The Deterministic Global Optimization Algorithm DIRECT

Layne T. Watson and Jian He
Departments of Computer Science and Mathematics
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106 USA

HPCS/BIS Tutorial, April 16, 2008
Ottowa, Canada

Virginia Tech
1872

Outline

1. Introduction
2. DIRECT Global Search Algorithm (Jones, Perttunen, and Stuckman, 1993)
   - DIRECT = global + local
   - DIviding RECTangles in action
   - Algorithm description
   - Global convergence property
   - Algorithm modifications
3. Dynamic Data Structures
   - Motivation
   - Box structures
   - Linked list structures
   - Performance studies

Outline

4. Parallel Implementation
   - Memory reduction techniques
   - Parallel scheme
   - Systems biology applications
   - Effect of chunk size
   - Worker workload range modelling
   - Extreme parallel schemes
   - Computer code VTDIRECT95
5. Mesh Adaptive Direct Search (MADS) (Audet and Dennis, 2006)
   - Example
   - Algorithm details
6. Applications
   - Aircraft design: HSCT (high speed civil transport)
   - Wireless communication system design: S^4W (site-specific system simulator for wireless system design)
   - Budding yeast cell cycle model
7. Conclusion
8. References

Introduction
Optimization problem and category

1. Problem statement:

$$\min_{x \in D} f_0(x),$$

$$D = \{x \in D_0 \mid f_j(x) \leq 0,\ j = 1, \ldots, J\},$$

where $D_0 = \left\{x \in E^n \mid \ell \leq x \leq u\right\}$ is a simple box constraint set.

2. Problem category

- Discrete domain D.
  Combinatorial problems.
- Continuous domain D.
  Unconstrained (or with simple bound constraints) and constrained problems.

Introduction

Optimization approaches (classified by Taylor's approximation)

1. **Second order:**

   For example, Newton-like methods using the Hessian matrix.

2. **First order:**

   For example, steepest descent methods using the gradient vector.

3. **Zero order:**

- Deterministic search methods:

  For example, direct search methods—pattern search (DIRECT) and simplex-based search (multi-directional search, adaptive directional search, etc.), and branch and bound search methods.

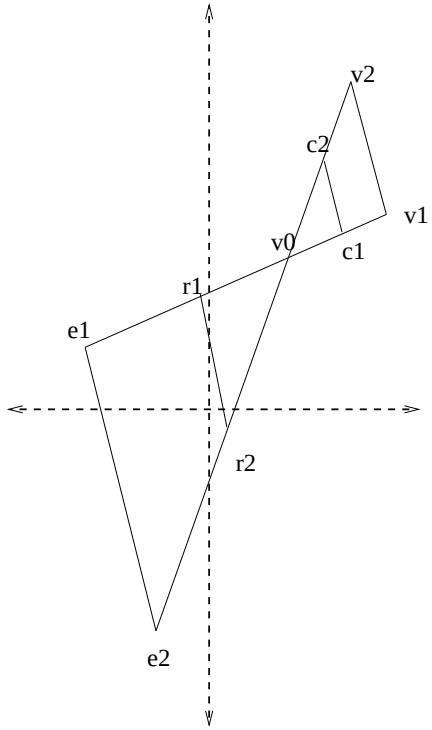
- Nondeterministic (heuristic) search methods:

  For example, simulated annealing, Tabu search, and genetic algorithms.

Introduction
Direct search methods

1. **Characteristics**

   Derivative-free; only function values are needed; global convergence properties are guaranteed under some conditions.
2. **Example: simplex-based multi-directional search** (Dennis and Torczon, 1991)



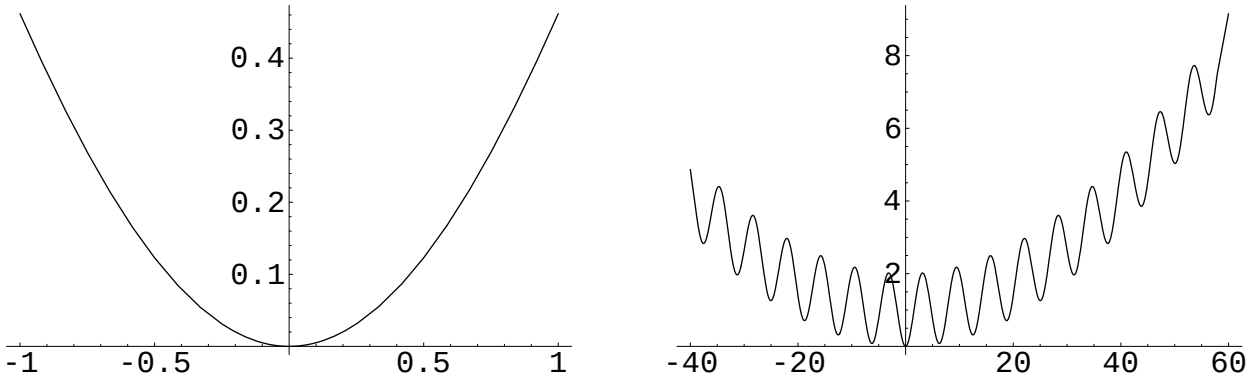
Original simplex: (v0, v1, v2)
Search steps:
(1) Reflection: (v1, v2) to (r1, r2)
(2) Expansion: (r1, r2) to (e1, e2)
(3) Contraction: (v1, v2) to (c1, c2)

DIRECT Global Search Algorithm

DIRECT = global search + local search

It is a good choice for engineering design problems, which have

1. no convexity assumptions (local==global for strictly convex functions),
2. multiple local minima (local search is easily trapped),
3. and black-box designs (unpredictable system parameters).

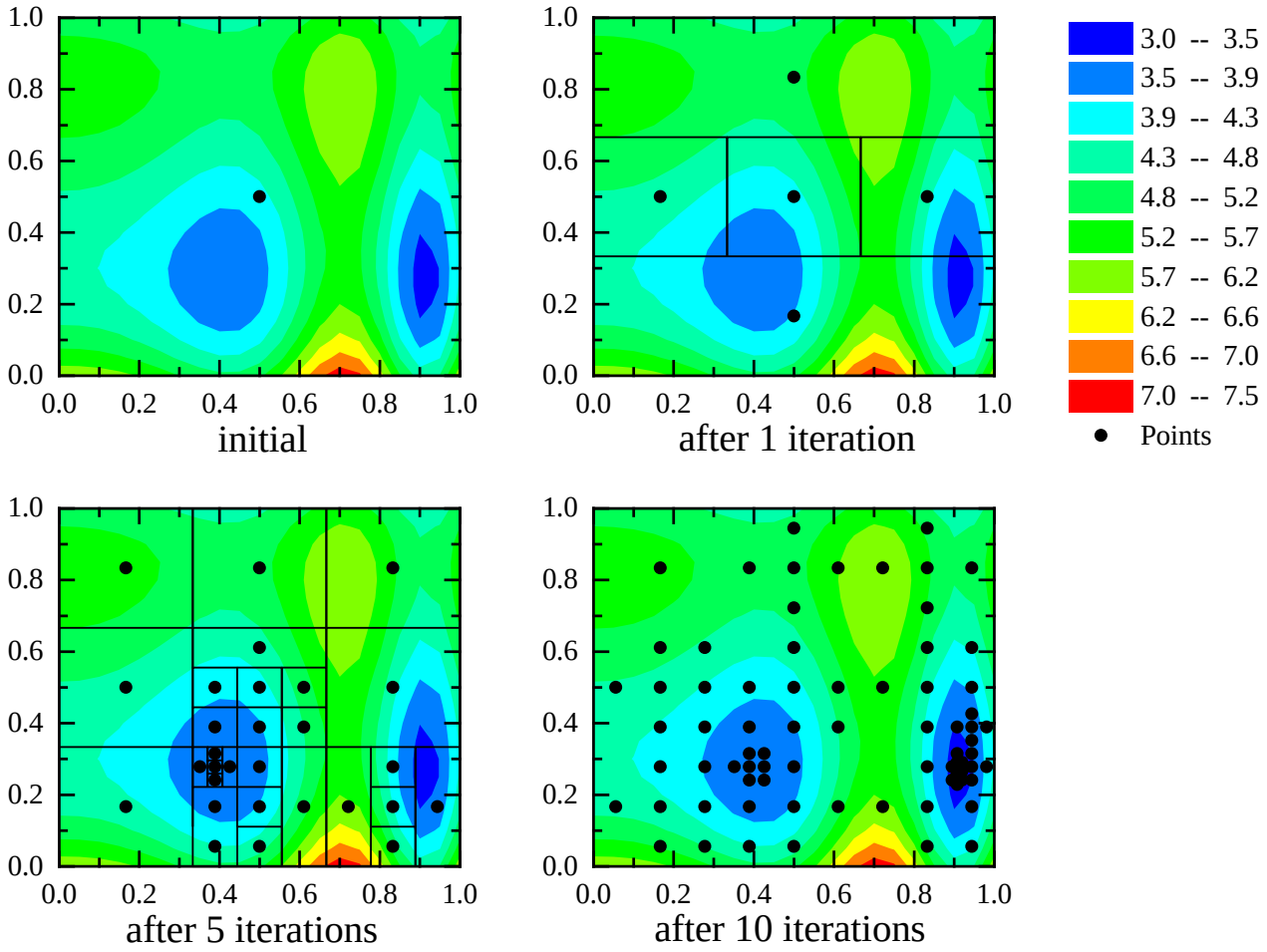


Example: one-dimensional Griewank function with $d = 500$ (right).

$$f(x) = 1 + \sum_{i=1}^{n} \frac{{x_i}^2}{d} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

DIRECT Global Search Algorithm

DIviding-RECTangles in action



DIRECT Global Search Algorithm
Algorithm description

Given an objective function f and the design space $D = D_0$:

Step 1. Normalize the design space D to be the unit hypercube. Sample the center point c_i of this hypercube and evaluate $f(c_i)$. Initialize $f_{\min} = f(c_i)$, 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.

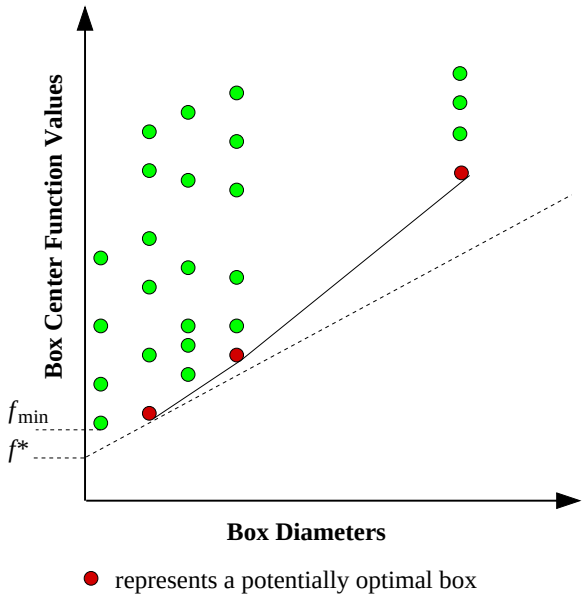
(2) Sample the function at the points $c \pm \delta e_i$ for all $i \in I$, where c is the center of the box and e_i is the ith unit vector.

(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 Search Algorithm
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.

DIRECT Global Search Algorithm
Algorithmic Options

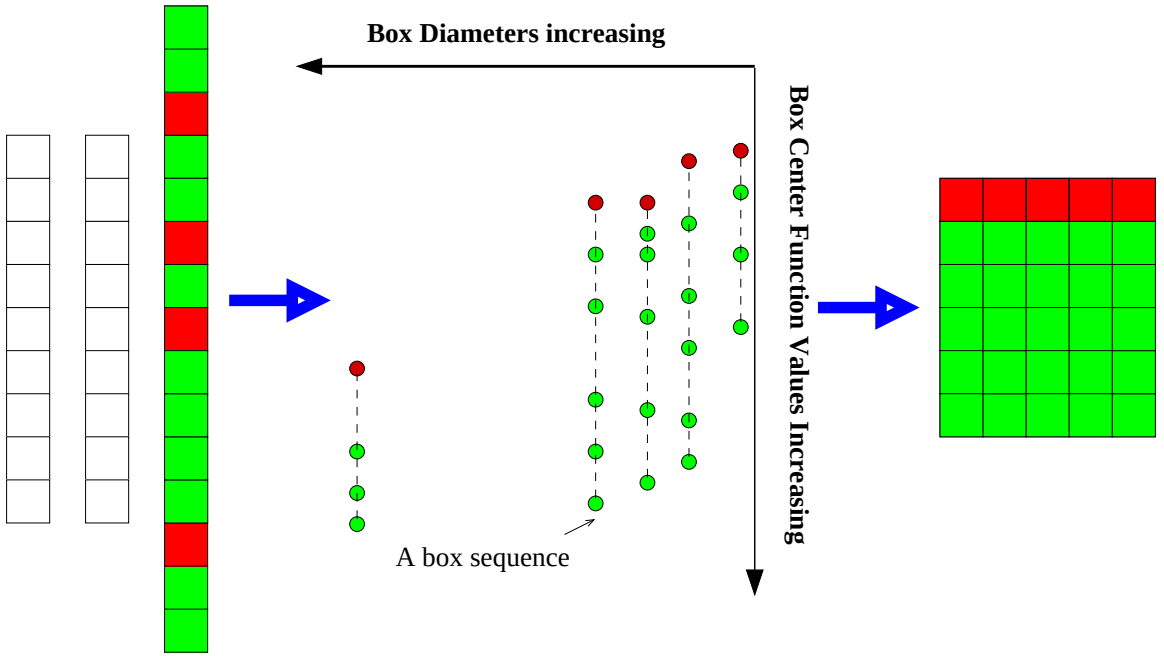
1. **Box selection rules**
   - Optional "aggressive switch": Switch on/off the convex hull processing used in identifying potentially optimal boxes.
   - $\epsilon = 0$ by default.
2. **Stopping rules**
   - Number of iterations or function evaluations.
   - Minimum diameter: Terminate when the best potentially optimal box's diameter is less than this minimum diameter.
   - Objective function convergence tolerance:

$$\tau_f = \frac{\tilde{f}_{\min} - f_{\min}}{1.0 + \tilde{f}_{\min}},$$

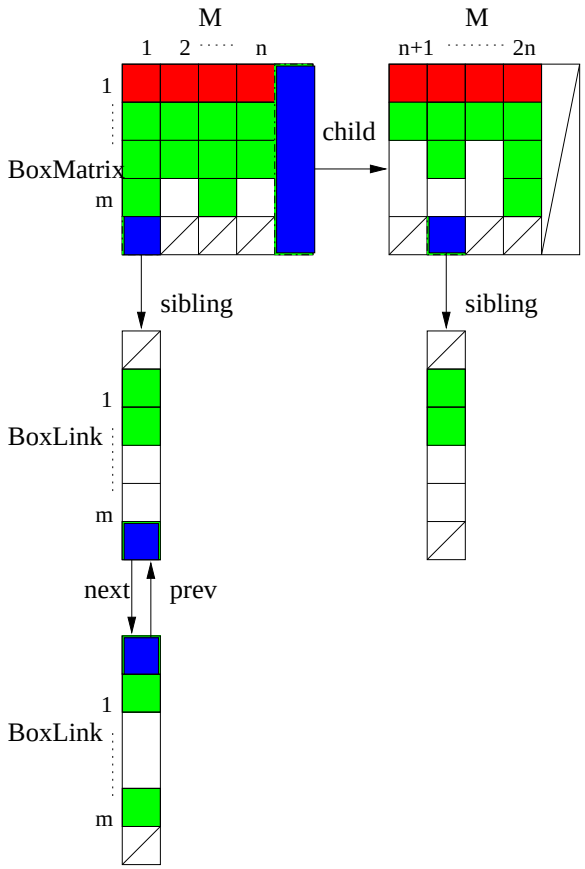
where $\tilde{f}_{\min}$ represents the previous computed minimum. The algorithm stops when τ_f becomes less than a user specified value.

Dynamic Implementation
Motivation

1. **Challenges:**
   - Unpredictable storage requirement
   - Execution overhead of subdividing potentially optimal boxes
2. **Better mapping:** from 1-D data structures to a 2-D data structure.

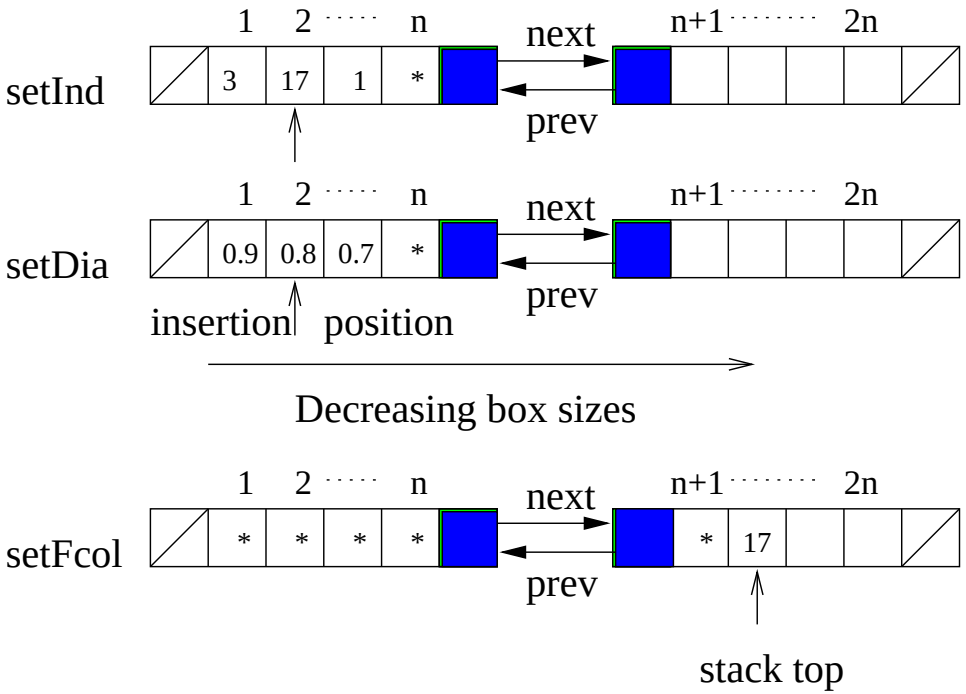


Dynamic Implementation
Box structures



- Two-dimensional dynamic structure
- Priority queue vs. sorted list

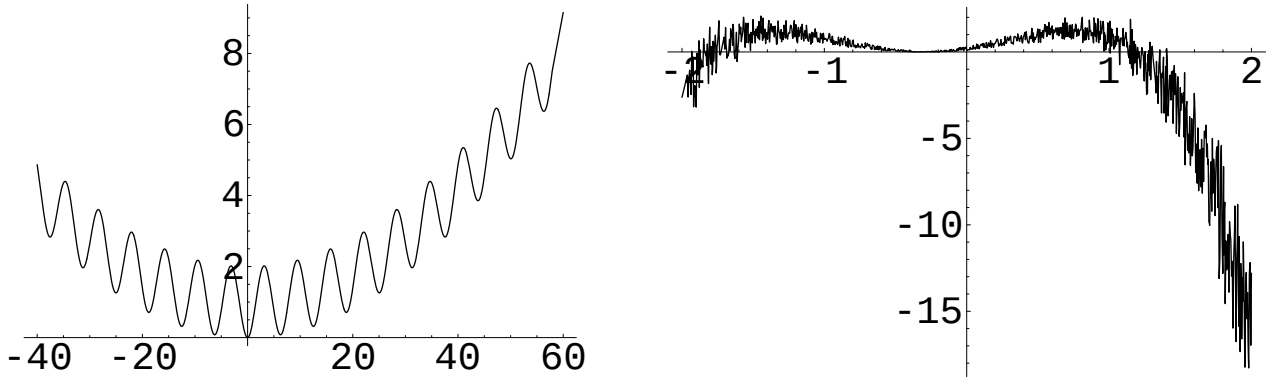
Dynamic Implementation
Linked list structures



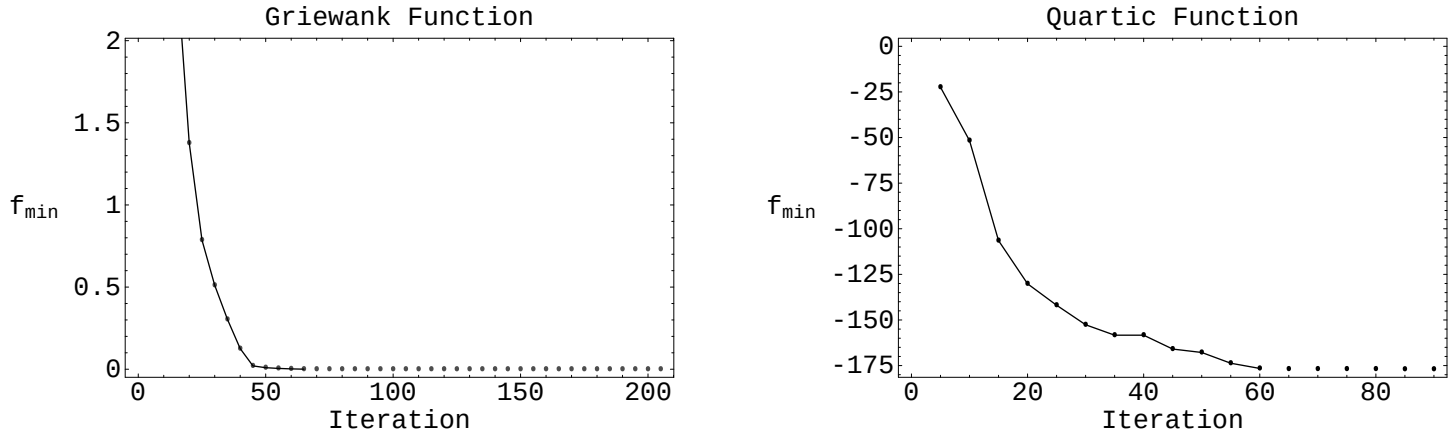
- Maintain 2-D structure
- Recycle box sequence columns

Dynamic Implementation
Performance studies

- Objective function convergence tolerance τ



One-dimensional Griewank function and quartic function



Comparison of $\tau = 0.0001$ (solid) and $\tau = 0$ (dotted)

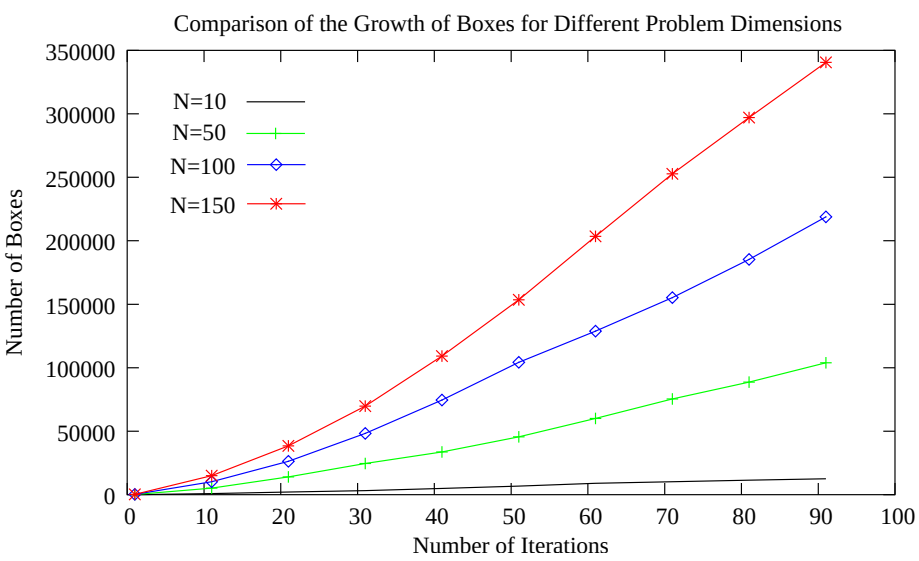
Dynamic Implementation
Performance studies (cont.)

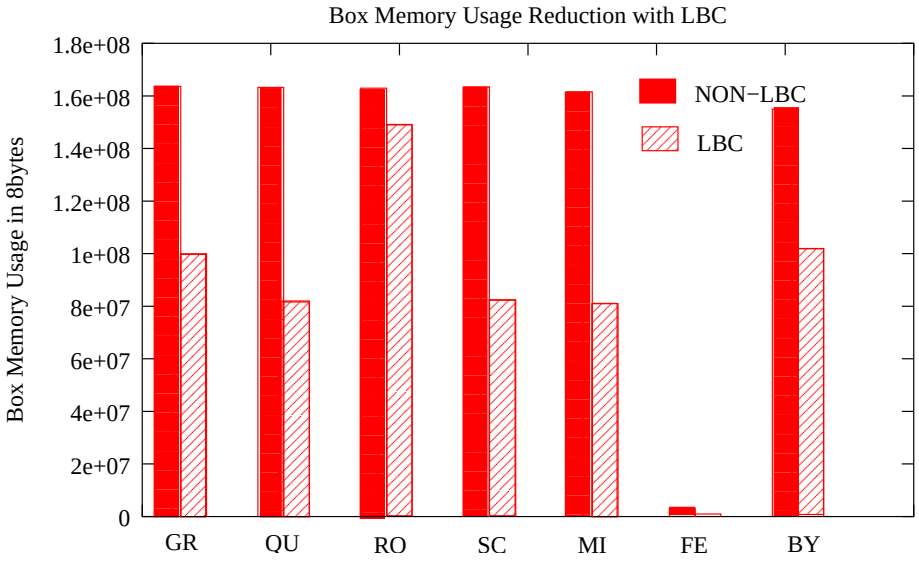
- Comparison of static and dynamic implementations (dimension n, L iterations, time in $msec$, memory usage (maximum working set size) in pages (1 page = 512 bytes)).

			Baker		Gablonsky		dynamic structures	
Problem	n	L	time	memory	time	memory	time	memory
Griewank	2	50	172	10264	34	2224	85	1040
Griewank	5	50	199	11504	34	2352	73	1024
Griewank	10	50	310	15424	51	2648	110	1616
Griewank	15	50	639	18280	88	3232	192	2744
Griewank	20	50	*	*	170	4464	397	6080
Griewank	50	70	*	*	*	*	6161	82664
Quartic	2	50	108	10240	26	2176	25	520
Quartic	5	50	151	11488	31	2240	27	528
Quartic	10	50	441	15432	36	2472	58	1160
Quartic	15	50	1260	18336	54	2992	125	2176
Quartic	20	50	*	*	82	3872	240	4560
Quartic	50	90	*	*	*	*	6572	86656

Memory Reduction Techniques

If MAX_ITER is used, **LBC (limiting box columns)** only keeps $I_{\max} - I_{\text{current}} + 1$ boxes in each box column.



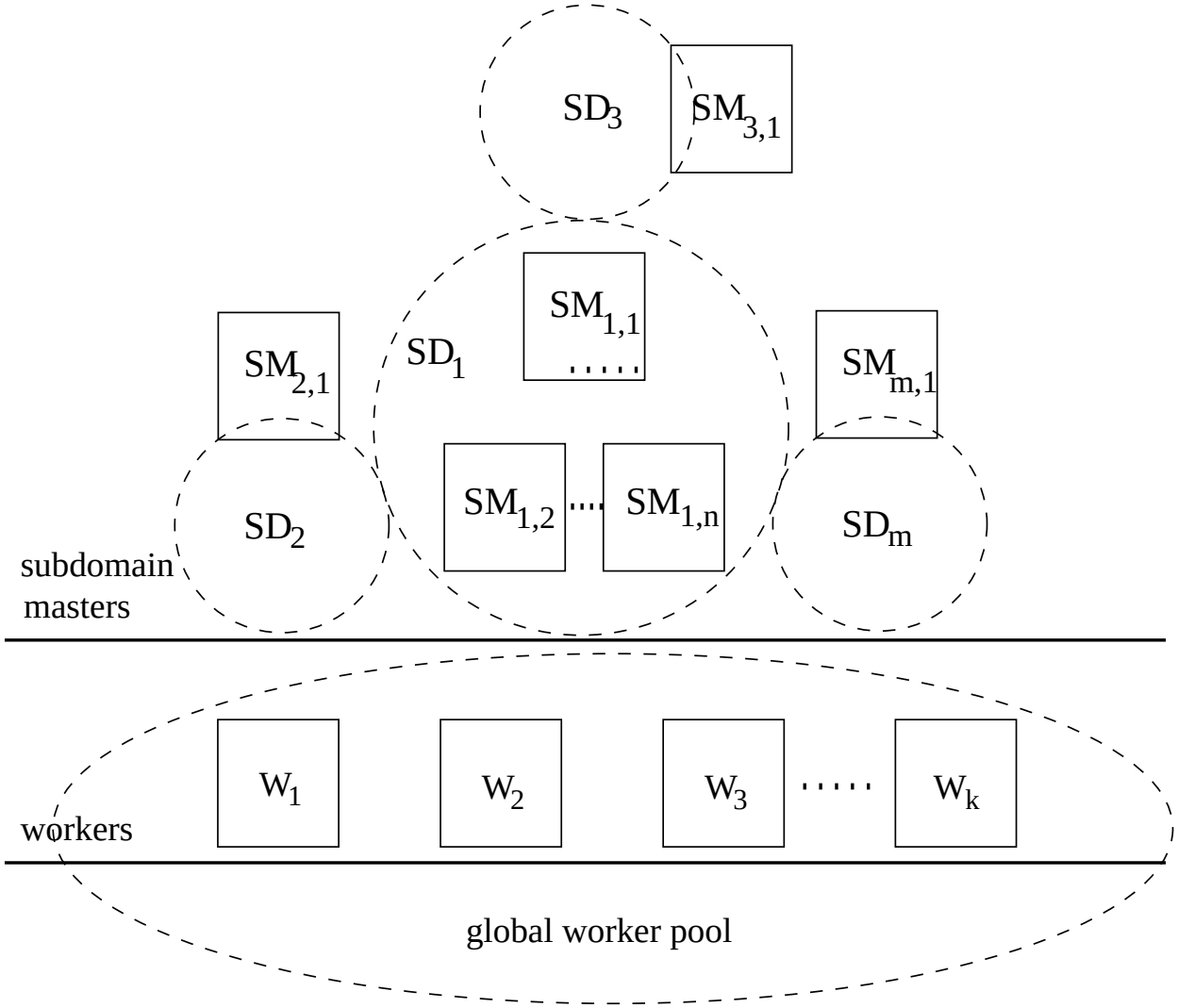


(Left) For the original DIRECT, the memory requirement imposed by the intermediate data grows rapidly as the problem dimension N grows.

(Right) LBC reduces the memory usage by 10–70% for the selected high-dimensional test problems, including two real world applications—cell cycle modeling for frog eggs (FE) and budding yeast (BY) in systems biology.

Parallel Scheme

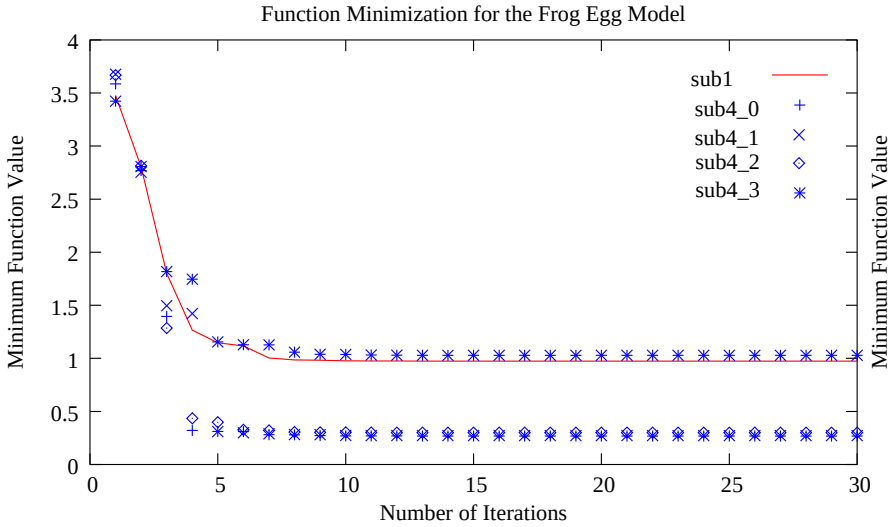
A design with a pure message passing model and globally shared workers.

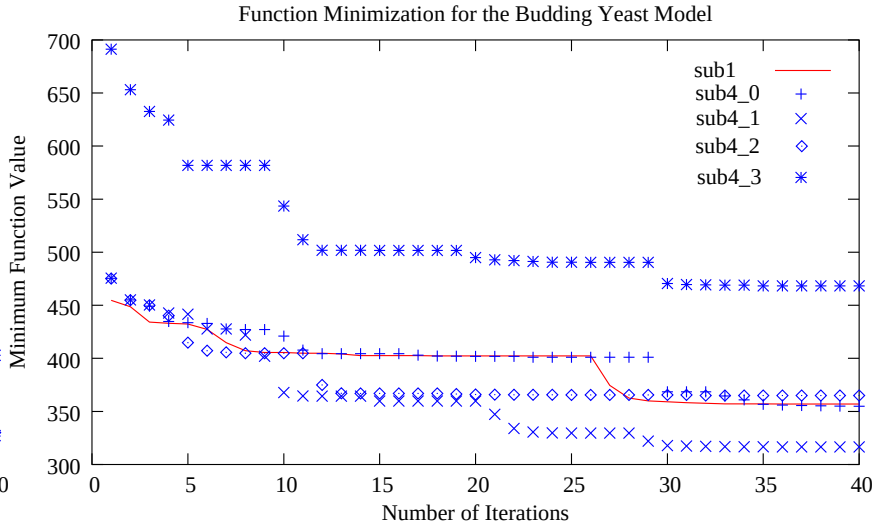


Application in Systems Biology

Good test cases for pVTdirect: high-dimensional, nonlinear ODEs, expensive function evaluation cost, multiple local minima.

Model	# ODEs	# parameters	Cost
Frog Eggs	3	16	3 sec.
Budding Yeast	36	143	11 sec.



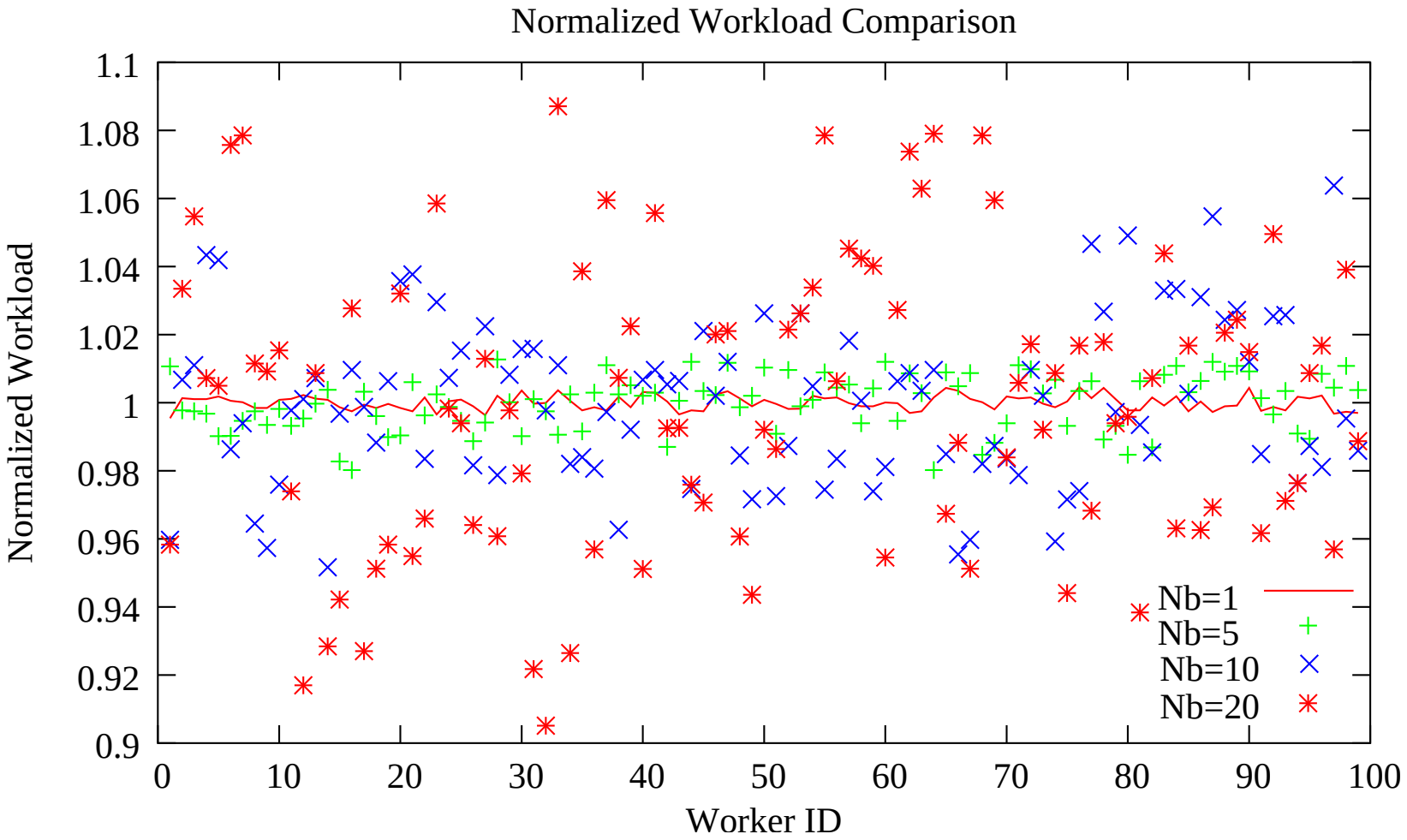


With more function evaluations generated across multiple subdomains, the same MAX_ITER likely yields a better solution than the single domain search, especially for problems with irregular, asymmetric structures.

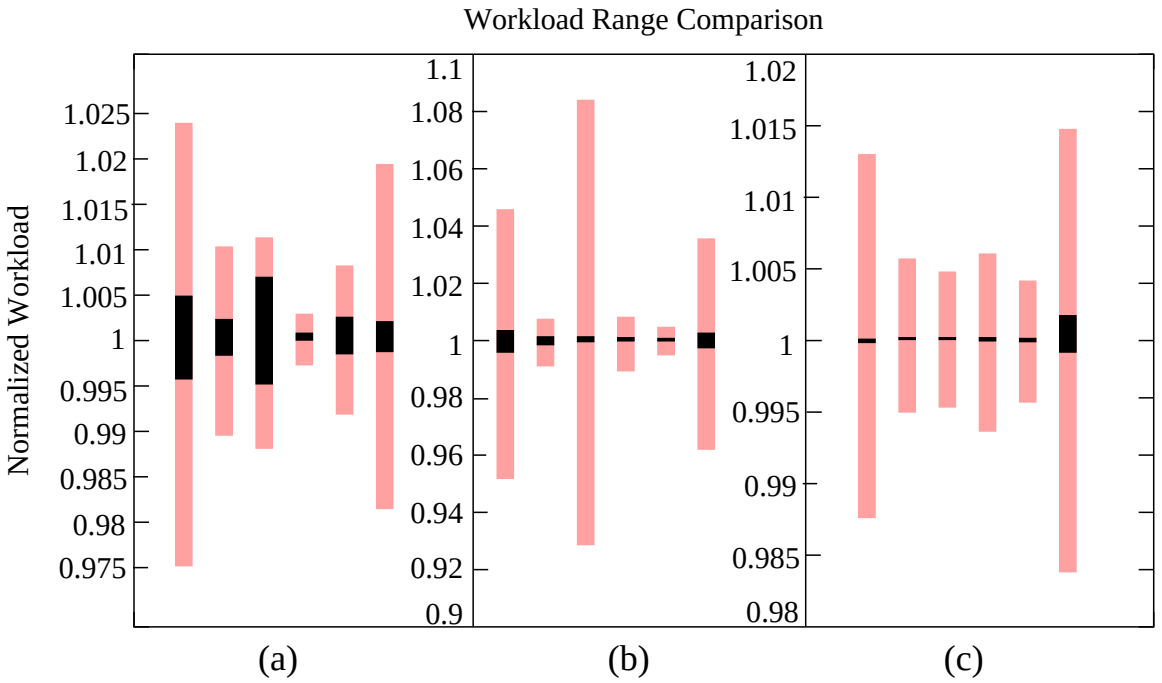
Effect of Chunk Size

N_b: number of evaluations per task.

A better load balancing is achieved when $N_b = 1$. $N_b > 1$ should only be used to stack cheap function evaluations.



Worker Workload Range Modelling



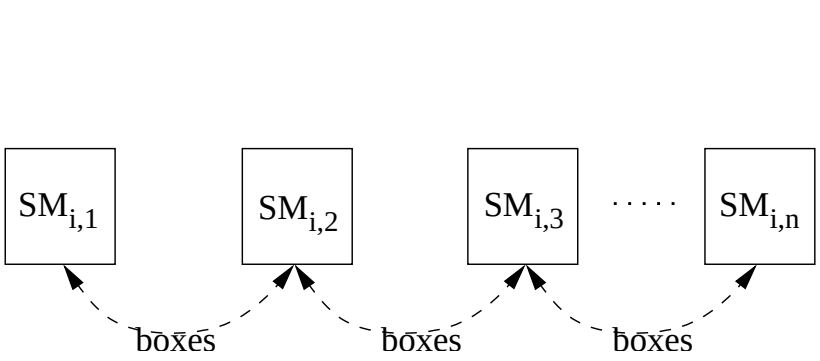
(a) $m = 1$, $n = 1$, $k = 99$; (b) $m = 1$, $n = 4$, $k = 196$; (c) $m = 4$, $n = 1$, $k = 196$ (pink bars are the model estimations and black ones are the experimental measurements).

A bounding workload model estimates the workload range (WL_l, WL_u) given m, n, k, N_b, T_f, $I_{\max}$, and $\{F_i\}$ (a sequence of function evaluation numbers, $i = 1, 2, \ldots, I_{\max}$). All measurements are within the range of the model estimations.

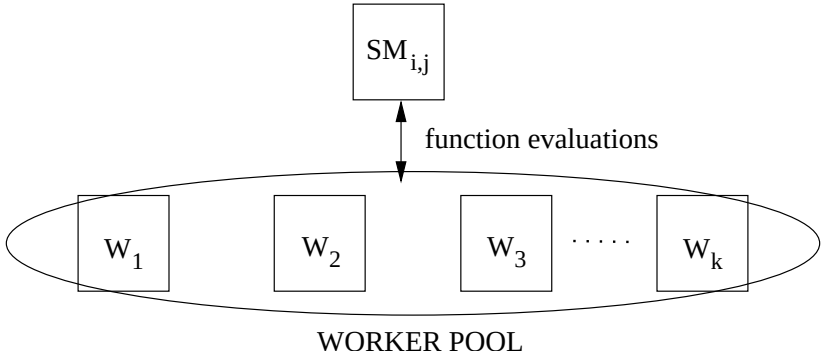
The randomness of workers' requests to masters results in a better balanced workload than the worst case estimated by the model.

Extreme Parallel Schemes

n: number of masters per subdomain; k: number of workers.



Horizontal Scheme



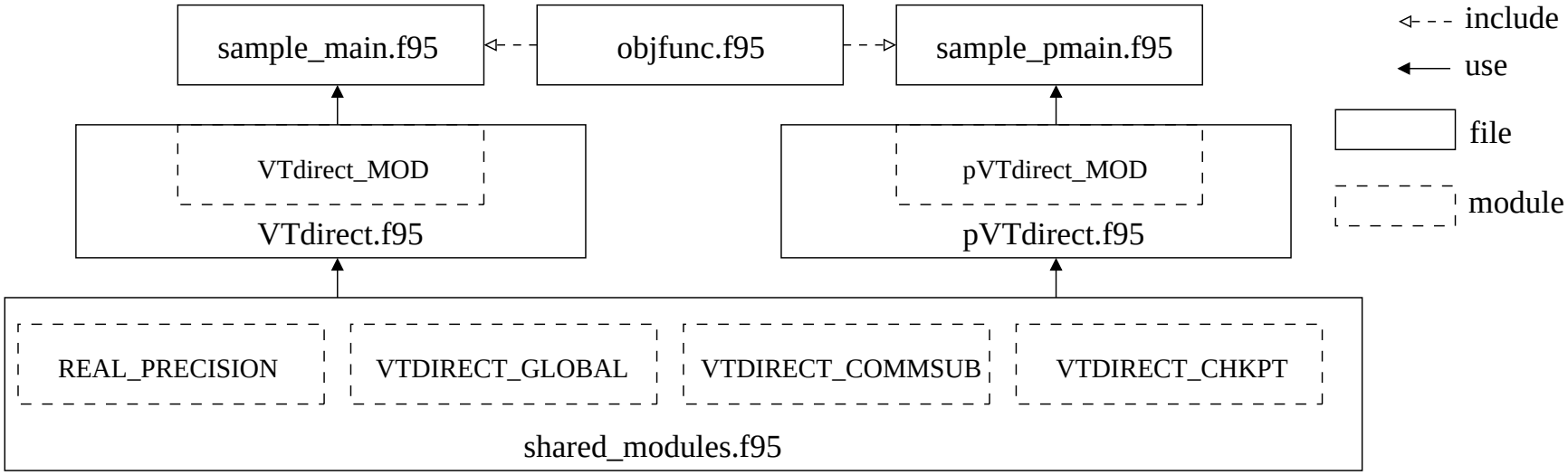
Vertical Scheme

n affects the efficiency of SELECTION and data distribution, but has little impact on the workload balance among workers under the combined schemes. The purpose of using $n > 1$ masters per subdomain is to share the memory burden.

$k > 2mn$ workers should only be used for expensive objective functions ($T_f > T_{cp}$). Workload of function evaluations is better balanced in the vertical scheme with workers than in the horizontal scheme without workers.

Package VTDIRECT95

- Code & usage documentation.
- Portable distribution and user-configurable installation.
- Verification and test suites.



Checkpointing Overhead

10^5 evaluations, $T_f = 0.0$,
nc: no checkpointing,
sv: saving, r: recovery

Serial

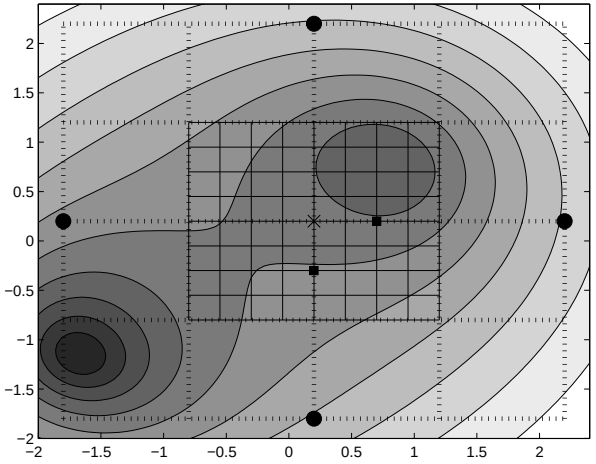
#	I	T_{nc}	T_{sv}	T_r
GR	3057	10.58	18.85	11.50
QU	12238	56.70	87.38	57.28
RO	1198	8.12	11.61	9.22
SC	3637	11.97	21.43	12.96
MI	1968	29.69	34.61	24.05

Parallel

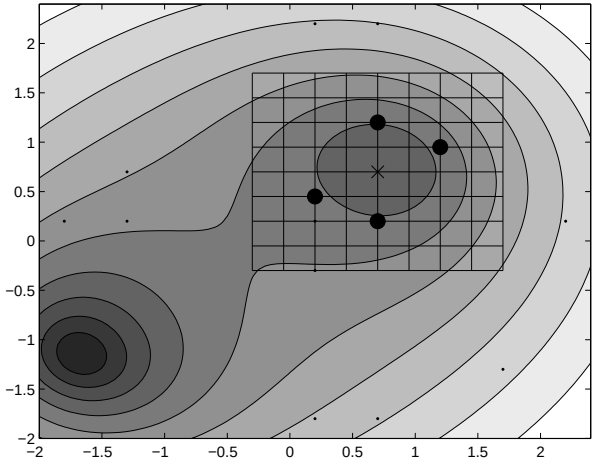
#	T_{nc}	T_{sv}	$T_r(1)$	$T_r(2)$	$T_r(3)$	$T_r(4)$	$T_r(5)$	$T_r(7)$	$T_r(8)$
GR m1	13.22	21.06	14.43	26.11	–	22.76	–	–	27.86
GR m3	11.95	21.07	–	–	12.71	–	23.49	27.24	–
QU m1	55.87	109.15	57.42	76.27	–	86.51	–	–	104.47
QU m3	68.58	83.04	–	–	49.52	–	95.78	107.46	–
RO m1	9.33	12.80	10.70	16.89	–	13.74	–	–	14.50
RO m3	6.61	10.47	–	–	7.02	–	13.28	13.89	–
SC m1	14.44	23.38	15.37	30.05	–	26.11	–	–	30.39
SC m3	14.49	25.59	–	–	13.68	–	34.40	29.87	–
MI m1	30.31	35.58	23.30	21.57	–	18.87	–	–	20.68
MI m3	14.40	17.84	–	–	11.74	–	18.28	20.03	–

Mesh Adaptive Direct Search

MADS in action

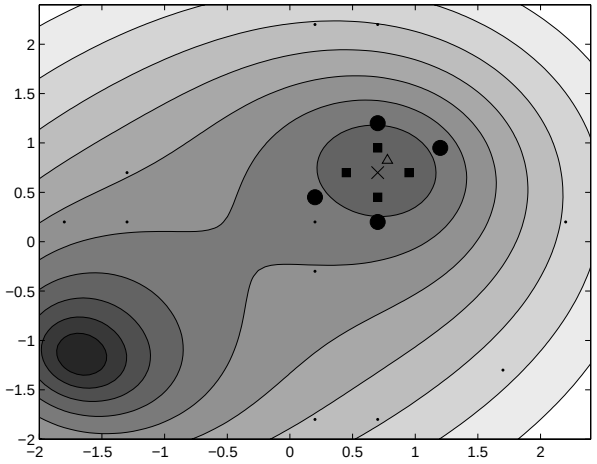


(a) $k = 0$



(b) $k = 5$

- $\times$ x_k
- $\bullet$ trial points in iteration $k + 1$
- $\blacksquare$ trial points in iteration $k + 2$
- $\triangle$ trial points in iteration $k + 3$
- $\cdot$ points in S_k



(c) $k = 5$

Mesh Adaptive Direct Search
Algorithm description

Given $\Omega \in \mathcal{R}^n$, $f : \mathcal{R}^n \to \mathcal{R}$, $f_\Omega(x) = \begin{cases} f(x), x \in \Omega \\ \infty, \text{otherwise} \end{cases}$, mesh size $\Delta^m > 0$, poll size $\Delta^p > 0$, $x_0 \in \Omega$, initial mesh, $k := 0$.

Step 1. **(search step)** Evaluate f_Ω at a (possibly empty) set of points on the mesh.

Step 2. **(poll step)** Evaluate f_Ω at all points in a frame (subset of current mesh) centered at the current best point x_k.

Step 3. **(adapt)** Adjust the mesh and poll sizes and extend the mesh.

Step 4. If an appropriate stopping criterion has been met, stop. Otherwise, set $k := k + 1$ and go back to Step 1.

Mesh Adaptive Direct Search
Definitions

S_k: the set of all points at which f_Ω was evaluated before iteration k.

D: an $n \times n_D$ matrix, where each column $D_{\cdot j} = Gz_j$ (for $j = 1, 2, \ldots, n_D$) for some fixed nonsingular generating matrix $G \in \mathcal{R}^{n \times n}$ and nonzero integer vector $z_j \in \mathcal{Z}^n$. The columns of D must also be a positive spanning set, Pos$(D) = \mathcal{R}^n$.

M_k: the mesh $M_k = \bigcup_{x \in S_k} \{x + \Delta_k^m Dz : z \in \mathcal{N}^{n_D}\}$.

D_k: positive spanning set derived from the columns of matrix D.

P_k: the frame $P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k$ determined by poll size.

Mesh Adaptive Direct Search
Detailed algorithm description

Step 1. Let $x_0 \in \Omega$ and $0 < \Delta_0^m \leq \Delta_0^p$. Let D be an $n \times n_D$ matrix with the properties described earlier. Set the iteration counter $k := 0$.

Step 2. Perform the search step. This step varies among the individual algorithms; in all algorithms f_Ω is evaluated at a finite subset of points (called trial points) on the mesh M_k. If a trial point y is found such that $f_\Omega(y) < f_\Omega(x_k)$, then the algorithm may go to Step 4 with $x_{k+1} := y$.

Step 3. Perform the poll step, evaluating f_Ω at points from the frame $P_k \subset M_k$ until a frame point x_{k+1} is found with $f_\Omega(x_{k+1}) < f_\Omega(x_k)$ or f_Ω has been evaluated at all of the points in P_k.

Step 4. Update Δ_{k+1}^m and Δ_{k+1}^p according to the specific algorithm's rules. In all algorithms,

(1) Δ_{k+1}^m is greater than or equal to Δ_k^m if an improved mesh point is found,

(2) Δ_{k+1}^m is less than Δ_k^m if an improved mesh point is not found,

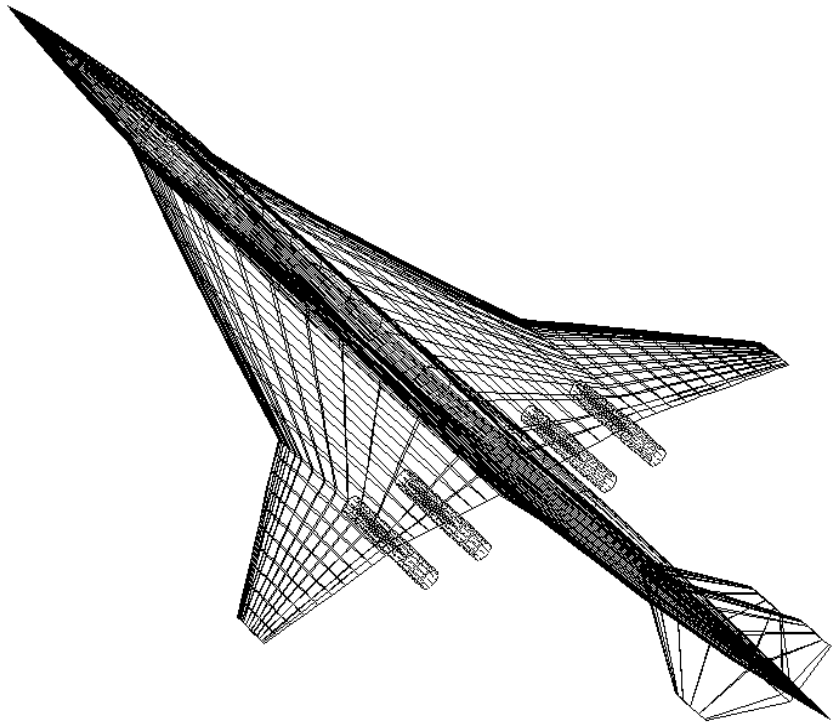
(3) Δ_{k+1}^p is greater than or equal to Δ_{k+1}^m, and

(4) $\liminf_{j\to\infty} \Delta_j^m = 0$ if and only if $\liminf_{j\to\infty} \Delta_j^p = 0$.

Step 5. If an appropriate stopping criterion has been met, stop. Otherwise, set $k := k + 1$ and go back to Step 2.

Application in Aircraft Design: HSCT
Problem scenario

Optimization objective: minimize takeoff gross weight (TOGW) for a range of 5500 nautical miles and a cruise Mach number of 2.4, while carrying 251 passengers.



Typical high speed civil transport (HSCT) configuration.

Application in Aircraft Design: HSCT

Optimization design variables and constraints

1. **28 design variables:**

- Geometry of the aircraft, 24 variables in 6 categories:

  wing planform,
  airfoil shape,
  tail areas,
  nacelle placement,
  and fuselage shape.

- Idealized cruise mission, 4 variables:

  mission fuel,
  engine thrust,
  initial cruise altitude,
  and constant climb rate.

2. **68 constraints in 3 categories:**

- Geometry
- Performance
- Aerodynamic

Application in Aircraft Design: HSCT
Problem formulation

1. **Issue #1 — convergence determination.**

   For this study, the algorithm was run for a fixed number of loops or iterations. Since the purpose of the optimization was to identify promising regions of the design space, it was unnecessary to tightly converge to a global optimum.

2. **Issue #2 — incorporation of constraints.**

   Constraints were accounted for through the use of a simple penalty function, as follows. Let x be the 28-dimensional design vector, $f(x)$ the TOGW, and $g_i(x) \leq 0$ the constraints. The constrained optimization problem

$$\min f(x) \quad \text{subject to } g_i(x) \leq 0,\ i = 1, ..., 68,$$

   is converted to the unconstrained optimization problem

$$\min f(x) + 10 \sum_{i=1}^{68} \max\{0, g_i(x)\}.$$

Application in Aircraft Design: HSCT
Parallel DIRECT

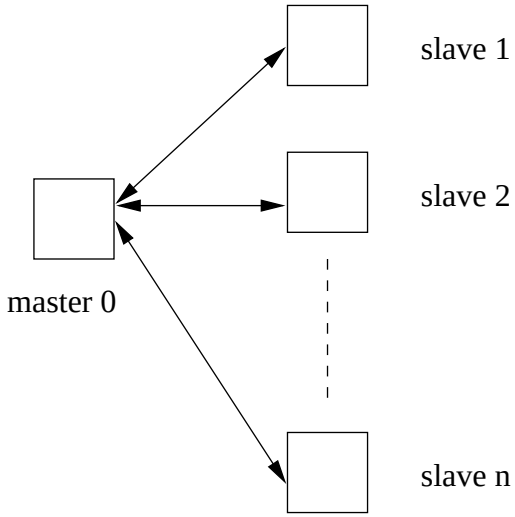
1. **Challenges**
   - Load balancing: moving from static to dynamic.
   - Communication bottleneck: adapting bin size.

2. **Parallel schemes**
   - Master-slave

     STATIC: static load balancing.

     DLBMS01: dynamic load balancing with bin size 1.

     DLBMS10: dynamic load balancing with bin size 10.

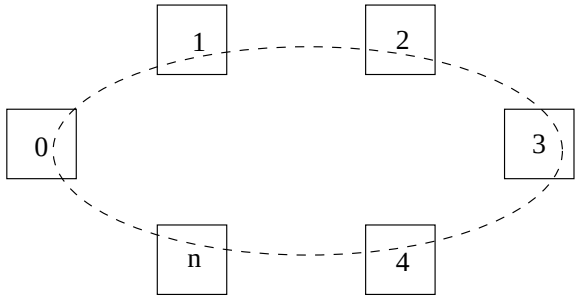


Application in Aircraft Design: HSCT
Parallel DIRECT (cont.)

- Distributed control

  DLBDC: dynamic load balancing with fully distributed control.

  DLBDCT: dynamic load balancing with fully distributed control using pthreads.

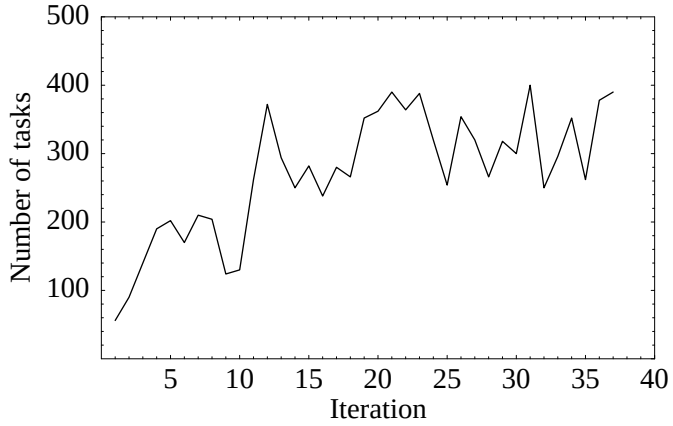


- Aggressive DIRECT

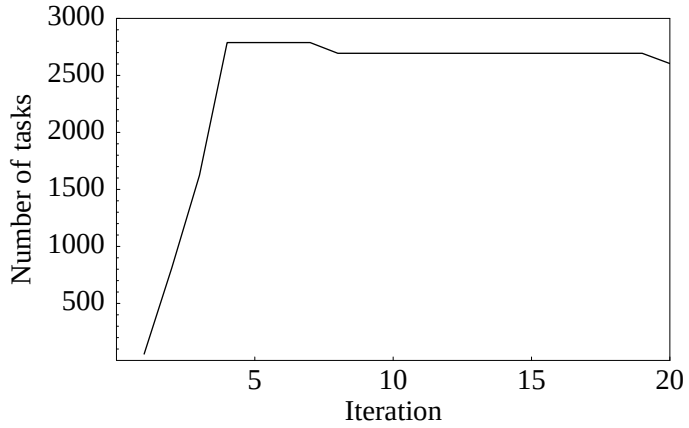
  Discard the step of identifying potentially optimal box set. Each box with the smallest objective function value for that box size is considered “potentially optimal”. This results in a much larger set of new tasks to be evaluated and load balanced at each iteration.

Application in Aircraft Design: HSCT

Task history comparison



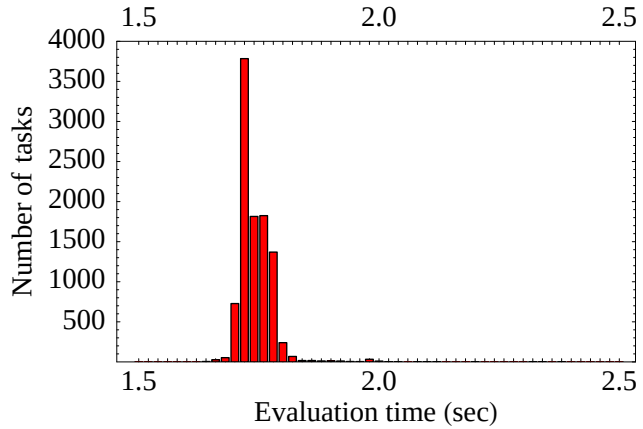
Original DIRECT.



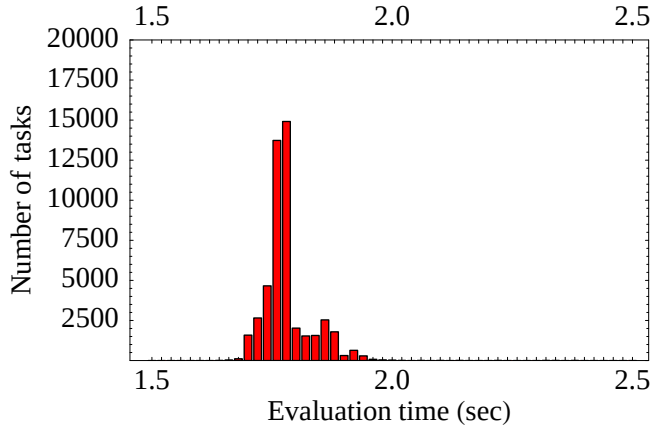
Aggressive DIRECT.

Application in Aircraft Design: HSCT

Time distribution comparison



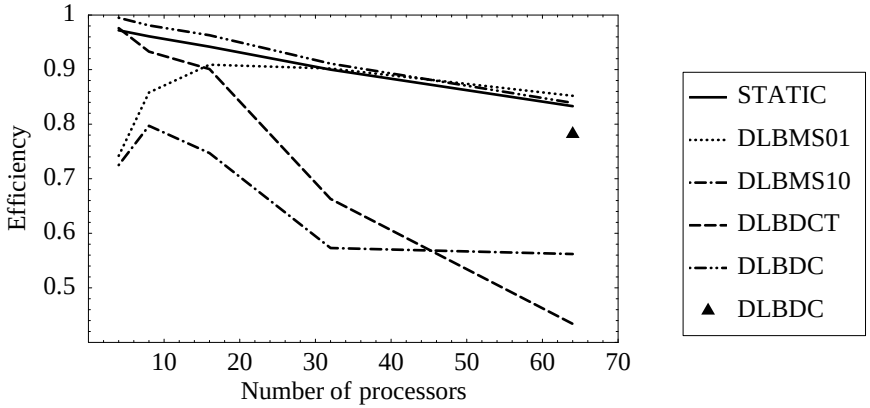
Original DIRECT.



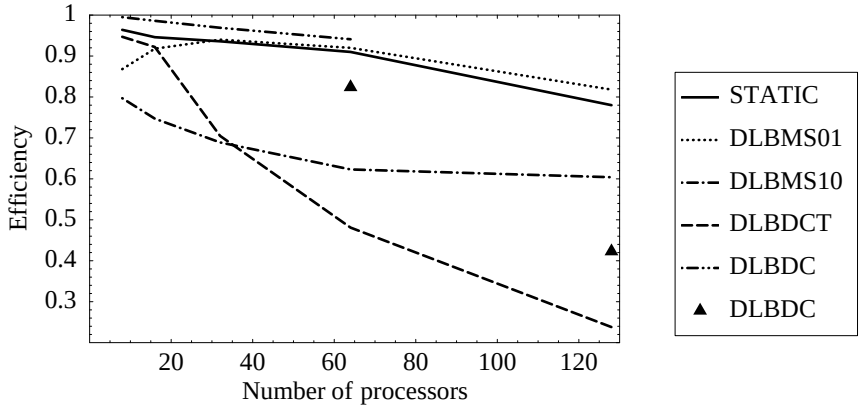
Aggressive DIRECT.

Application in Aircraft Design: HSCT

Parallel efficiency comparison



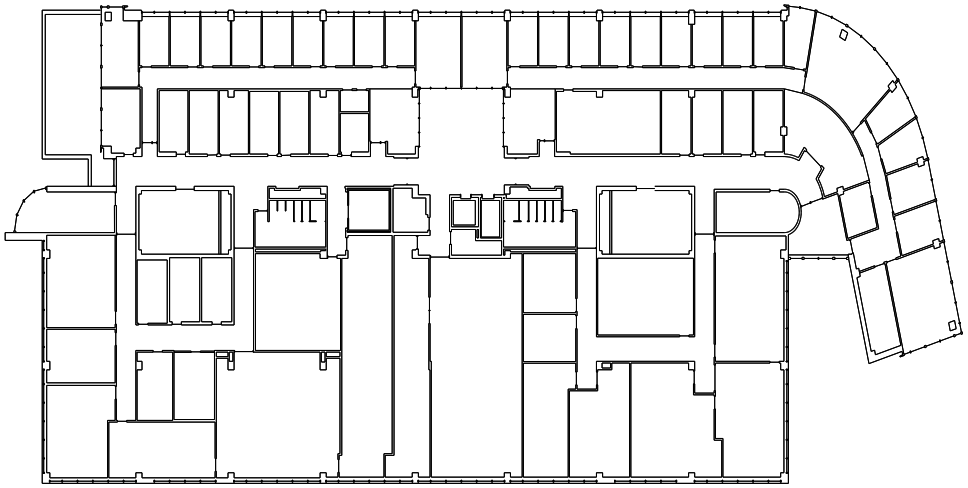
Original DIRECT.



Aggressive DIRECT.

Application in Wireless Design: S^4W
Problem scenario

1. Transmitter placement optimization: ensuring an acceptable level (threshold) of wireless system performance within a geographical area of interest at a minimum cost.



Durham Hall 4th floor, Virginia Tech

2. Problem abstraction:

$$\min_{x \in D} f_0(x),$$

$$D = \{x \in D_0 \mid f_j(x) \leq 0,\ j = 1, \ldots, J\},$$

where $D_0 = \left\{x \in E^n \mid \ell \leq x \leq u\right\}$ is a simple box constraint set.

Application in Wireless Design: S^{4}W
Objective formulation

1. **Power coverage:**

$$\frac{\text{Number of receivers with received power above threshold}}{\text{Total number of receivers}}$$

2. **Bit error rate (BER):**

$$\frac{\text{Number of incorrectly received bits}}{\text{Total Number of received bits}}$$

3. **Observation:** Discrete vs. continuous.

4. **Reformulation:**

- **Decision variables for n transmitters over m receivers:**

$$X = (x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n),$$

where $z_i = z_0$.

Application in Wireless Design: S^{4}W

Objective formulation (cont.)

- **Objective function**

  Average shortfall of the estimated performance metric from the given threshold T:

$$f(X) = \begin{cases} \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(T - p_{ki})_+, & \text{coverage,} \\ \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(p_{ki} - T)_+, & \text{BER.} \end{cases}$$

  p_{ki}: performance metric of transmitter (k,i) evaluated at the ith receiver location, where transmitter (k, i), located at $(x_k, y_k, z_0),\ 1 \le k \le n$, generates the highest power level $P_{ki}(x_k, y_k, z_0) \ge P_{ji}(x_j, y_j, z_0),\ 1 \le j \le n$, at the receiver location $i,\ 1 \le i \le m$.

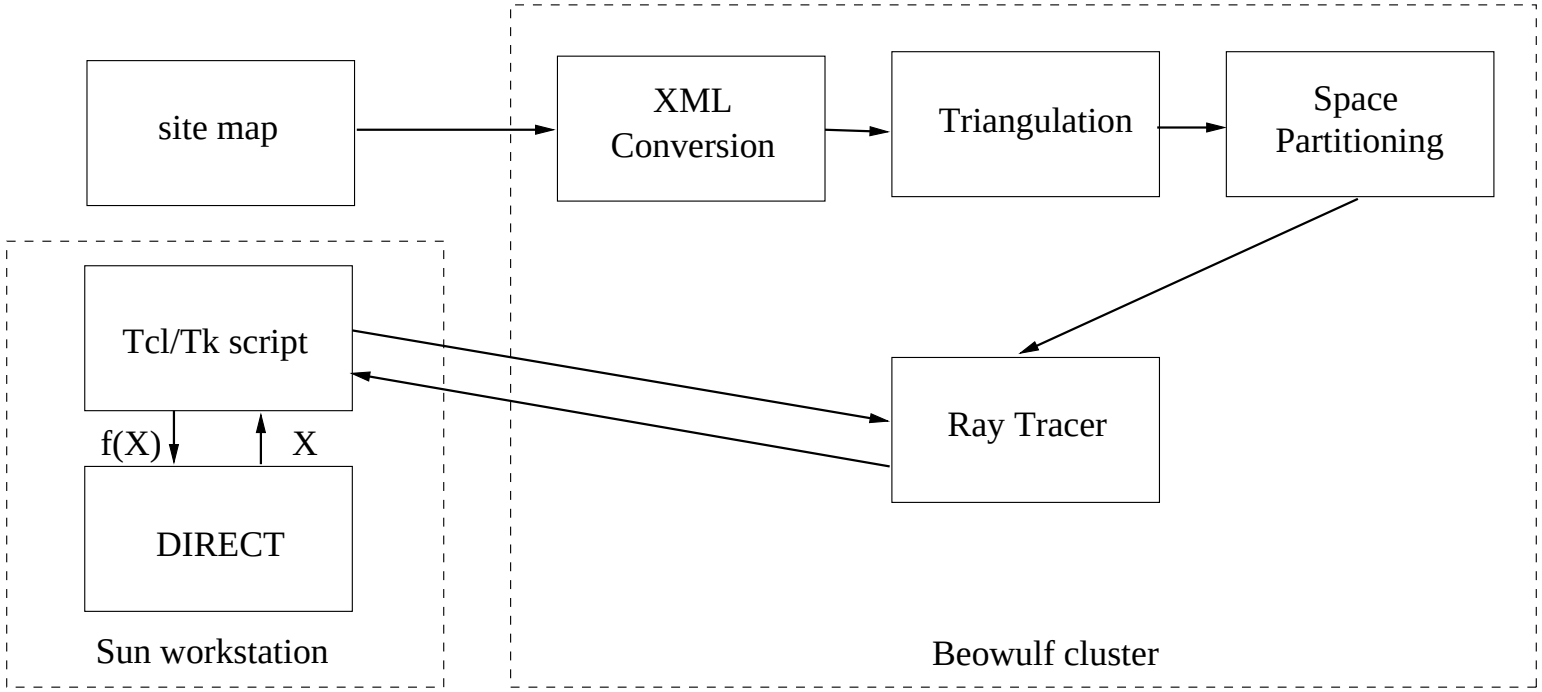
  Power coverage optimization:

  $p_{ki} = P_{ki}(x_k, y_k, z_0), (T - p_{ki})_+$ is the penalty for a low power level.

  BER optimization:

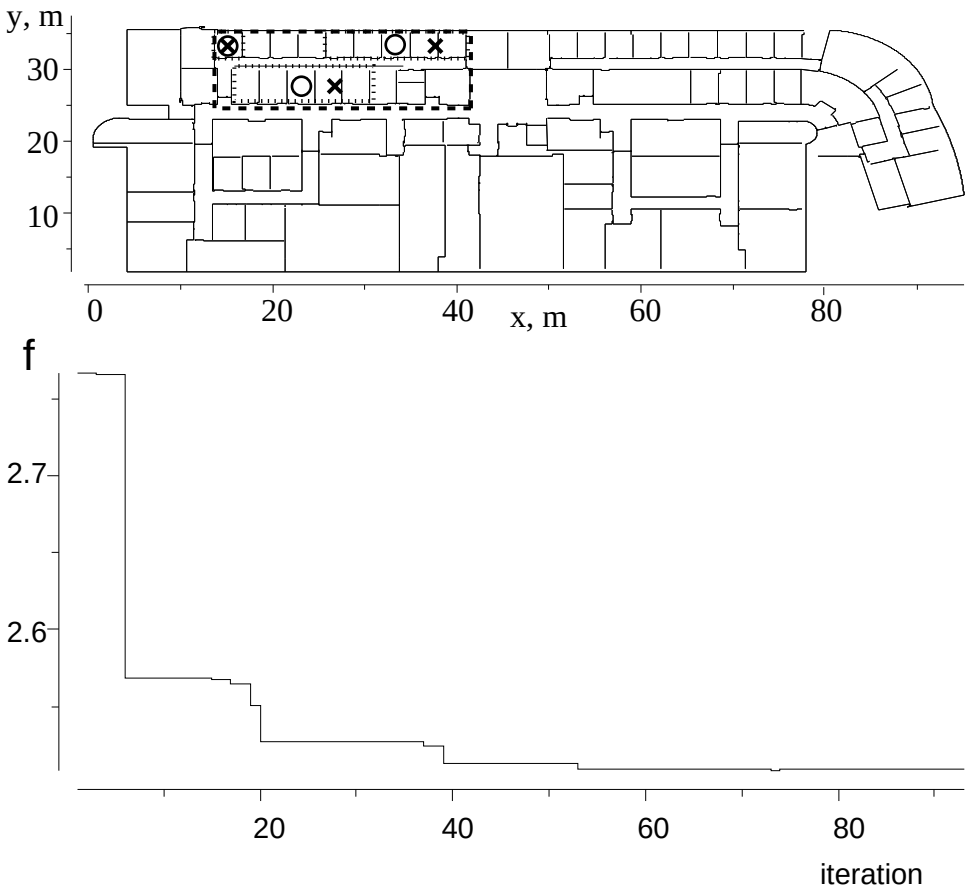
  $p_{ki} = \log_{10}(\mathrm{BER}_{ki}), (p_{ki} - T)_+$ is the penalty for a high bit error rate.

Application in Wireless Design: S^4W
Problem solving environment



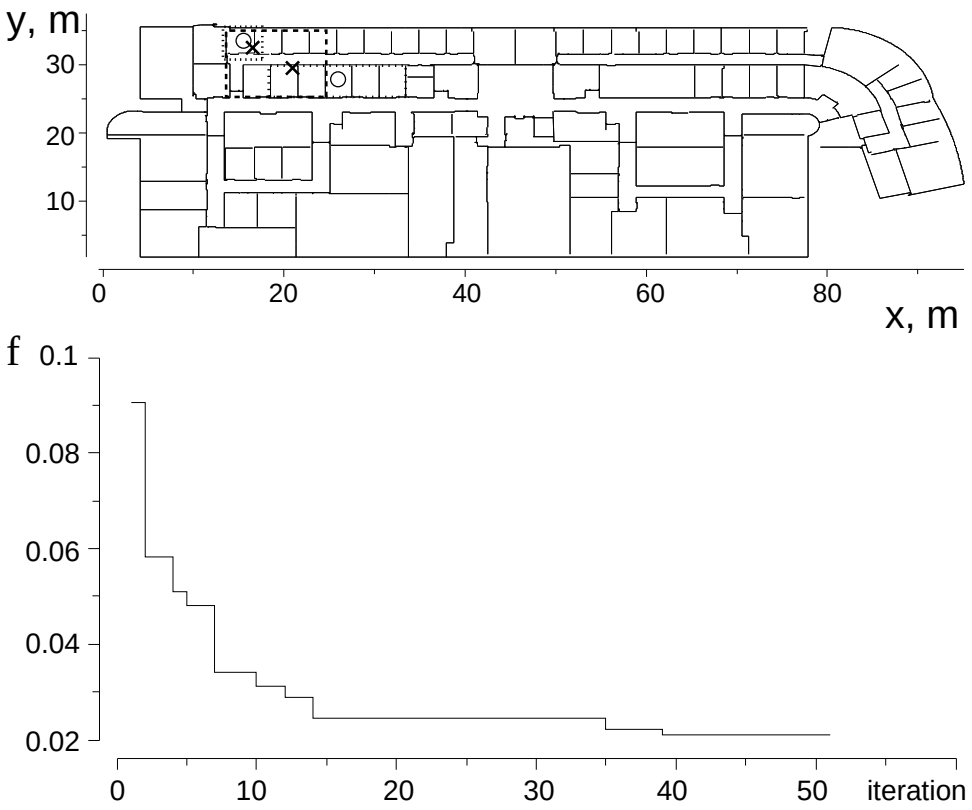
- Problem decomposition
- Parallel processing
- Interprocess communication
- Surrogate modeling

Application in Wireless Design: S^4W
Optimization results



Power coverage optimization results for three transmitters. The starting (optimal) locations are marked with circles (crosses).

Application in Wireless Design: S^4W
Optimization results



BER optimization results for two transmitters. The starting (optimal) locations are marked with circles (crosses).

Budding Yeast Cell Cycle

The problem considered here is a model of the cell cycle of budding yeast (also known as brewer's or baker's yeast). In the cell cycle, the yeast cell must perform various actions (checkpoints) in the proper order (e.g., the chromosomes must be duplicated before they can be aligned, and the chromosomes must be aligned before they can be separated).

The model.

The model of the cell cycle consists of 36 ODEs with 143 regression parameters. The checkpoints are modeled by a variable rising through a threshold value.

The experimental data.

Biologists over or under express one or more genes in the cell, and then observe how the cell proceeds.

The biologists can observe the size at which the cell divides, and the time it takes to pass certain checkpoints.

If the cell dies, the biologists can determine if it had passed certain checkpoints or not.

The information for each experiment can then be condensed into a six-tuple (v, g, m, a, t, c)—viability, G1 length, mass at division, arrest stage, arrest type, number of cell cycles completed.

There are about 120 experiments applicable to this model.

In the model, the checkpoints are indicated by a variable rising through a threshold value. In this way, the ODE solutions are converted to a six-tuple that can be compared to the experimental data.

The objective function.

$$R(O,P) = \begin{cases} \omega_g \times \left(\frac{O_g - P_g}{\sigma_g}\right)^2 + \omega_m \times \left(\frac{\ln \frac{O_m}{P_m}}{\sigma_m}\right)^2, & O_v = P_v = \text{viable}, \\ \omega_v \times \frac{1}{1+P_c}, & O_v = \text{viable}, P_v = \text{inviable}, \\ \delta_{O,P} + \omega_c \times \left(\frac{O_c - P_c}{\sigma_c}\right)^2, & O_v = P_v = \text{inviable}, \\ \omega_v \times \frac{1}{1+O_c}, & O_v = \text{inviable}, P_v = \text{viable}, \end{cases}$$

where the ωs and σs are weighting constants, and δ is a real valued discrete function, used to assess a penalty for the arrest stage and type, given by

$$\delta_{O,P} = \begin{cases} \omega_a, & \text{if } O_a \neq P_a, \\ \omega_t, & \text{if } O_a = P_a \text{ and } O_t \neq P_t, \\ 0, & \text{if } O_a = P_a \text{ and } O_t = P_t. \end{cases}$$

Parallel VTDIRECT results.

The center of the hyperbox was the biologist's best point. In most dimensions, the bounds were 100-fold of the biologist's best point. A few of the dimensions were more tightly constrained because the respective parameters were known with more accuracy.

Ran for 40 hours on 512 processors (equivalent to 20,000 hours on one processor).

Evaluated 1,500,000 points over 813 iterations.

The biologist's best point was scored at 433; the point found by VTDIRECT scored 212.

Parallel MADS results.

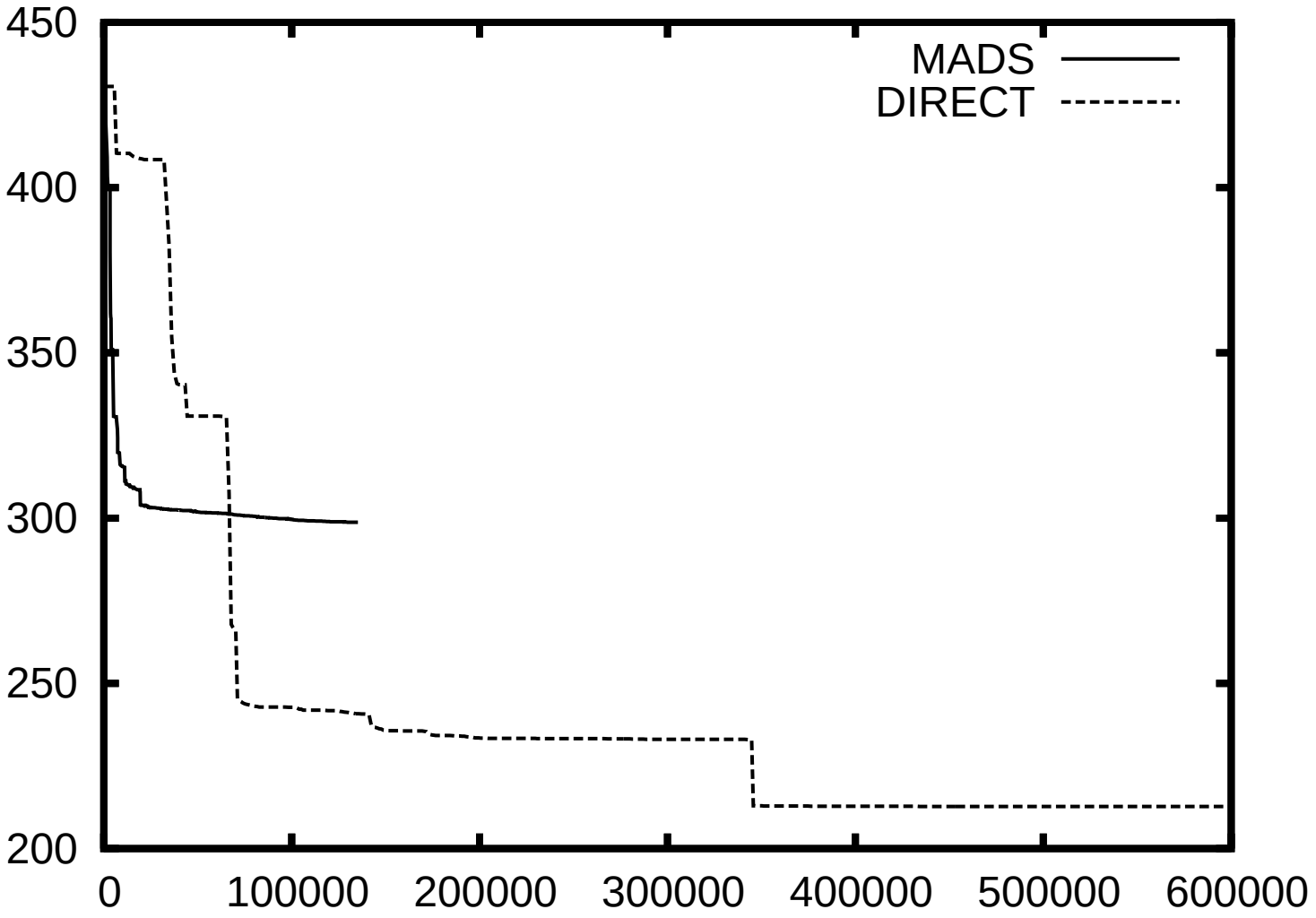
Used the same bounding box as VTDIRECT, with the starting point set to the biologist's best point.

Ran for 6 hours on 64 processors (equivalent to 400 hours on one processor).

Evalated 135,000 points over 473 iterations.

The best point found by MADS scored 299.

Best point vs. number of evaluations



Conclusion

1. **Effectiveness of DIRECT**

   The DIRECT algorithm solves global optimization problems effectively, especially for continuous unconstrained problems. However, future work is needed for solving discrete problems and nonlinear constrained problems.

2. **Dynamic implementation**

   Addresses the issues of
   - data structure extensibility,
   - memory allocation efficiency,
   - and computation simplicity.

3. **Parallel implementation**

   Massively parallel code features
   - load balancing via distributed data,
   - flexible schemes for different cost functions,
   - multiple termination criteria,
   - output multiple best boxes with centers separated by MIN_SEP,
   - robust checkpointing for hot restarts.

4. **Future research**
   - Petascale versions of VTDIRECT and MADS.
   - Hybrid algorithm combining the global features of VTDIRECT with the local efficiency of MADS.

References

1. N. A. Allen, C. A. Shaffer, M. T. Vass, N. Ramakrishnan, and L. T. Watson, "Improving the development process for eukaryotic cell cycle models with a modeling support environment", *Simulation*, vol. 79, pp. 674–688, 2003.
2. C. Audet and J. E. Dennis, Jr., "Mesh adaptive direct search algorithms for constrained optimization", *SIAM Journal on Optimization*, vol. 17(1), pp. 188–217, 2006.
3. J. E. Dennis and V. Torczon, "Direct search methods on parallel machines", *SIAM Journal on Optimization*, vol. 1, pp. 448–474, 1991.
4. J. He, A. Verstak, L. T. Watson, C. A. Stinson, N. Ramakrishnan, C. A. Shaffer, T. S. Rappaport, C. R. Anderson, K. Bae, J. Jiang, and W. H. Tranter, "Globally optimal transmitter placement for indoor wireless communication systems", *IEEE Transactions on Wireless Communications*, vol. 3, pp. 1906–1911, 2004.
5. J. He, A. Verstak, L. T. Watson, and M. Sosonkina, "Design and implementation of a massively parallel version of DIRECT", *Computational Optimization and Applications*, to appear, also Technical Report TR-06-02, Dept. of Computer Science, VPI&SU, Blacksburg, VA, 2006.
6. J. He, A. Verstak, L. T. Watson, and M. Sosonkina, "Performance modeling and analysis of a massively parallel DIRECT—Part 1", Technical Report TR-07-01, Dept. of Computer Science, VPI&SU, Blacksburg, VA, 2007.

References

7. J. He, A. Verstak, M. Sosonkina, and L. T. Watson, “Performance modeling and analysis of a massively parallel DIRECT—Part 2”, Technical Report TR-07-02, Dept. of Computer Science, VPI&SU, Blacksburg, VA, 2007.
8. 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.
9. J. He, L. T. Watson, and M. Sosonkina, “VTDIRECT95: serial and parallel codes for the global optimization algorithm DIRECT”, Technical Report TR-07-33, Dept. of Computer Science, VPI&SU, Blacksburg, VA, 2007.
10. D. R. Jones, C. D. Perttunen, and B. E. Stuckman, “Lipschitzian optimization without the Lipschitz constant”, *Journal of Optimization Theory and Application*, vol. 79, no. 1, pp. 157–181, 1993.
11. R. M. Lewis, V. Torczon, and M. W. Trosset, “Direct search methods: then and now”, *Journal of Computational and Applied Mathematics*, vol. 124, pp. 191–207, 2000.
12. L. T. Watson and C. A. Baker, “A fully-distributed parallel global search algorithm”, *Engineering Computations*, vol. 18, no. 1/2, pp. 155–169, 2001.
13. Tutorial URL: http://www.cs.vt.edu/~ltw/lecture_notes/HPCS08.4up.pdf .
14. VTDIRECT95 URL: http://www.cs.vt.edu/~ltw/VTDIRECT95.tgz .