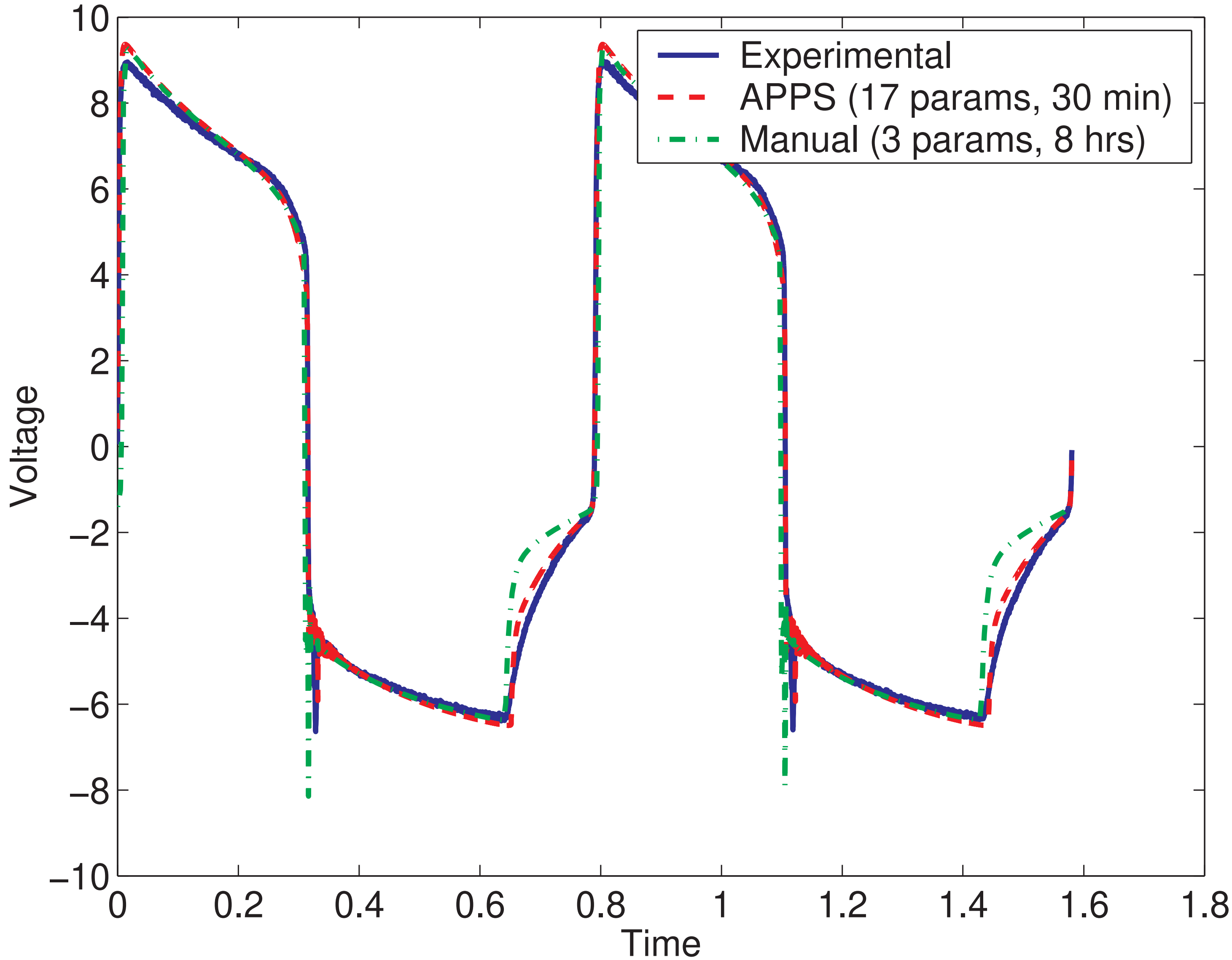
APPS Dramatically Reduces Idle Time for the Circuit Simulation Problem

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

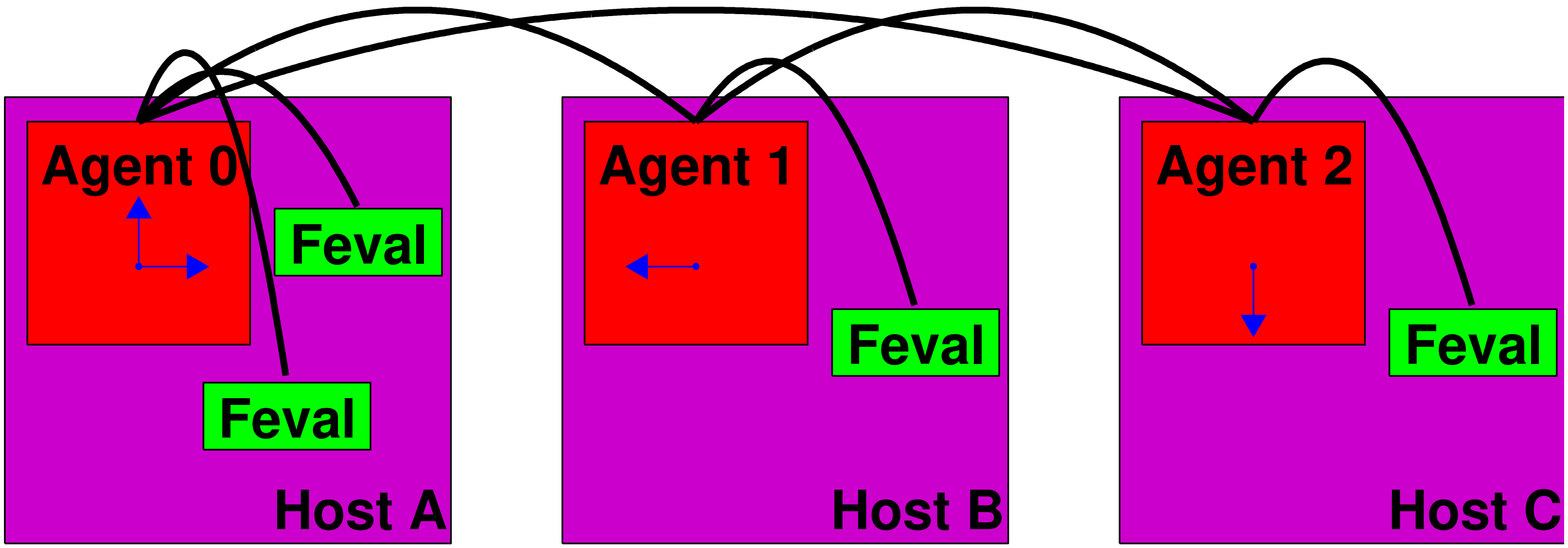
- Search directions are $\pm$ Unit Vectors (34) plus Random
- The variables are bound constrained; i.e., $l_i \leq x_i \leq u_i$

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

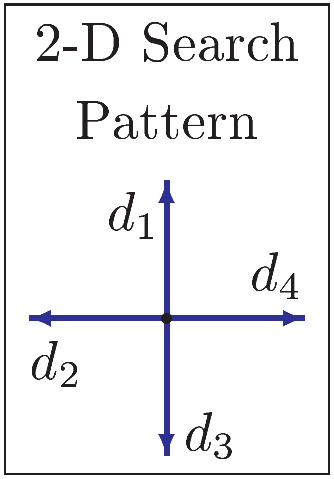
APPS Solution is Faster and More Accurate than Hand-Tuning



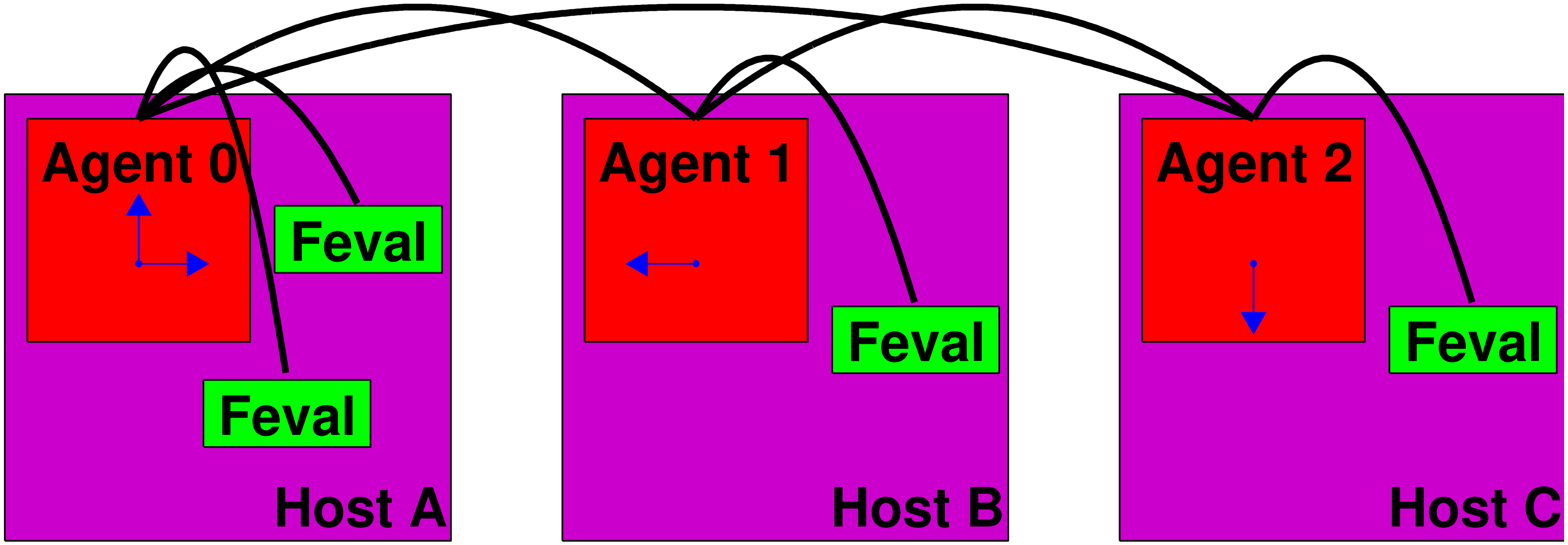
APPS has Agent and Feval Processes



- Each host has an Agent, and each agent owns a subset of the Search Pattern.
- Each agent dynamically spawns Function Evaluations, and the maximum number of function evaluations per agent is equal to the number of search directions it owns.



APPS Distributes Control via Agents



- Each Agent has a unique integer id, starting with 0 and numbered consecutively.
- Agents communicate with each other and with any function evaluations that they own.
- The agents contain the logic of the search, but the bulk of the computation work is done by the function evaluations.

Agent Information

Agent	Alive?	HostId	TaskId	Speed	Hostname
0	true	0x40000	0x4006f	1	su1cn1
1	true	0x80000	0x8006e	1	su1cn2
2	true	0xc0000	0xc0068	1	su1cn3
Job Assignment = [0 1 2 0]					

- Agent numbers are never re-used.
- The HostId and TaskId are generated by PVM and are never reused.
- The speeds are used to balance the job assignments.
- The job assignments map search directions (4 in this example) to agents.

Two Keys to Fault Tolerance

Notify

```
GCI::notify(MsgTag::F_EXIT, ftid);
GCI::notify(MsgTag::AGENT_EXIT, agentid);
```

PVM automatically sends a message with the appropriate flag when the specified process exits, either successfully or unsuccessfully. The message body contains the process id.

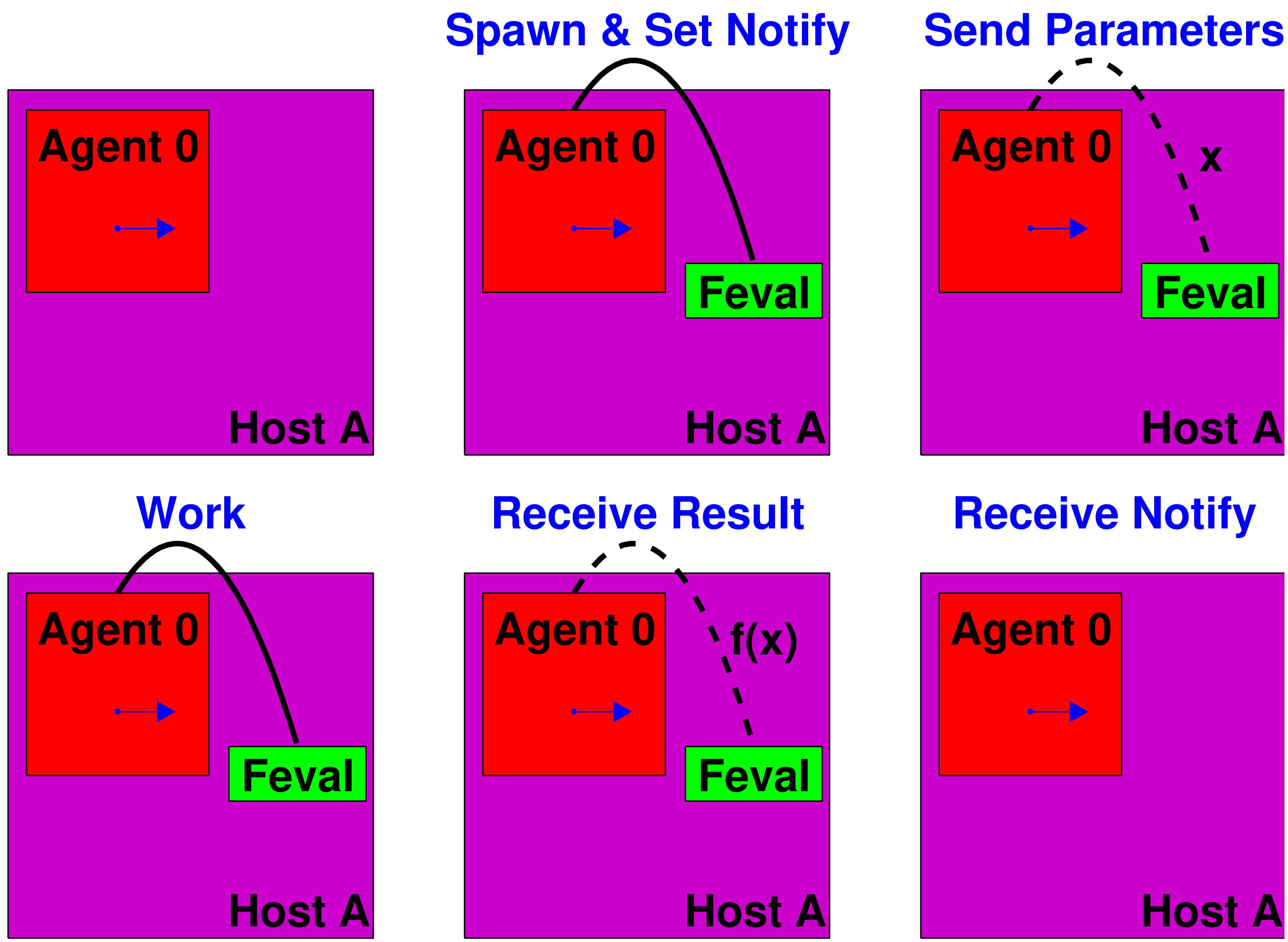
```
GCI::notify(MsgTag::AGENT_NEW_HOST);
```

PVM automatically sends a message with the appropriate flag if a new host is added.

GUID (Globally Unique Id)

A two-part identification that is used to differentiate and prioritize information. The first part of the GUID is the agent id, and the second part is produced by a counter which increments each time a new GUID is produced by that agent.

FEVAL Fault Tolerance is Easy



A fault can happen in any step, in which case we set $f(x) = +\infty$.

Agent Failure

1. All the other agents get a notify message containing the TaskId of the failed agent, and they each update their "Agent Information" to show that the agent is dead.

2. Each agent deteremines whether or not it is the *temporary master*, which is defined to be the agent with the lowest number.

If Agent 0 fails, then Agent 1 is the temporary master.

Agent	Alive?	HostId	TaskId	Speed	Hostname
0	false	0x40000	0x4006f	1	su1cn1
1	true	0x80000	0x8006e	1	su1cn2
2	true	0xc0000	0xc0068	1	su1cn3

Job Assignment = [0 1 2 0]

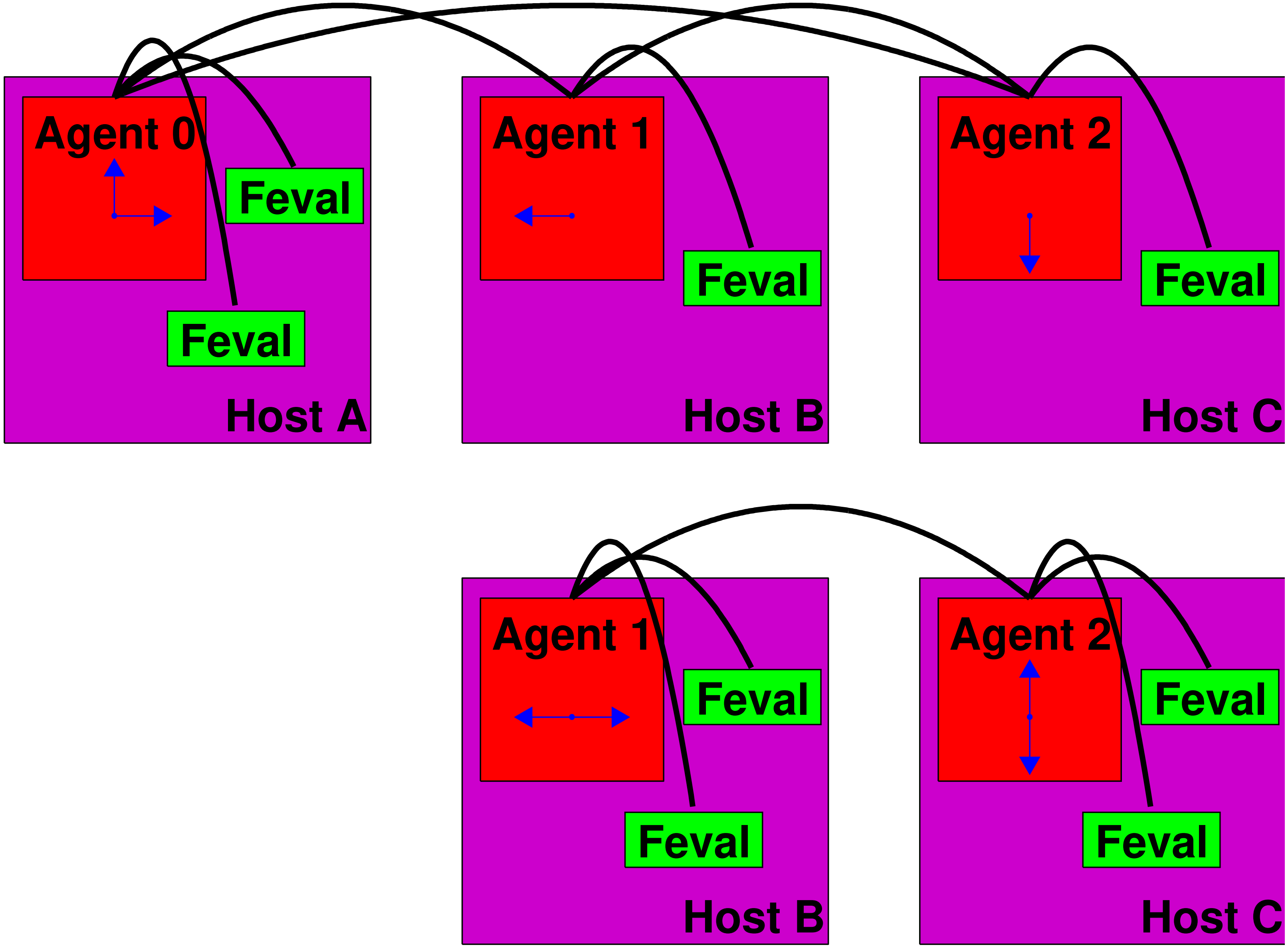
We see that the job assignment must be updated...

Temporary Agent Responsibilities

1. Check for convergence.
2. Re-verify which of the agents/hosts are alive.
3. Reset search direction assignments, if necessary.
4. Generate a new GUID for the agent information.
5. Send an update message to all other agents containing the updated agent information.

Agent	Alive?	HostId	TaskId	Speed	Hostname
0	false	0x40000	0	1	su1cn1
1	true	0x80000	0x8006e	1	su1cn2
2	true	0xc0000	0xc0068	1	su1cn3
Job Assignment = [1 1 2 2]					

Resetting Searches After Update



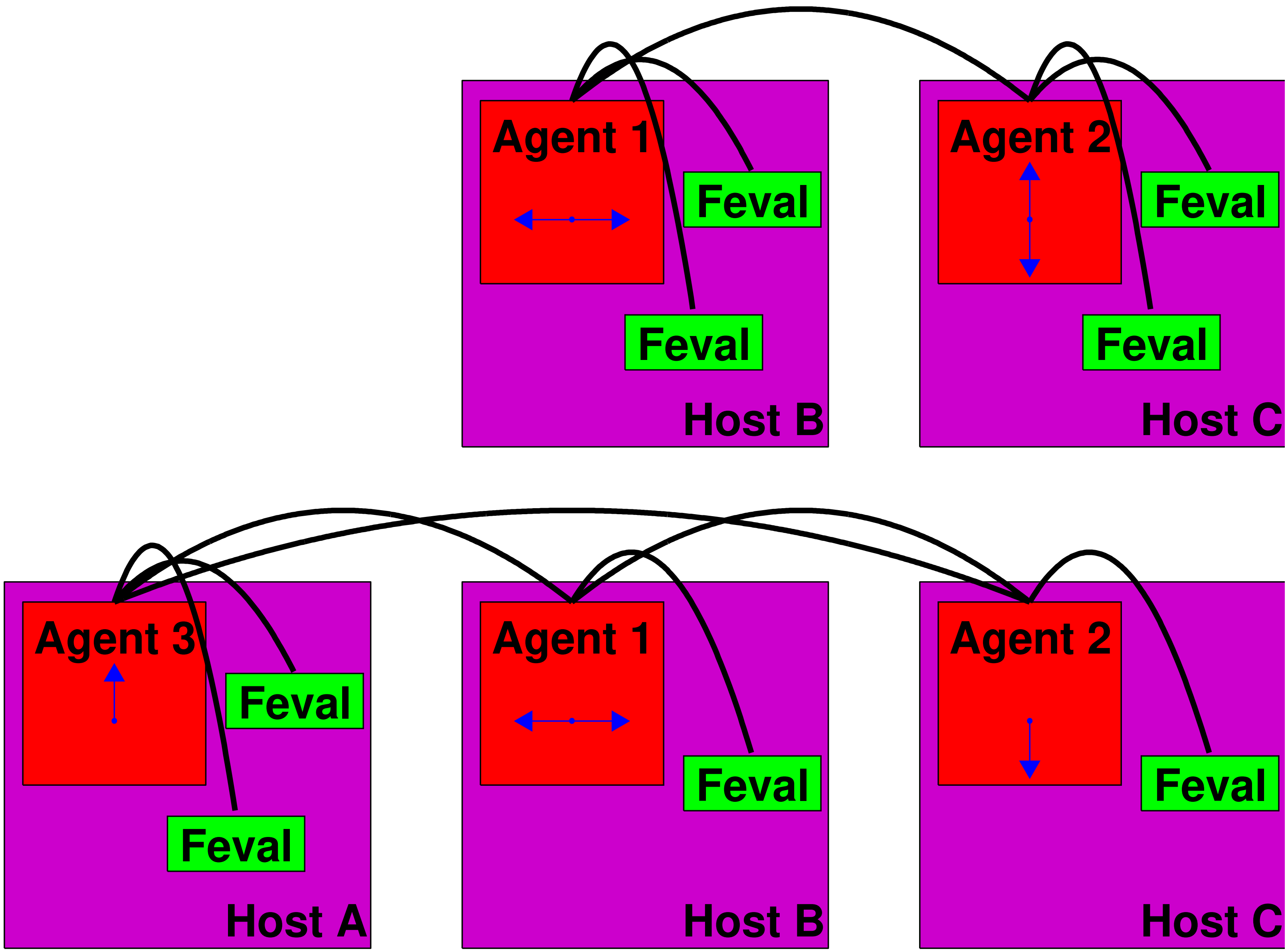
New Host

Every agent gets a notify message. The temporary master must...

1. Re-verify which of the old agents/hosts are alive.
2. Spawn agents on new hosts.
3. Reset search direction assignments, if necessary.
4. Generate a GUID for the agent information.
5. Send initialization message to new agents (including "best" point and search parameters)
6. Send an update message to old agents.

Agent	Alive?	HostId	TaskId	Speed	Hostname
0	false	0x40000	0	1	su1cn1
1	true	0x80000	0x8006e	1	su1cn2
2	true	0xc0000	0xc0068	1	su1cn3
3	true	0x100000	0x100004	1	su1cn1
Job Assignment = [1 1 2 3]					

Resetting Searches After Update



Sorting Agent Information

Concern: What if multiple agents fail in succession and we receive multiple agent information updates. How do we know which one to use?

Solution: The GUIDs and Temporary Master concepts allow us to sort out the information.

- The first part of the GUID is the agent number. Since the lowest numbered agent that is alive is the temporary master, we know that a higher first part of the GUID indicates a more recent message.
- The second part of the GUID is produced by a counter, so we know that the higher the counter, the more recent the information.

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.

Where We Are

- Fault tolerance for function evaluations and agents.
- No dependence on a “master” agent.
- Automatic redistribution of jobs.

What I Didn’t Mention

- Function Value Cache Process
- Surrogate Model Process

What’s Missing?

- Independence from PVM master daemon.
- Fault tolerance in MPI version.
- Disk check-pointing and restart.
- Method to detect that a process is “slow”.

To Obtain APPSPACK...

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

For More Information...

Tammy Kolda

tgkolda@sandia.gov

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

925-294-4769