Performance Studies of a Parallel Global Search Algorithm on System X

J. He, A. Verstak, and L. T. Watson
Virginia Polytechnic Institute and State University

M. Sosonkina
Ames Laboratory and Iowa State University

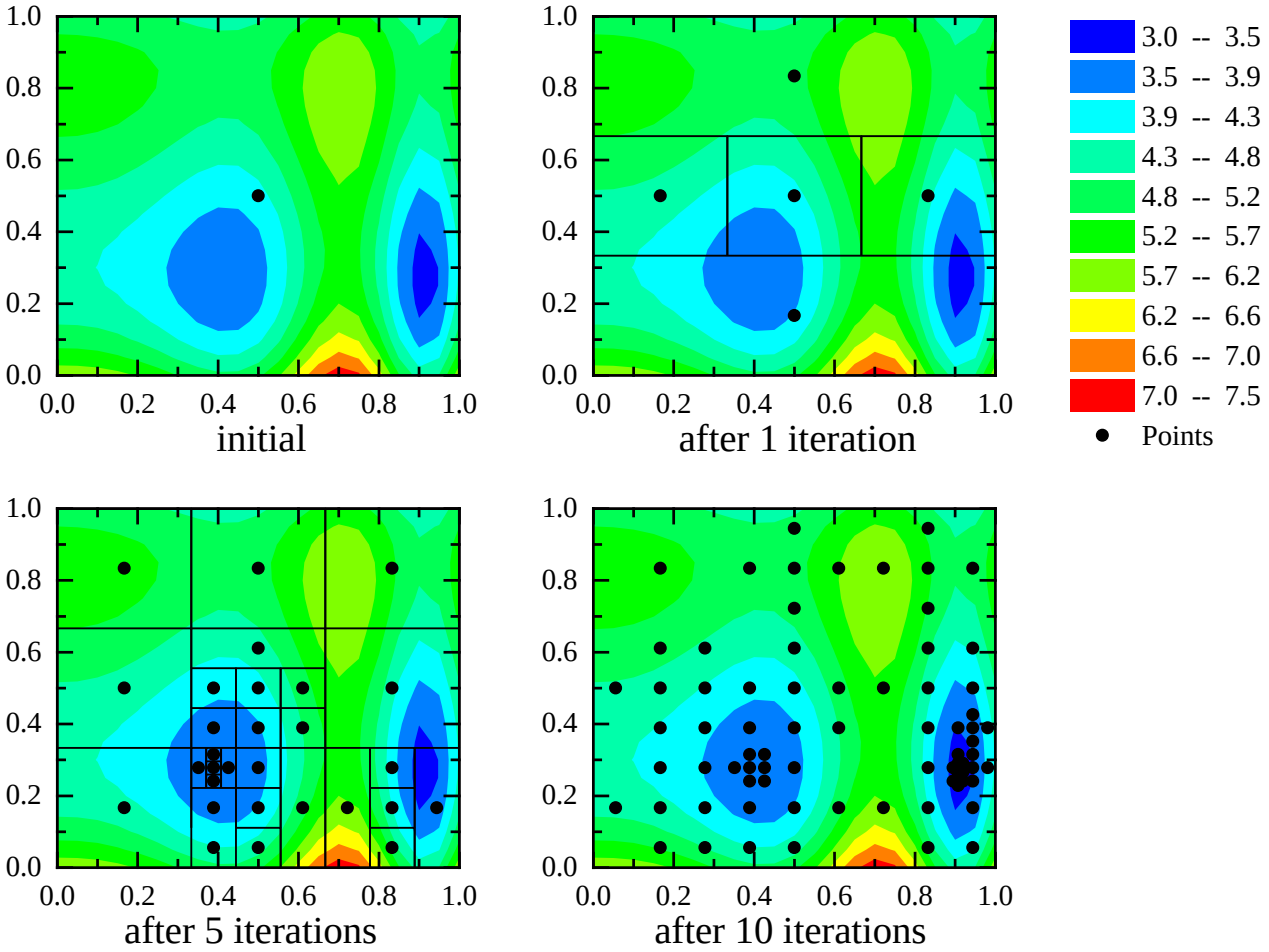
Sparse Days 2006
CERFACS, Toulouse

Outline

1. Overview of DIRECT
2. Motivation for parallelization strategies — pDIRECT
3. Implementation and Evaluation
   - Memory usage
   - Task allocation
4. Conclusion and Future Work

DIRECT Global Search Algorithm: Overview

DIviding-RECTangles, [Jones et al., 1993]



DIRECT Global Search Algorithm: Algorithm description

Given an objective function f and the design space $D = [\ell, u]$:

Step 1. Normalize the design space D to be the unit hypercube. Sample the center point c of this hypercube and evaluate $f(c)$. Initialize $f_{\min} = f(c)$, evaluation counter $m = 1$, and iteration counter $t = 0$.

Step 2. Identify the set S of potentially optimal boxes.

Step 3. Select any box $j \in S$.

Step 4. Divide the box j as follows:

(1) Identify the set I of dimensions with the maximum side length. Let δ equal one-third of this maximum side length.

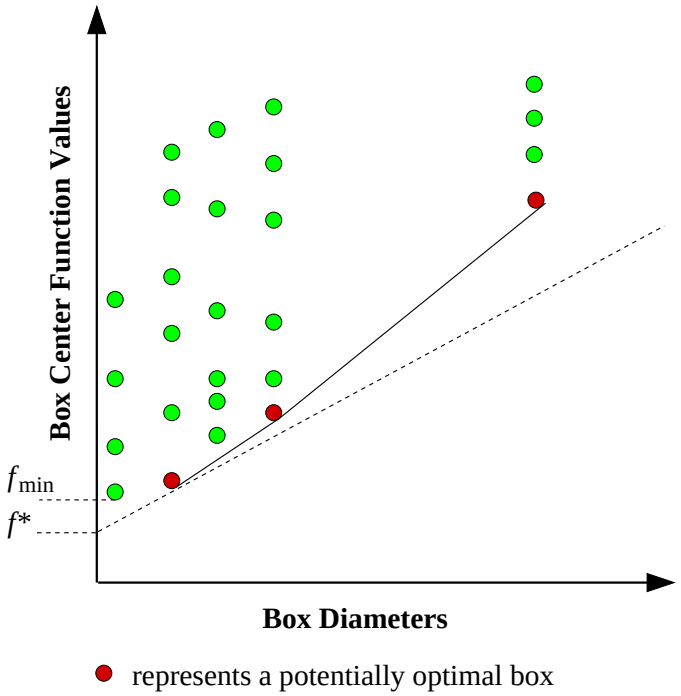
(2) Sample the function at the points $c \pm \delta e_i$ for all $i \in I$, where c is the center of the box and e_i is the ith unit vector.

(3) Divide the box j containing c into thirds along the dimensions in I, starting with the dimension with the lowest value of $w_i = \min\{f(c + \delta e_i), f(c - \delta e_i)\}$, and continuing to the dimension with the highest w_i. Update $f_{\min}$ and m.

Step 5. Set $S = S - \{j\}$. If $S \neq \emptyset$ go to Step 3.

Step 6. Set $t = t + 1$. If iteration limit or evaluation limit has been reached, stop. Otherwise, go to Step 2.

DIRECT: Global convergence property



- Box selection rule: box j is potentially optimal if

$$f(c_j) - \tilde{K} d_j \leq f(c_i) - \tilde{K} d_i,$$

$$f(c_j) - \tilde{K} d_j \leq f_{\min} - \epsilon |f_{\min}|,$$

  for some $\tilde{K} > 0$ and $i = 1, \ldots, m$ (the total number of subdivided boxes)

- Lipschitz continuity is required in the domain.

Summary of DIRECT

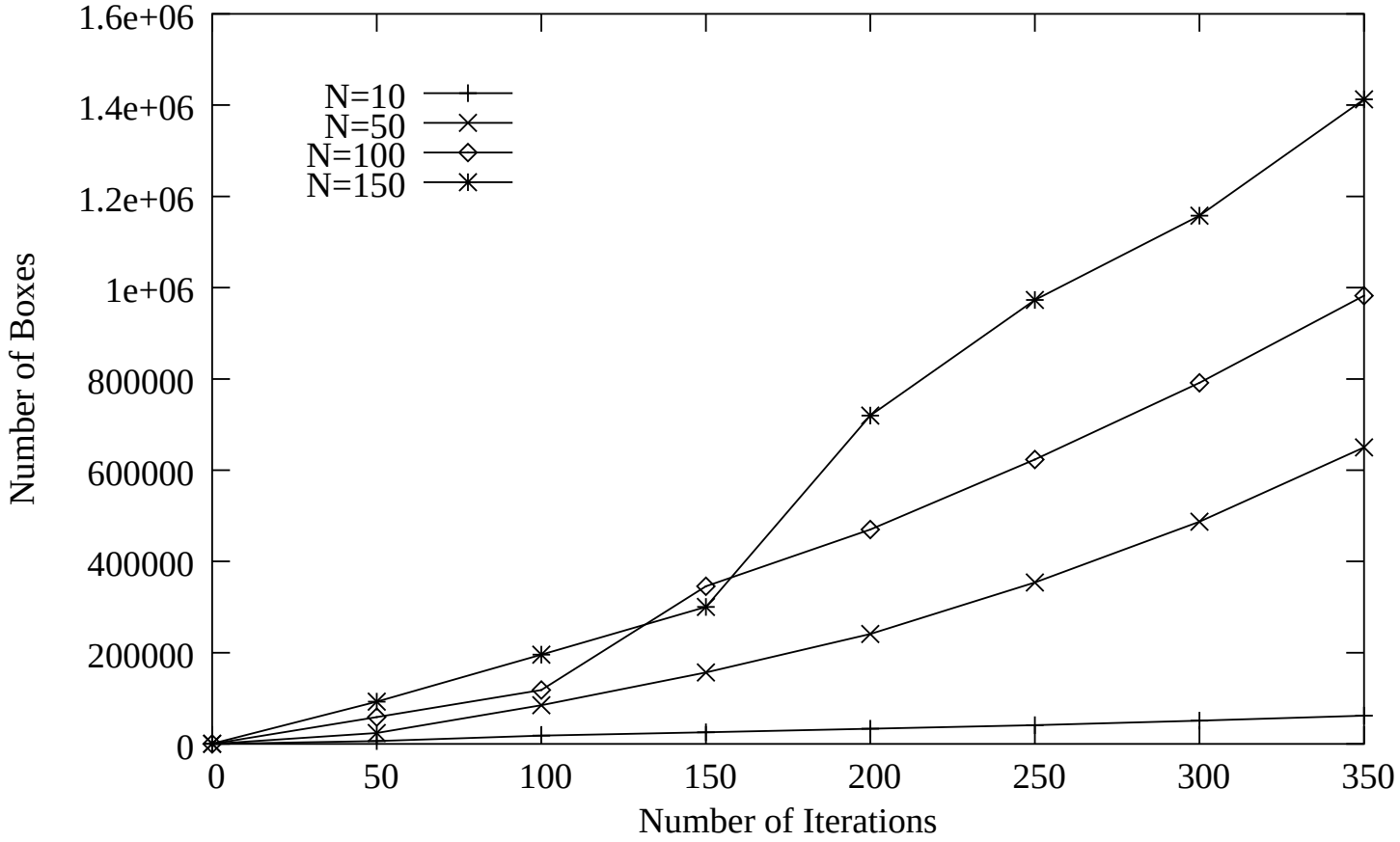
DIRECT finds the global minimum of a function $f(x)$ inside an N-dimensional box $\ell \leq x \leq u$. Each iteration of DIRECT consists of the following three main steps.

1. SELECTION identifies a set S of "potentially optimal" boxes with dimension N that are subregions inside the design domain.
2. SAMPLING evaluates new points around the center of each "potentially optimal" box in S.
3. DIVISION subdivides "potentially optimal" boxes in S based on the function values at the newly sampled points.

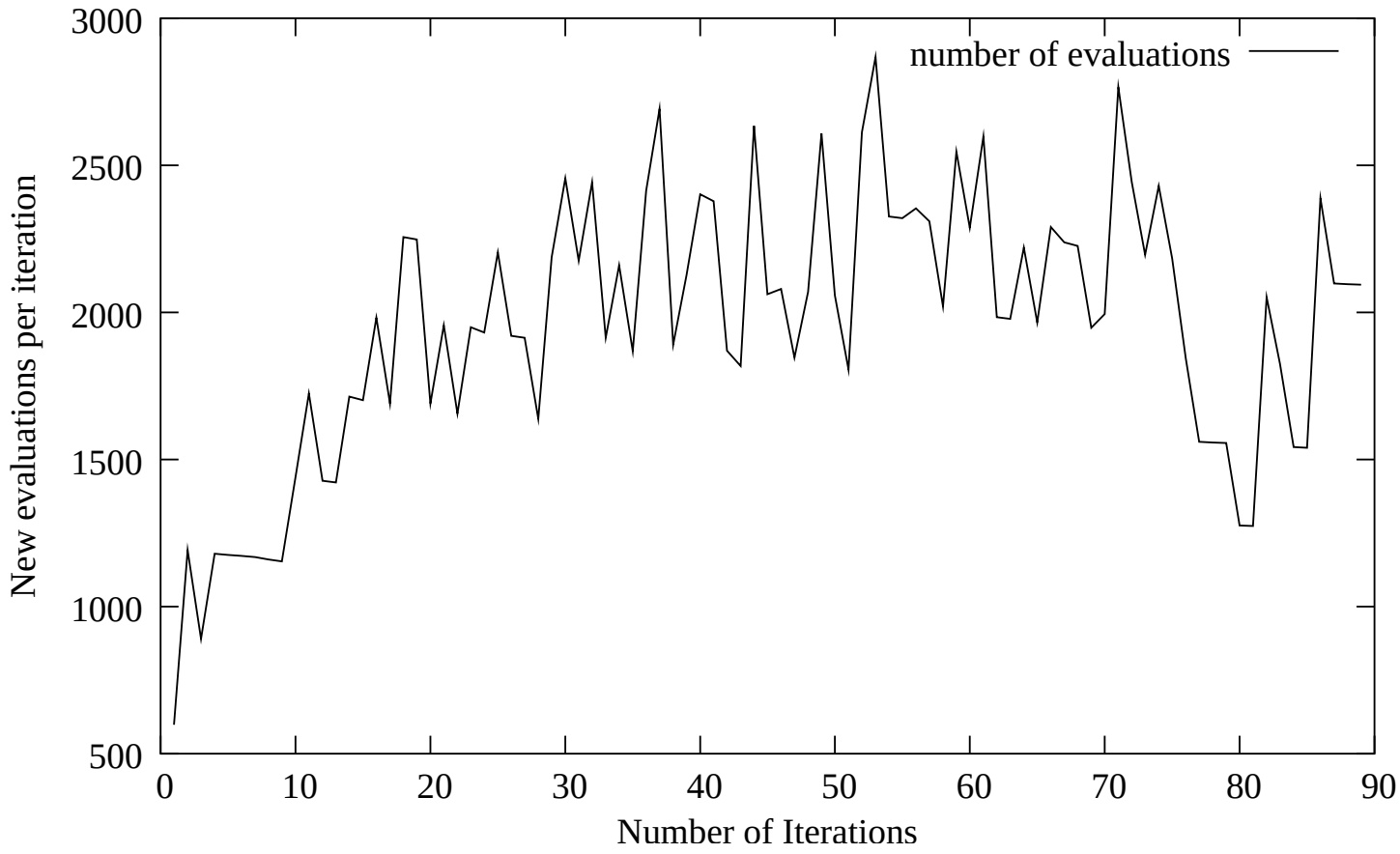
Observations: single start point and data dependency.

Motivation: Memory requirements

- "Curse of dimensionality".



Motivation: Unpredictable workload



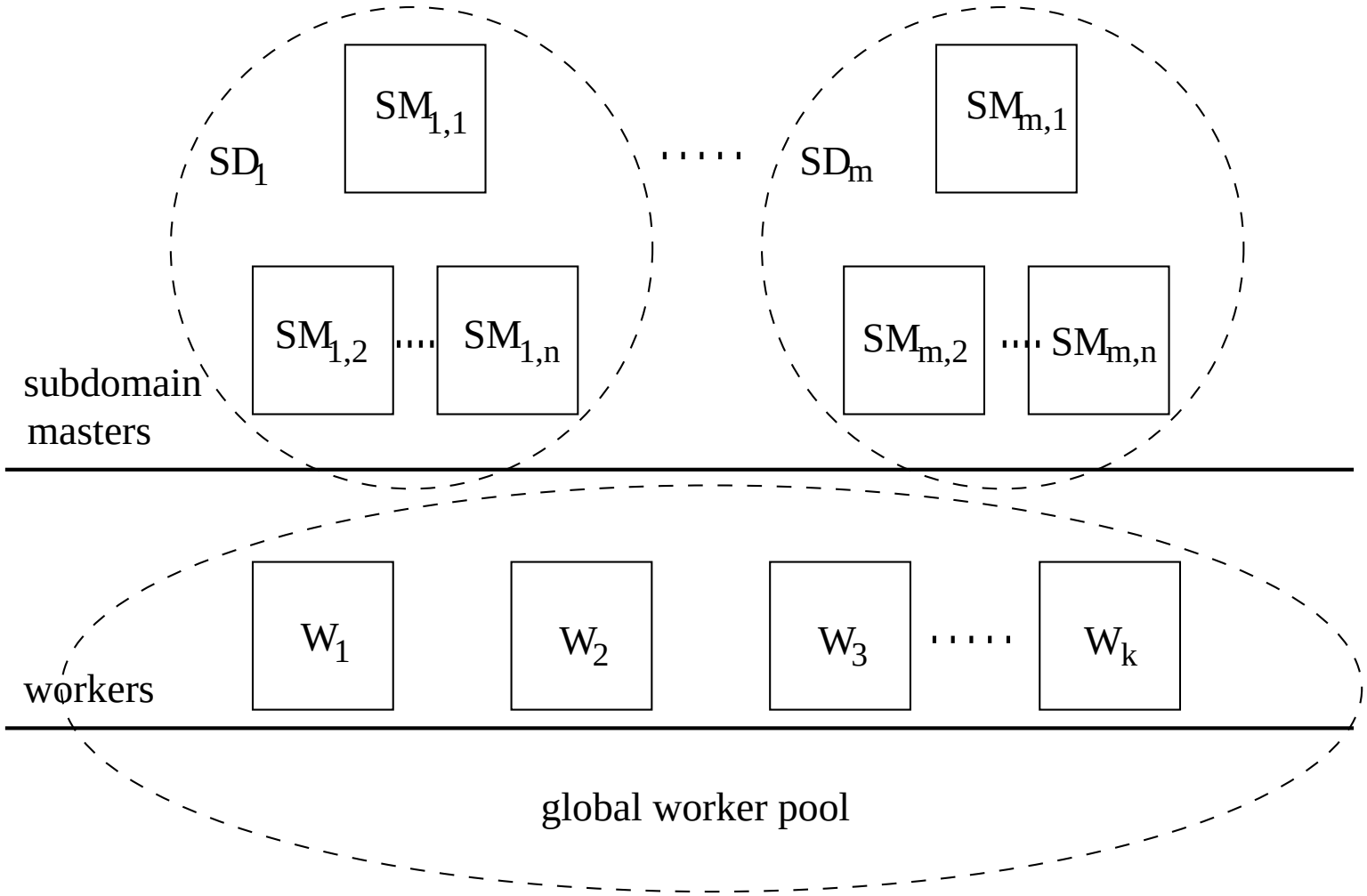
Motivation
Need for Performance Studies

1. Performance characteristics for pDIRECT
   - Memory usage
   - Load balancing
   - Parallel execution time and efficiency
2. Current state of System X
   - 2,200-processor Apple Xserve G5 cluster at Virginia Tech
   - 12.25 TFlops on LINPACK performance benchmark
   - Each node has two 2.3 GHz processors, 4 GB of main memory
   - Network: Infiniband
3. Test functions
   - Artificial functions (1–7): $N = 150$, settable function evaluation time T_f
   - A real world application (8): parameter estimation for the budding yeast cell cycle model, 36 ODEs, $N = 143$, $T \approx 8$ sec

The parallel scheme of pDIRECT

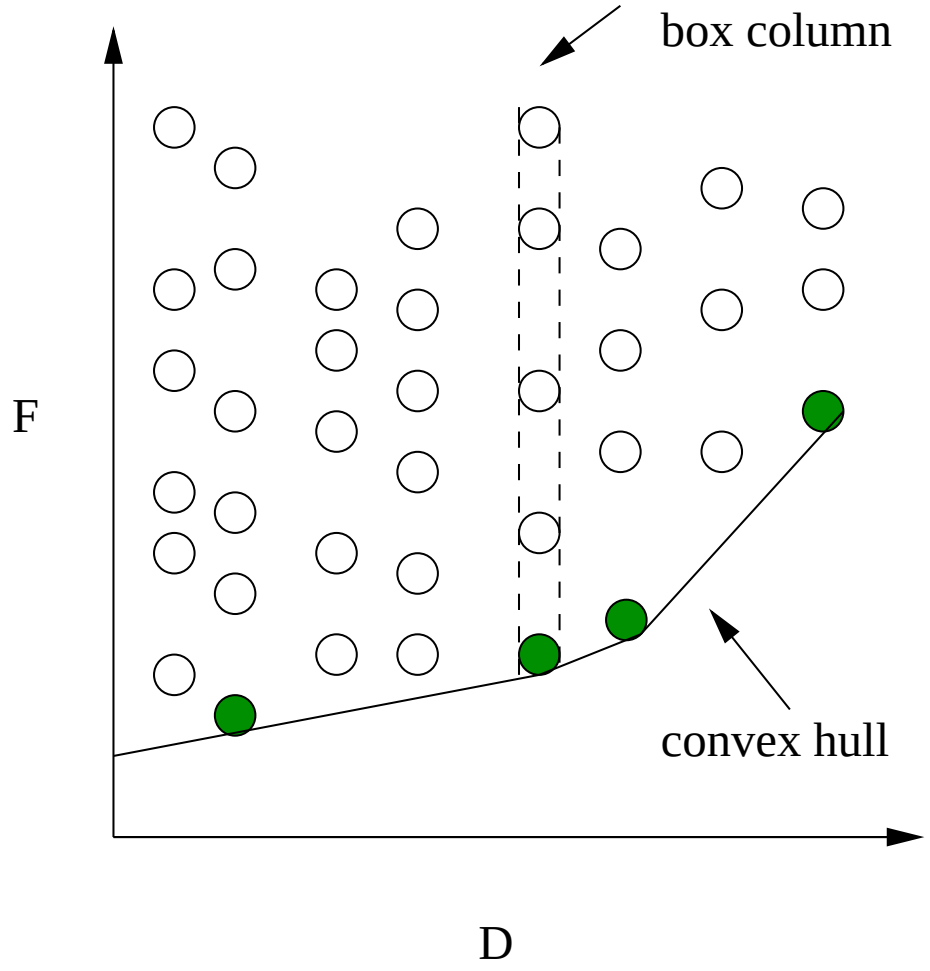
#	Description
1	$f = x \cdot x/3000$
2	$f = -\sqrt{\sum_{i=1}^{N} x_i - 0.5(i-1)}/N$
3	$f = 1 + \sum_{i=1}^{N} {x_i}^2/500 - \prod_{i=1}^{N} \cos(x_i/\sqrt{i}))$
4	$f = \sum_{i=1}^{N} 2.2 \times (x_i + 0.3)^2 - (x_i - 0.3)^4$
5	$f = \sum_{i=1}^{N} \sum_{j=1}^{i} {x_j}^2$
6	$f = \sum_{i=1}^{N} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$
7	$f = 10N + \sum_{i=1}^{N} x_i^2 - 10\cos(2\pi x_i)$
8	Budding yeast parameter estimator

The parallel scheme of pDIRECT



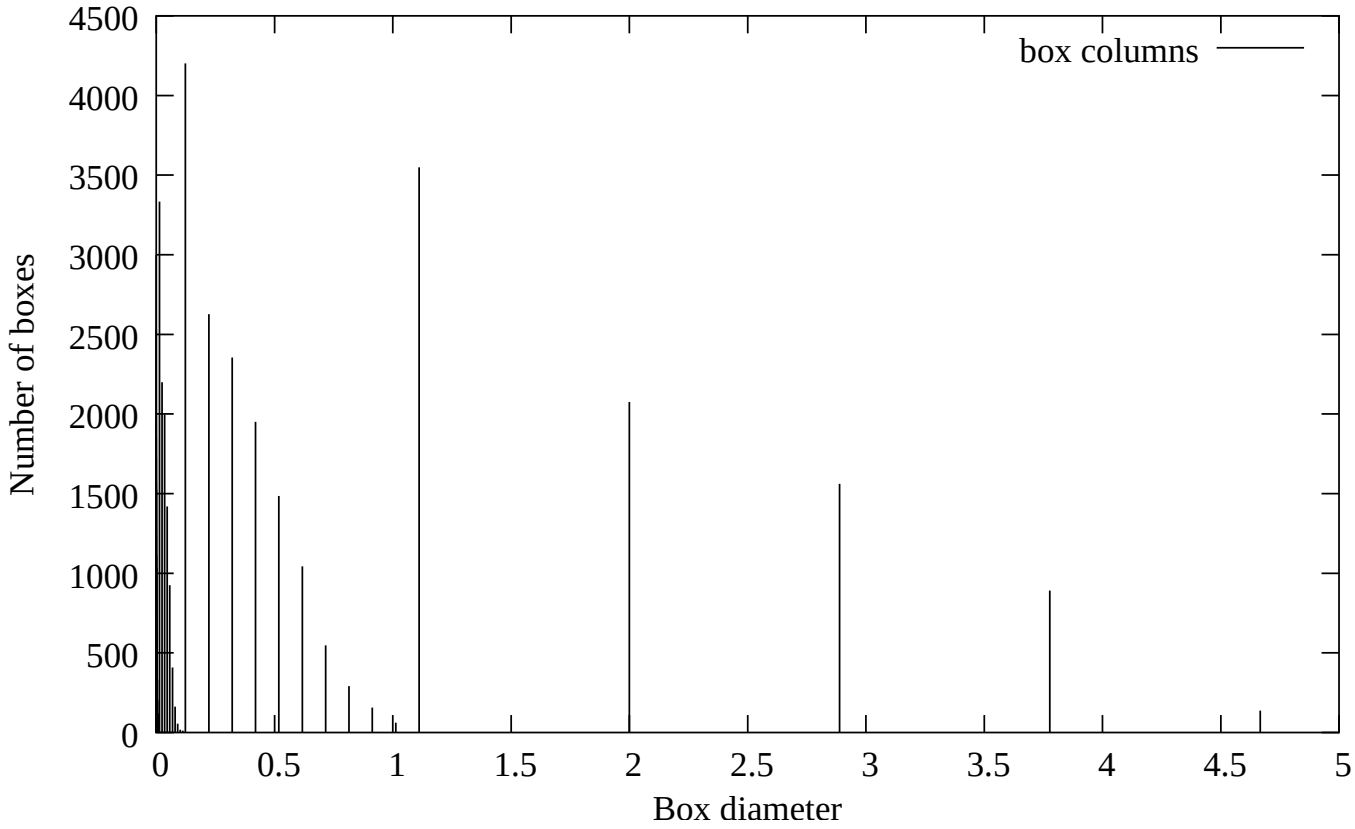
Implementation and Evaluation

Data structures



Implementation and Evaluation
Data structures

- LBC (limiting box column length) to $L = I_{max} - I_{current} + 1$.



Box column lengths at the last iteration $I_{\max} = 400$ for a test function with $N = 10$ without LBC.

Implementation and Evaluation

Memory usage with LBC and without LBC (NON-LBC)

	NON-LBC		LBC		% diff.
#	I_{out}	$C_{\text{real}}/10^6$	I_{out}	$C_{\text{real}}/10^6$	
1	159	153	273	112	26
2	79	164	647	77	52
3	213	162	391	111	31
4	90	163	1000	82	50
5	286	164	467	124	24
6	163	160	328	109	31
7	78	162	377	85	47

- LBC reduces the size of the data structures by 20–50%.
- As the number of box columns grows larger without limit, a single master may be able to run fewer iterations without LBC than with LBC.

Implementation and Evaluation
Task Allocation for SELECTION

The parallel SELECTION is implemented as follows:

1. $SM_{i,j}$ identify local convex hull box sets $S_{i,j}$, $j = 1, \ldots, n$.
2. $SM_{i,1}$ gathers the $S_{i,j}$ from all the $SM_{i,j}$.
3. $SM_{i,1}$ merges the $S_{i,j}$ by box diameters and finds the global convex hull box set S_i.
4. All the $SM_{i,j}$ receive the global set S_i and find their portion of the convex hull boxes.

Implementation and Evaluation
Task Allocation for SELECTION

- Comparison of N_{gc}, N_{lc}, and N_d, where N_{gc} is the number of global convex boxes, N_{lc} is the number of local convex boxes, and N_d is the number of all lowest boxes.

#	N_{gc}	N_{lc}	N_d	$\frac{(N_d - N_{lc})}{N_d}$
1	58	2605	33358	92%
2	89	1228	6805	81%
3	145	2056	22375	90%
4	3	3201	6756	52%
5	140	1830	20276	90%
6	144	1276	28003	95%
7	144	3531	20614	82%
8	20	159	6545	97%

Implementation and Evaluation
Task Allocation for SAMPLING

Three ways of searching in multiple (e.g., four) subdomains with P processors:

(a) All P processors are used to run a multiple subdomain search with four subdomains. The parallel execution time is T_a.

(b) Four single subdomain searches are run in parallel, each using $P/4$ processors on one of the four subdomains. The overall parallel execution time T_b is the longest duration of all four runs.

(c) Four single subdomain searches are run sequentially, each using all P processors on each of the four subdomains. The parallel execution time T_c is the total duration of these four runs.

Implementation and Evaluation
Task Allocation for SAMPLING

Compare T_a, T_b, and T_c in seconds:

#	T_a	T_b	T_c	$\frac{(T_b - T_a)}{T_a}$	$\frac{(T_c - T_a)}{T_a}$
1	382	407	441	6.2%	15.4%
2	1132	1139	1185	0.6%	4.6%
3	358	369	417	3.1%	16.4%
4	870	874	921	0.4%	5.9%
5	260	263	312	1.1%	20.0%
6	428	476	477	11.2%	11.4%
7	1142	1148	1196	0.5%	4.7%
8	8595	10643	11059	23.8%	28.7%

Observe that $T_a \leq T_b \leq T_c$.

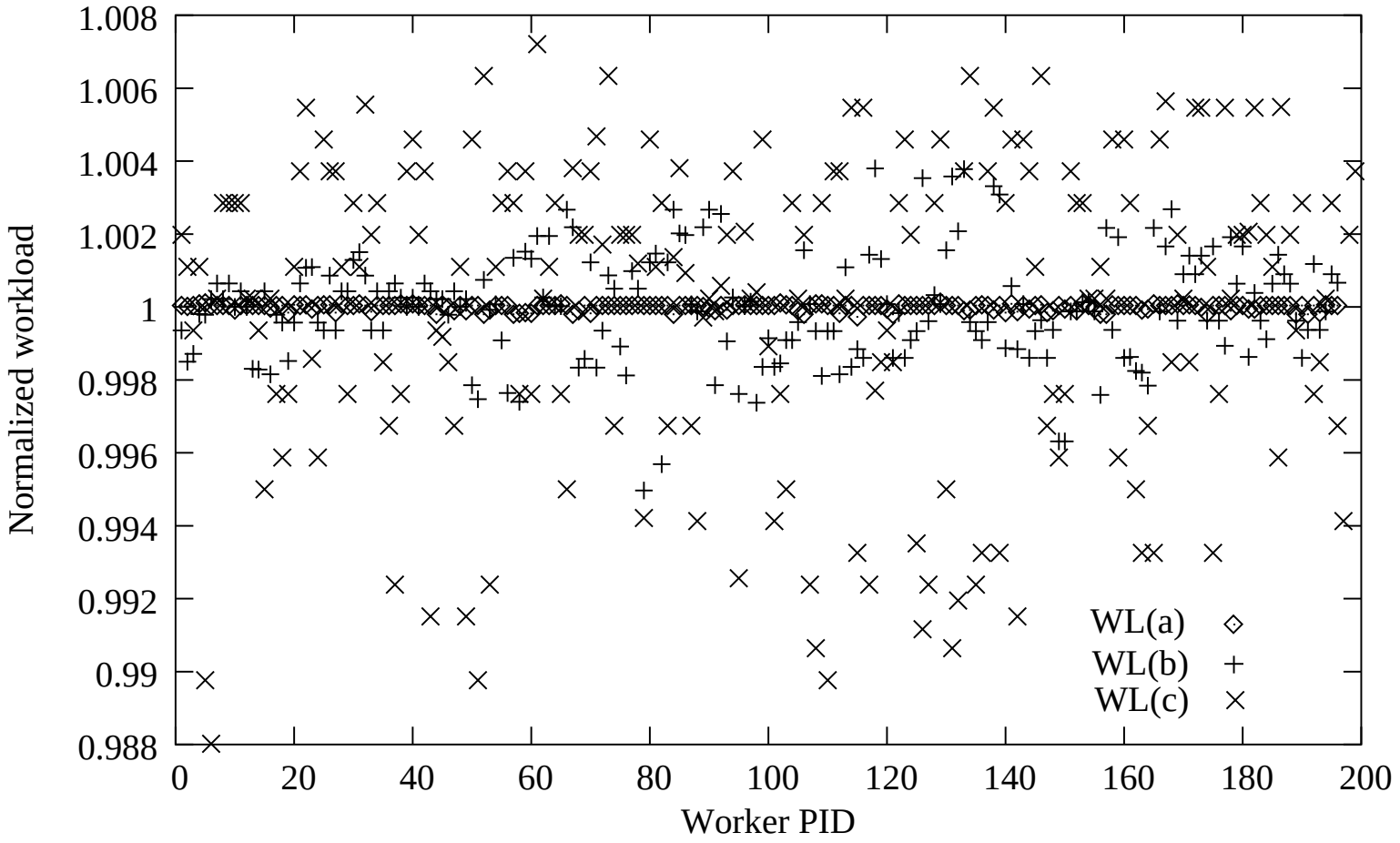
Implementation and Evaluation
Task Allocation for SAMPLING

Compare the total number of subdomain function evalutions:

#	e_1	e_2	e_3	e_4	$\bar{e}$	s^2
1	181409	194927	194927	181463	2090.9	7789.1^2
2	550691	550691	550691	550691	6118.8	0
3	176075	176075	176075	176075	1956.4	0
4	421723	421723	421723	421723	4685.8	0
5	123685	123685	123685	123685	1374.3	0
6	228193	203635	198727	192397	2286.0	15661^2
7	555435	555435	555435	555435	6171.5	0
8	82471	44631	47531	87063	1635.6	22445^2

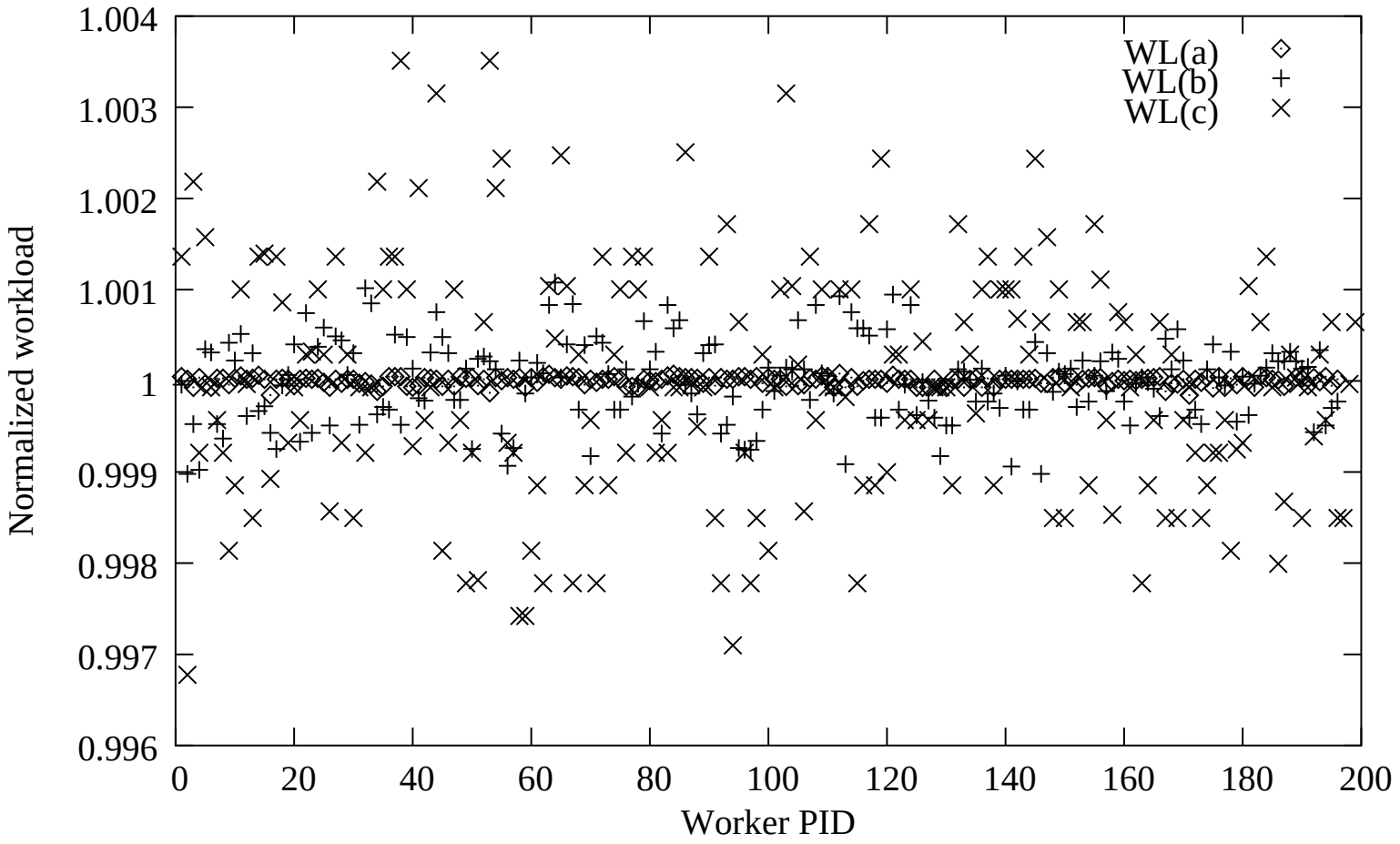
Note that $\bar{e} = \sum e_i / I_{\max}$ and s^2 is the variance of the total number of function evaluations for the four subdomains.

Implementation and Evaluation
Task Allocation for SAMPLING



The workload patterns for Test Function 6.

Implementation and Evaluation
Task Allocation for SAMPLING



The workload patterns for Test Function 7.

Conclusion and Future Work

Conclusions:

1. Evaluated memory usage: LBC reduces memory requirements by 20–50%.

2. Evaluated task allocation for SELECTION: parallel SELECTION reduces buffer size and network traffic by 50–95%.

3. Evaluated task allocation for SAMPLING: sharing workers across subdomains improves load balancing. Multiple subdomain search allows masters from different subdomains to provide work to the globally shared workers at different times, especially for subdomains that generate different amounts of work.

Future work:

1. Reduce memory usage further.
2. Pre-compute points to mitigate the data dependency.
3. Improve convex hull computation.
4. Evaluate performance of pDIRECT on different large scale parallel systems.

References

- J. He, L. T. Watson, N. Ramakrishnan, C. A. Shaffer, A. Verstak, J. Jiang, K. Bae, and W. H. Tranter, “Dynamic data structures for a direct search algorithm”, *Computational Optimization and Applications*, vol. 23, pp. 5–25, 2002.
- D. R. Jones, C. D. Perttunen, and B. E. Stuckman, “Lipschitzian optimization without the Lipschitz constant”, *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.
- T.D. Panning, L. T. Watson, N. A. Allen, K. C. Chen, C. A. Shaffer, and J. J. Tyson, “Deterministic global parameter estimation for a model of the budding yeast cell cycle”, *Journal of Global Optimization*, to appear.