

Traditional methods

MTH8418

S. Le Digabel, Polytechnique Montréal

Winter 2020

(v3)

Introduction

- ▶ We consider the unconstrained version of the problem with $\Omega = \mathbb{R}^n$:

$$\min_{x \in \mathbb{R}^n} f(x)$$

- ▶ Possible to treat constrained problems with meta-strategies such as the **extreme barrier** or the **augmented Lagrangian** approaches (discussed in [Lecture #9](#))
- ▶ The “traditional” methods were introduced as heuristics. For some of them, convergence was studied only in the 90’. This opened the era of modern DFO methods
- ▶ This lecture describes three of these methods

Plan

Latin Hypercube Sampling

The Coordinate Search method

The Nelder-Mead method

References

Latin Hypercube Sampling

The Coordinate Search method

The Nelder-Mead method

References

Latin Hypercube Sampling

- ▶ Traditional statistical method from [McKay et al., 1979]
- ▶ Often used in the design of computer experiments
- ▶ **Latin square**: With $n = 2$, a grid containing one sample in each row and each column
- ▶ **Latin hypercube**: Generalization in dimension n
- ▶ Bounds for each variable are necessary
- ▶ **Stochastic** method:
 - ▶ Use a good Random Number Generator (RNG)
 - ▶ Use the process identification number (PID) as the **random seed**
 - ▶ Change the seed for different results
 - ▶ Remember the seed for reproducibility

Latin Hypercube Sampling

- ▶ Optimization:
 1. Construct sampling: Generate p sample points
 2. Evaluate f at the sample points
 3. Remember the best solution

- ▶ Bad algorithm for optimization

- ▶ But good strategy to generate a set of starting solutions for other methods

Latin Hypercube Sampling: Pseudocode

```
vector<Point> LHS ( n , p , lb , ub ) {  
  
    vector<Point> pts;  
  
    for ( k = 0 ; k < p ; ++k ) {  
        Point x(n);  
        for ( i = 0 ; i < n ; ++i )  
            x[i] = lb[i] + ( ub[i]-lb[i] ) *  
                    ( permut(p) + rand(0.0,1.0) ) / p;  
        pts.insert(x);  
    }  
    return pts;  
}
```


Latin Hypercube Sampling

The Coordinate Search method

The Nelder-Mead method

References

The Coordinate Search method

- ▶ **Coordinate Search (CS)**, or **Compass Search**
- ▶ The CS is the ancestor of the modern **direct search** methods:
 - ▶ **Hooke & Jeeves** method: The original **Pattern Search** method (also defined “direct search”) [Hooke and Jeeves, 1961]
 - ▶ **Generalized Pattern Search (GPS)** [Torczon, 1997]
 - ▶ **Mesh Adaptive Direct Search (MADS)** [Audet and Dennis, Jr., 2006]
 - ▶ **Generating Set Search (GSS)** [Kolda et al., 2003]
- ▶ [Fermi and Metropolis, 1952] (first known reference)
- ▶ Tested on one of the first computers, the **MANIAC** (Mathematical Analyzer, Numerical Integrator, and Computer or Mathematical Analyzer, Numerator, Integrator, and Computer)

Coordinate Search: Algorithm

► INITIALIZATION:

x_0 : starting point in \mathbb{R}^n

$\Delta_0 > 0$: initial step size

► POLL STEP: for $k = 0, 1, \dots$

if $f(t) < f(x_k)$ for $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$:

$$x_{k+1} \leftarrow t$$

$$\Delta_{k+1} \leftarrow \Delta_k$$

Coordinate Search: Algorithm

▶ INITIALIZATION:

x_0 : starting point in \mathbb{R}^n

$\Delta_0 > 0$: initial step size

▶ POLL STEP: for $k = 0, 1, \dots$

if $f(t) < f(x_k)$ for $t \in P_k := \{x_k \pm \Delta_k e_i : i = 1, 2, \dots, n\}$:

$$x_{k+1} \leftarrow t$$

$$\Delta_{k+1} \leftarrow \Delta_k$$

else (failure): x_k is a local minimum relatively to P_k :

$$x_{k+1} \leftarrow x_k$$

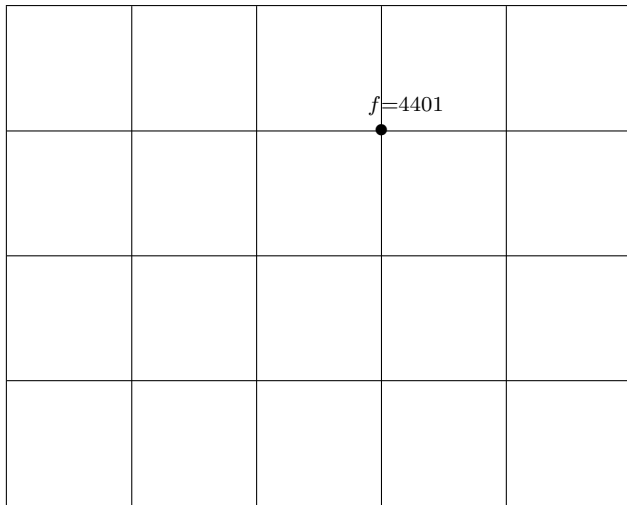
$$\Delta_{k+1} \leftarrow \Delta_k/2$$

Coordinate Search: Details

- ▶ Evaluation of x_0 is counted
- ▶ Extension with bounds: If the algorithm generates a trial point outside of the boundary, then do not evaluate and consider as a failure
- ▶ If the algorithm generates a trial point x found in the **cache**, then do not evaluate and consider as a failure

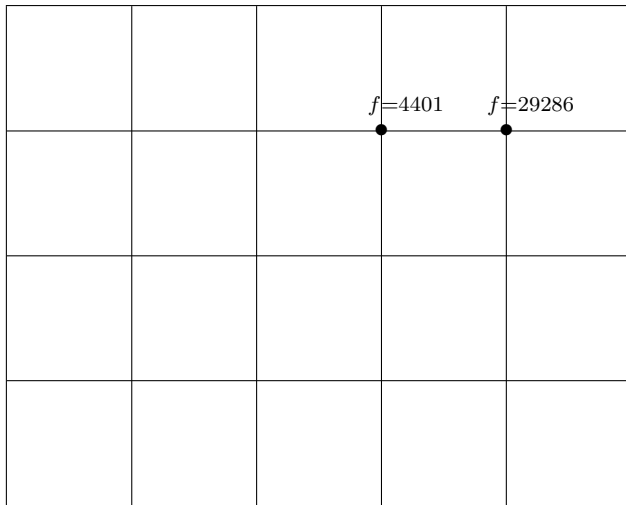
Coordinate Search: Basic algorithm

$$x_0 = (2, 2)^T, \Delta_0 = 1$$



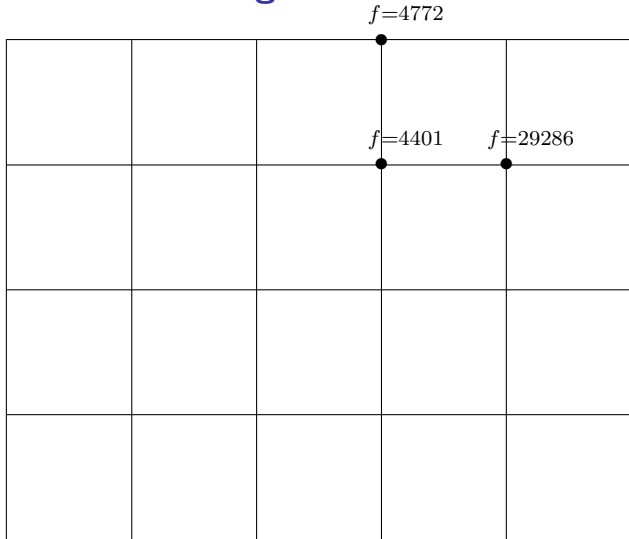
Coordinate Search: Basic algorithm

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



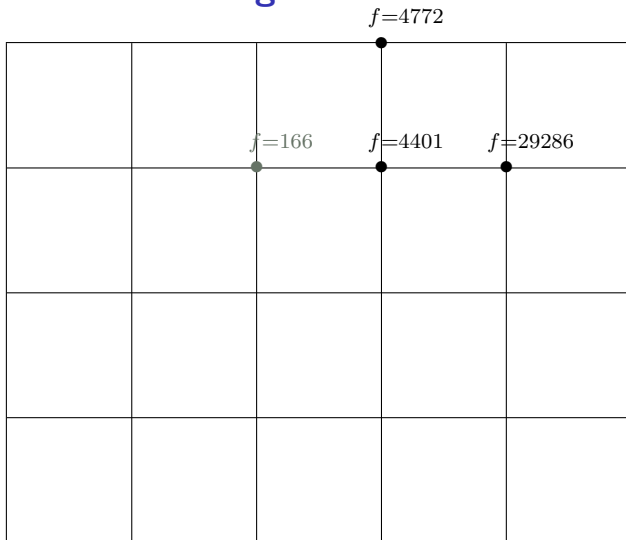
Coordinate Search: Basic algorithm

$x_0 = (2, 2)^\top, \Delta_0 = 1$



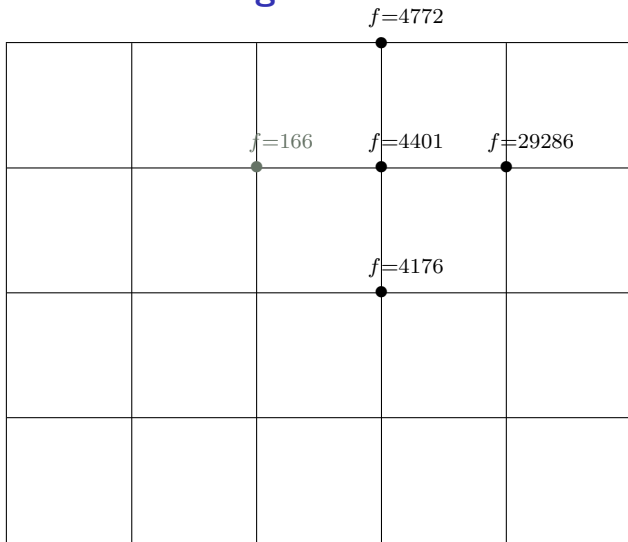
Coordinate Search: Basic algorithm

$$x_0 = (2, 2)^T, \Delta_0 = 1$$



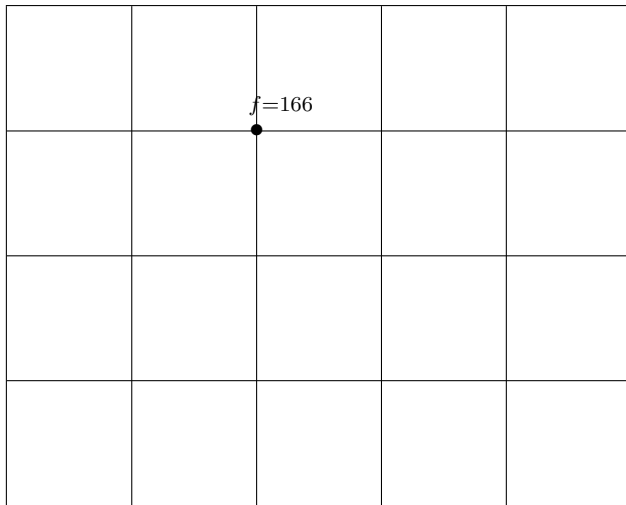
Coordinate Search: Basic algorithm

$$x_0 = (2, 2)^T, \Delta_0 = 1$$



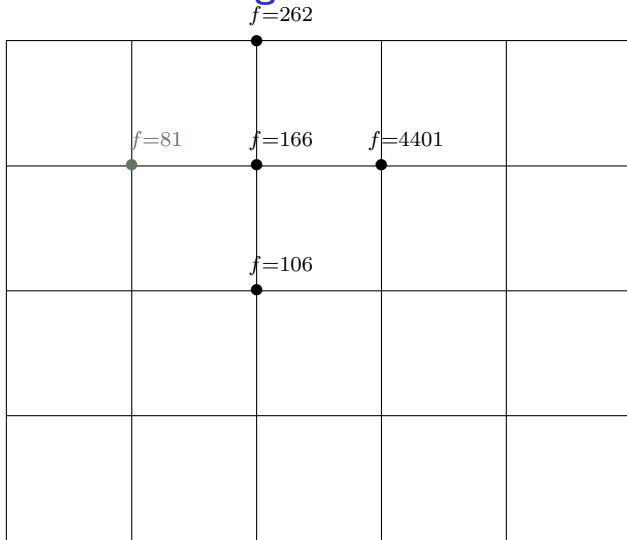
Coordinate Search: Basic algorithm

$$x_1 = (1, 2)^T, \Delta_1 = 1$$



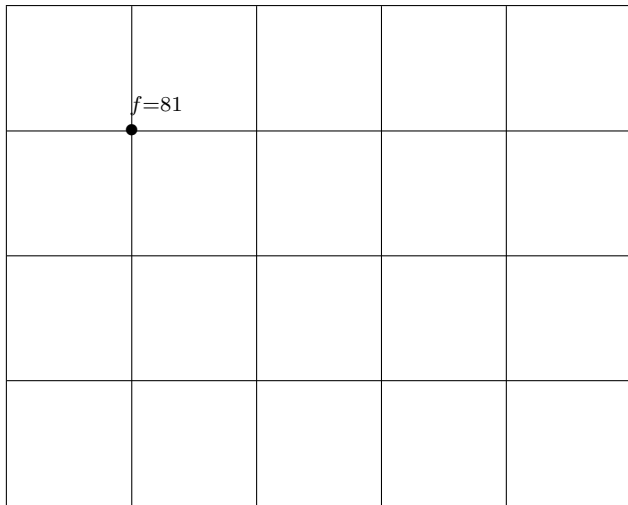
Coordinate Search: Basic algorithm

$$x_1 = (1, 2)^T, \Delta_1 = 1$$

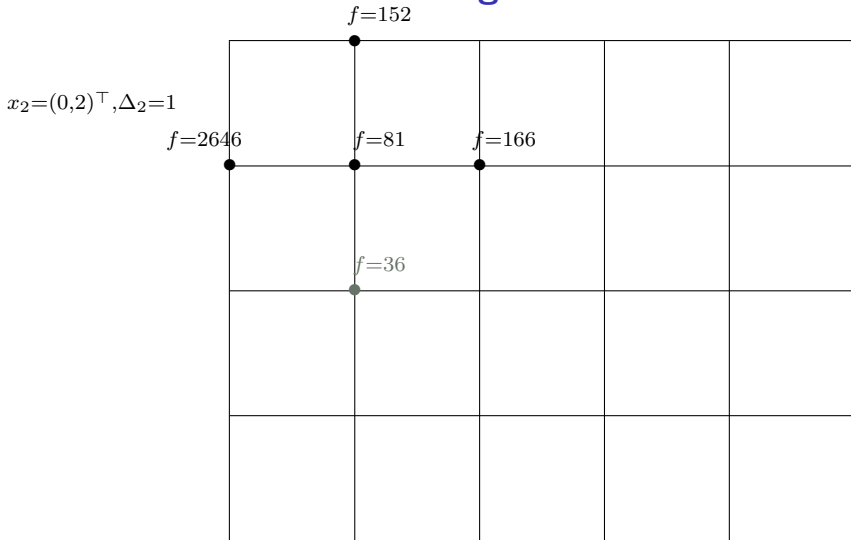


Coordinate Search: Basic algorithm

$$x_2 = (0, 2)^\top, \Delta_2 = 1$$

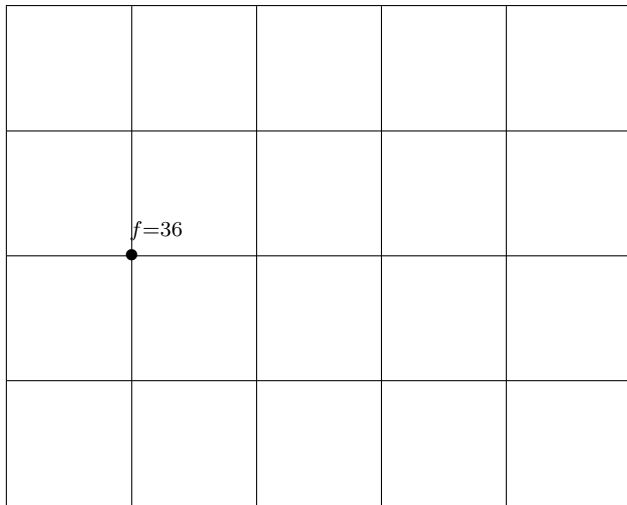


Coordinate Search: Basic algorithm



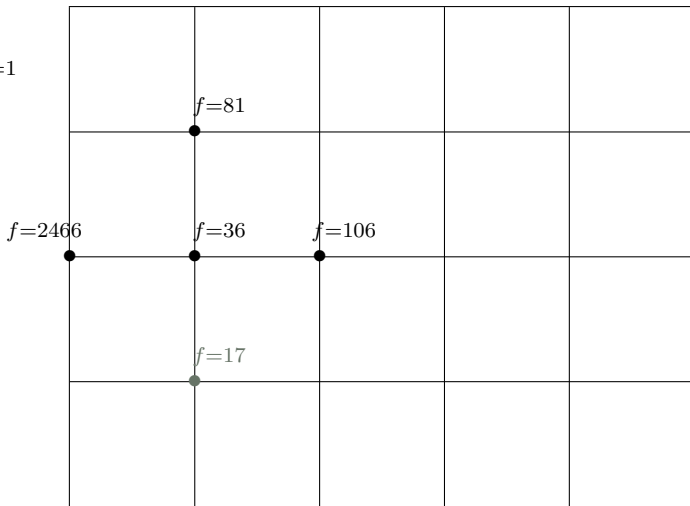
Coordinate Search: Basic algorithm

$$x_3 = (0, 1)^T, \Delta_3 = 1$$



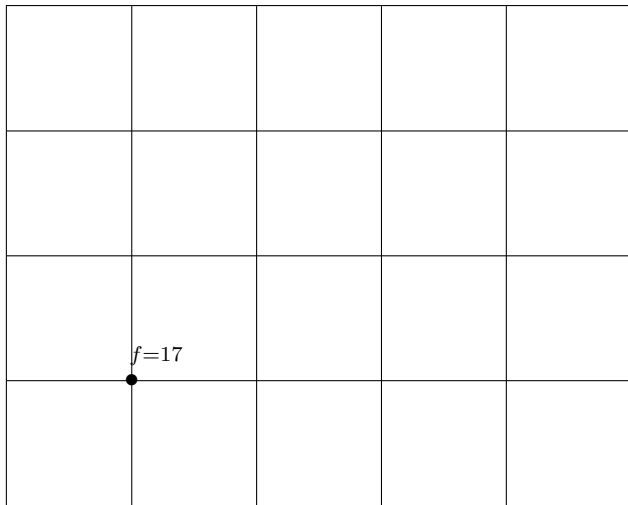
Coordinate Search: Basic algorithm

$$x_3 = (0, 1)^\top, \Delta_3 = 1$$



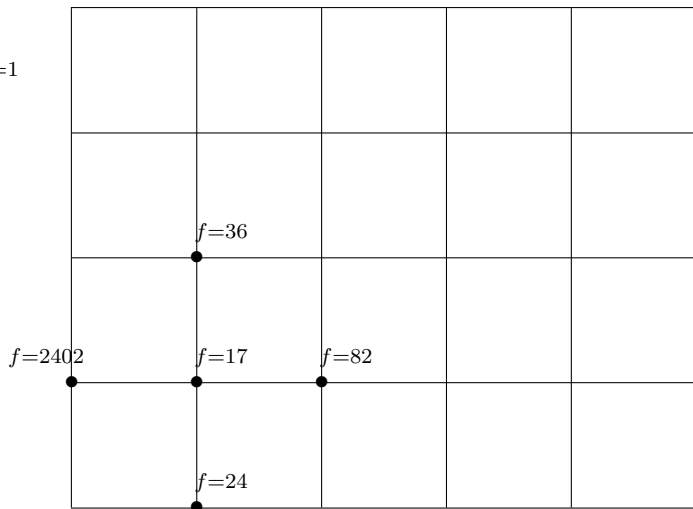
Coordinate Search: Basic algorithm

$$x_4 = (0, 0)^\top, \Delta_4 = 1$$



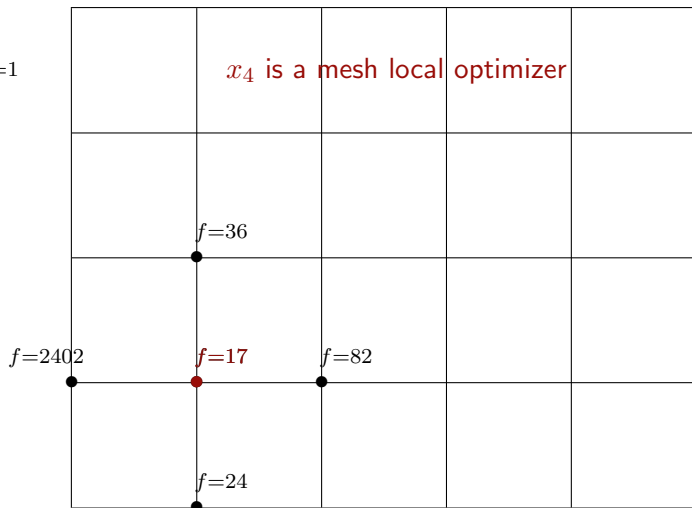
Coordinate Search: Basic algorithm

$$x_4 = (0,0)^\top, \Delta_4 = 1$$



Coordinate Search: Basic algorithm

$$x_4 = (0,0)^\top, \Delta_4 = 1$$



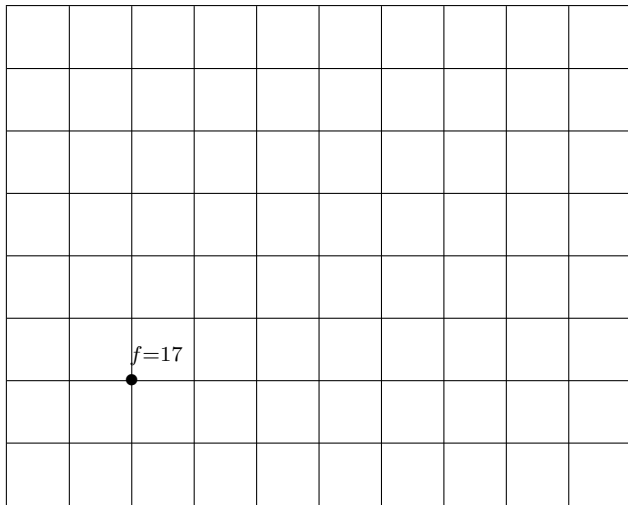
Coordinate Search: Basic algorithm

After 20 eval.:

$$x_5 = (0, 0)^T, \Delta_5 = \frac{1}{2}$$

$$f(x) = 17$$

Can we do better?



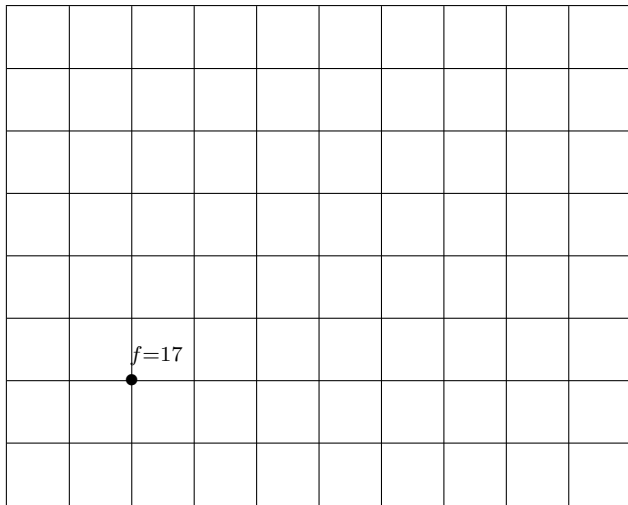
Coordinate Search: Basic algorithm

After 20 eval.:

$$x_5 = (0, 0)^T, \Delta_5 = \frac{1}{2}$$

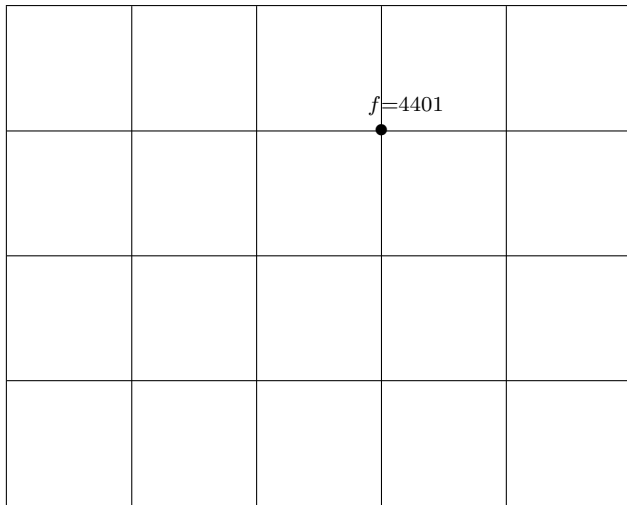
$$f(x) = 17$$

Can we do better?



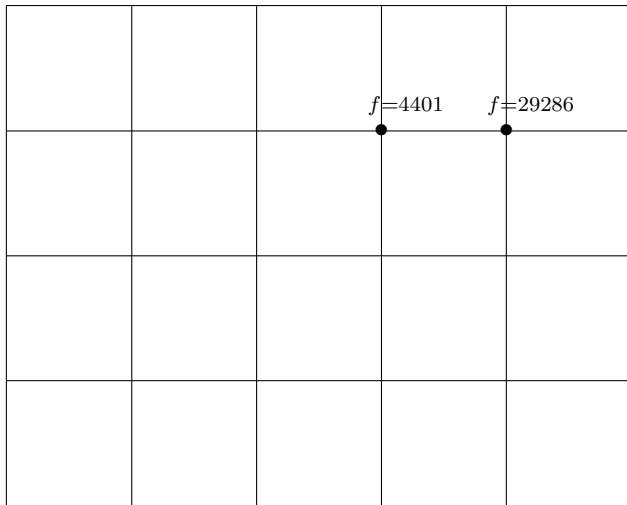
Opportunistic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



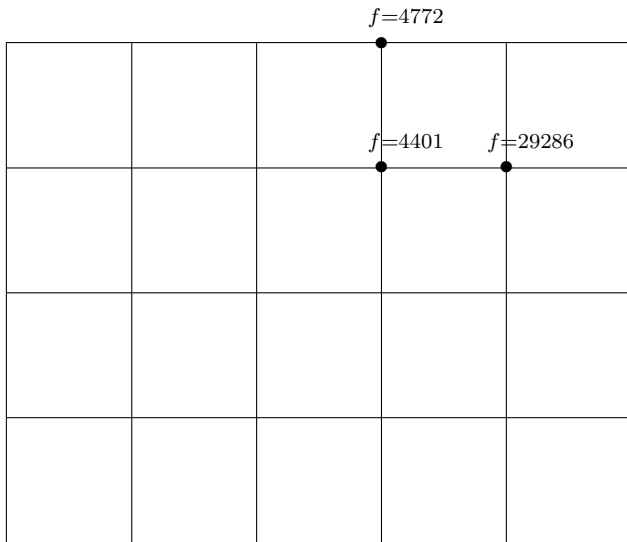
Opportunistic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



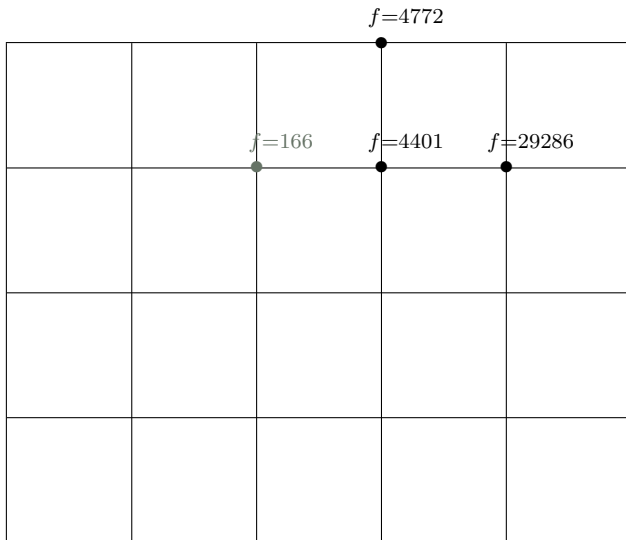
Opportunistic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



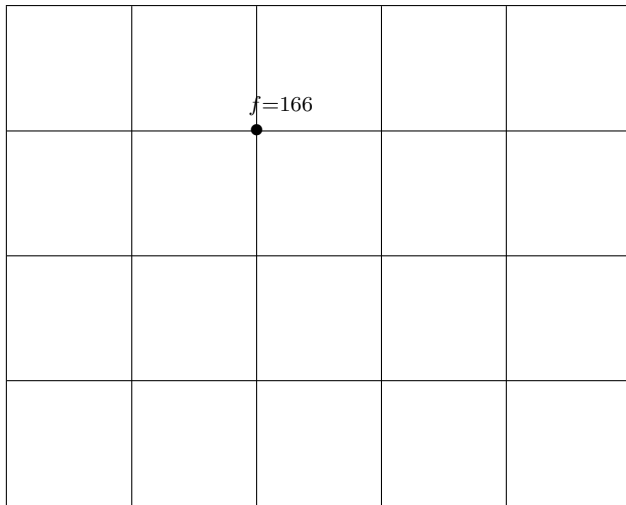
Opportunistic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



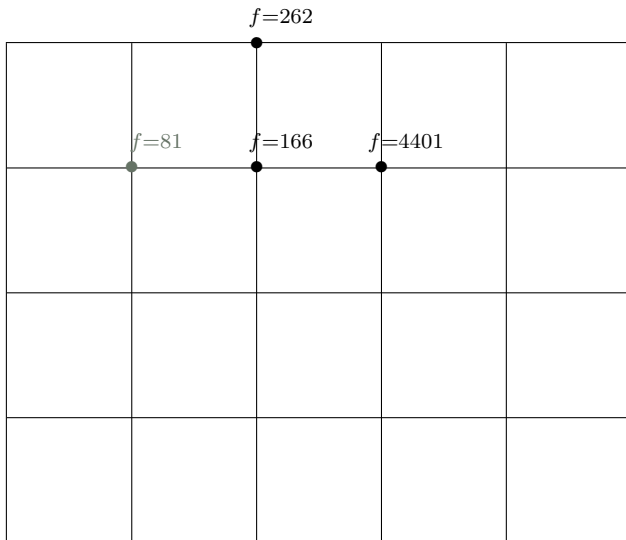
Opportunistic Coordinate Search

$$x_1 = (1, 2)^\top, \Delta_1 = 1$$



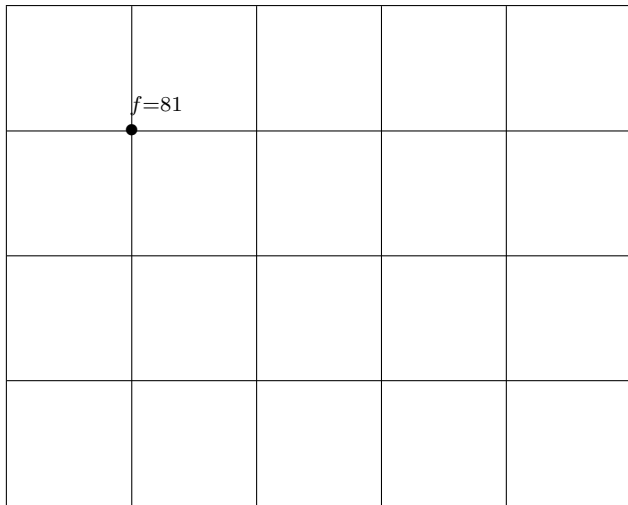
Opportunistic Coordinate Search

$$x_1 = (1, 2)^\top, \Delta_1 = 1$$

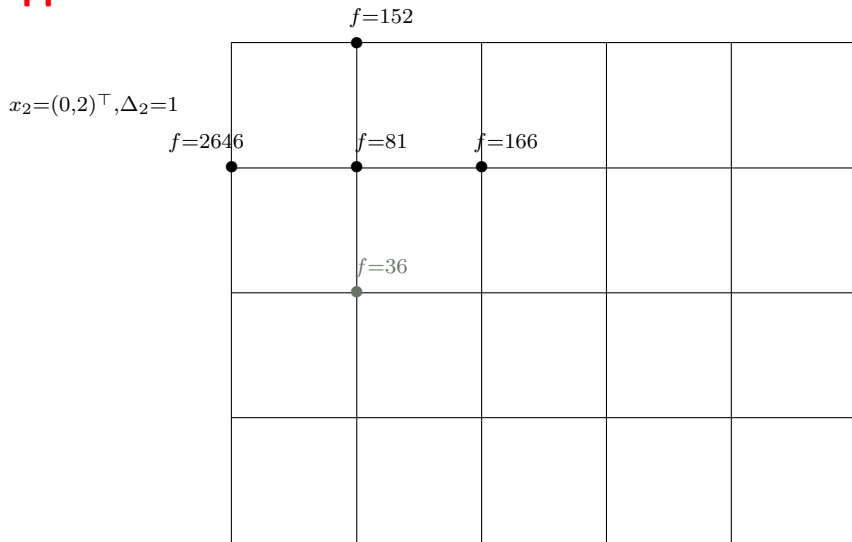


Opportunistic Coordinate Search

$$x_2 = (0, 2)^\top, \Delta_2 = 1$$

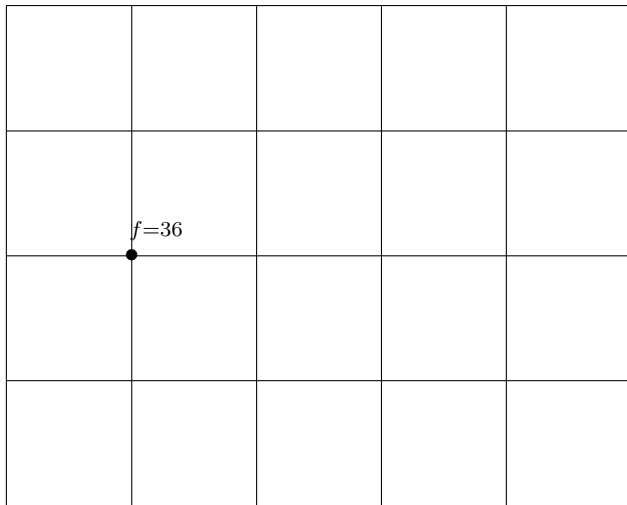


Opportunistic Coordinate Search



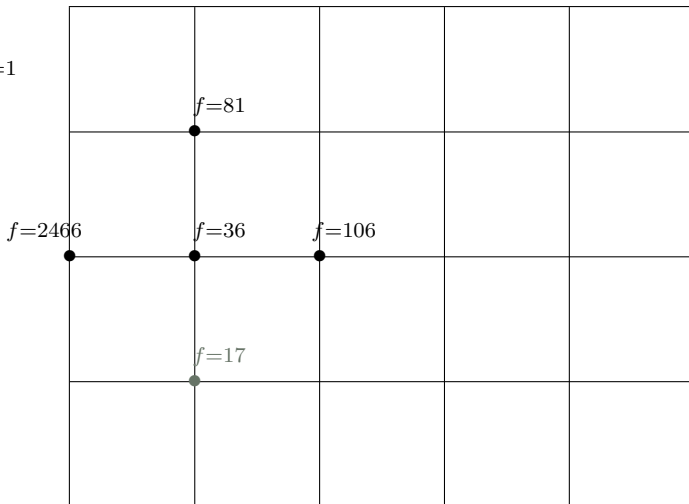
Opportunistic Coordinate Search

$$x_3 = (0, 1)^\top, \Delta_3 = 1$$



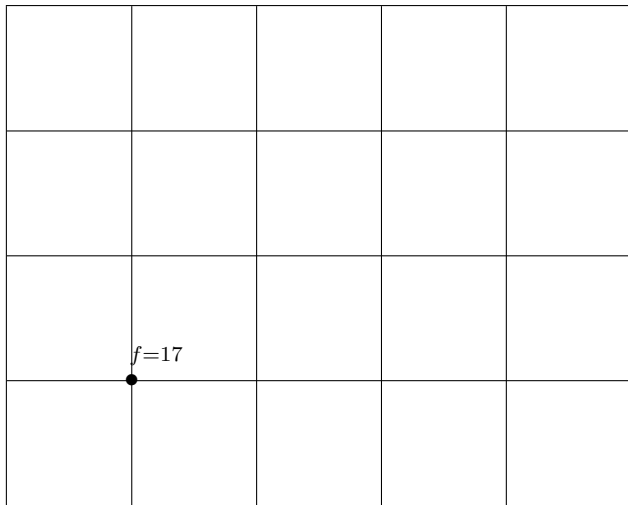
Opportunistic Coordinate Search

$$x_3 = (0, 1)^\top, \Delta_3 = 1$$



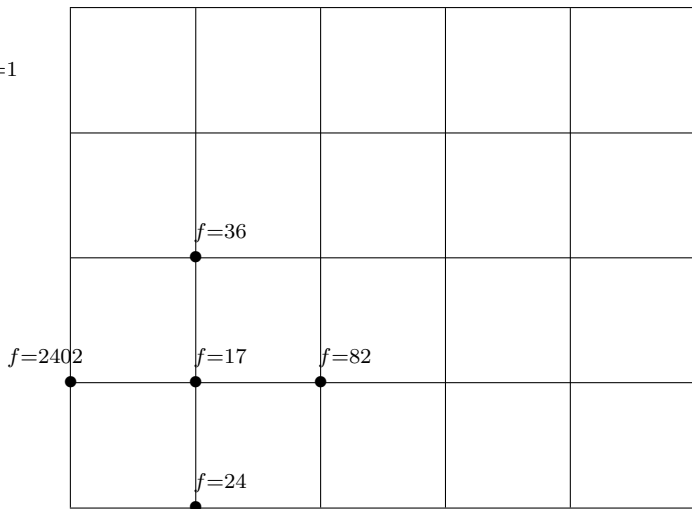
Opportunistic Coordinate Search

$$x_4 = (0, 0)^\top, \Delta_4 = 1$$



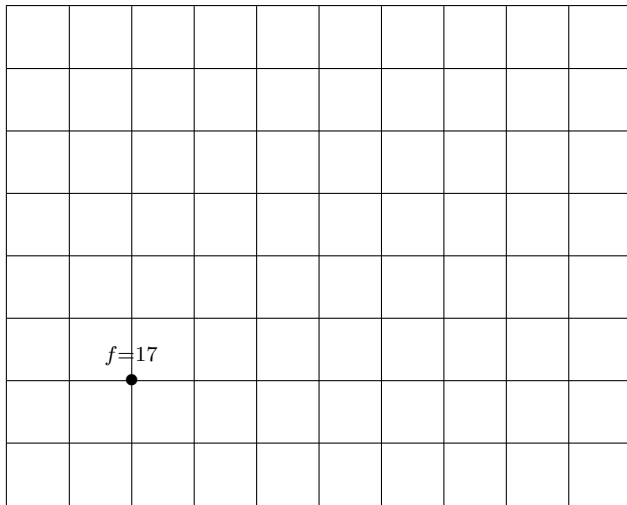
Opportunistic Coordinate Search

$$x_4 = (0, 0)^\top, \Delta_4 = 1$$



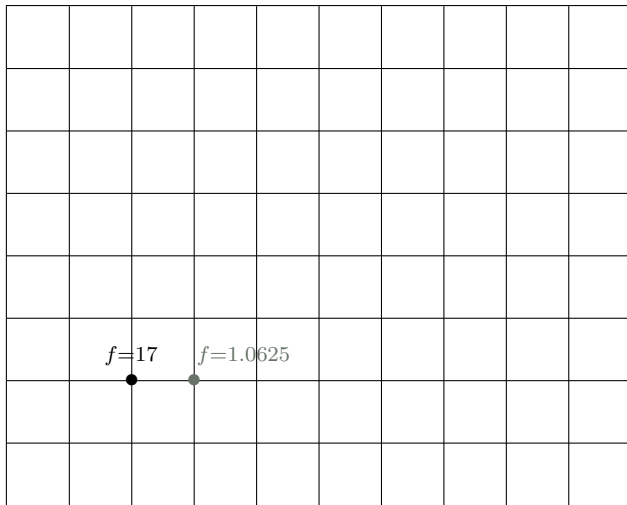
Opportunistic Coordinate Search

$$x_5(0,0)^\top, \Delta_5 = \frac{1}{2}$$



Opportunistic Coordinate Search

$$x_5(0,0)^T, \Delta_5 = \frac{1}{2}$$



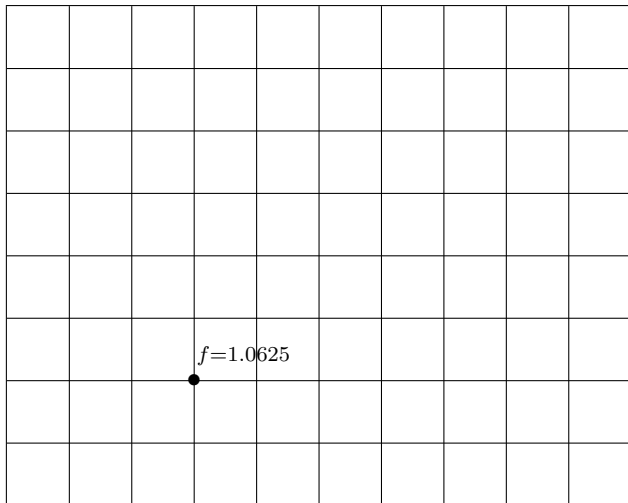
Opportunistic Coordinate Search

After 20 eval.:

$$x_6 = (0.5, 0)^T, \Delta_6 = \frac{1}{2}$$

$$f(x) = 1.0625$$

Can we do better?



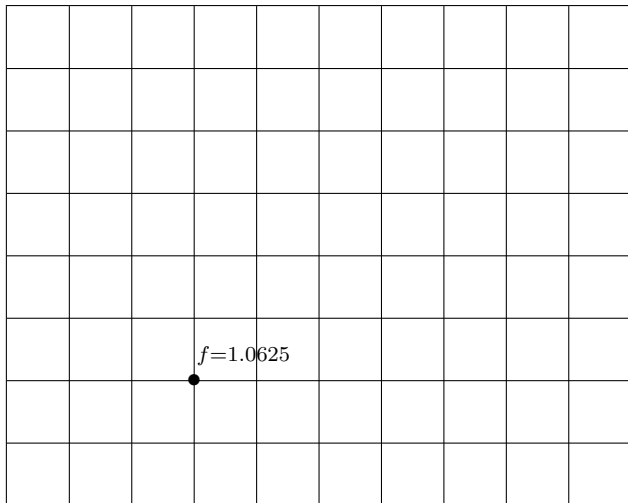
Opportunistic Coordinate Search

After 20 eval.:

$$x_6 = (0.5, 0)^T, \Delta_6 = \frac{1}{2}$$

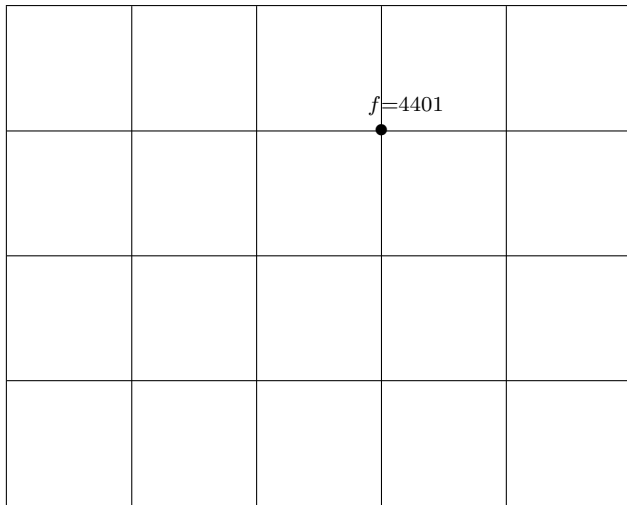
$$f(x) = 1.0625$$

Can we do better?



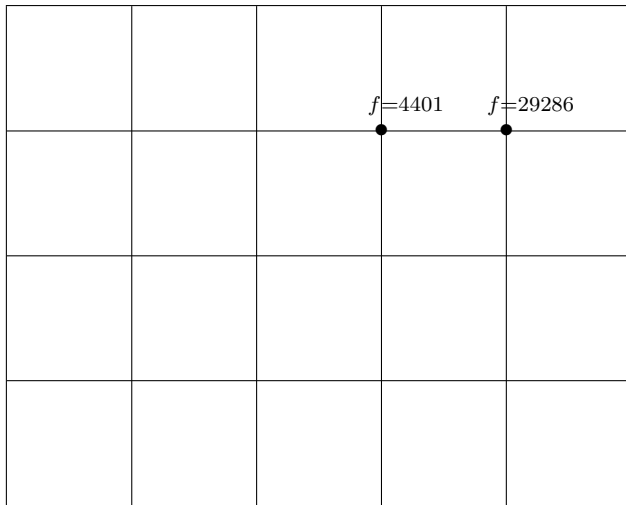
Dynamic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



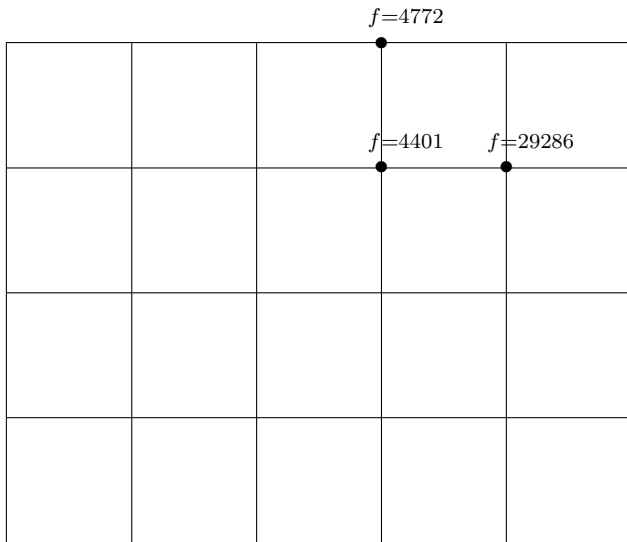
Dynamic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



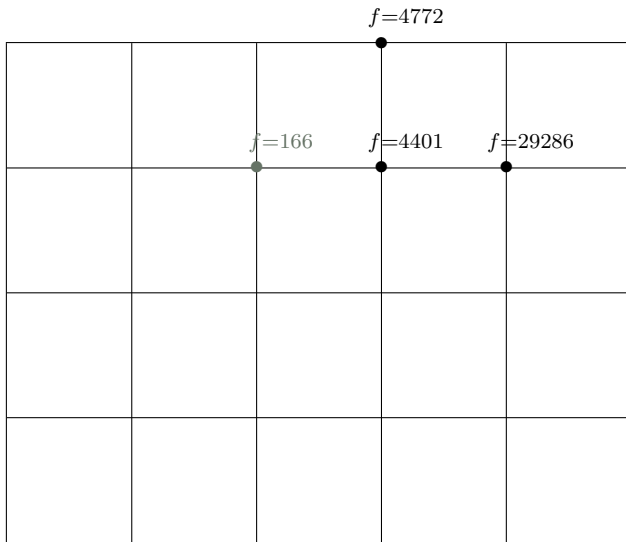
Dynamic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



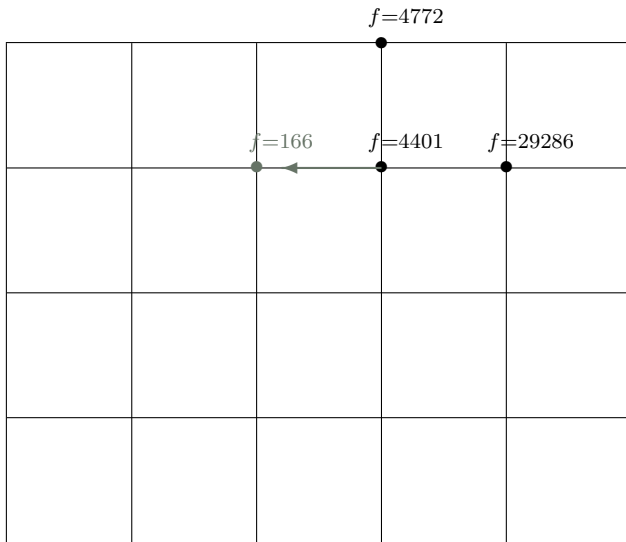
Dynamic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



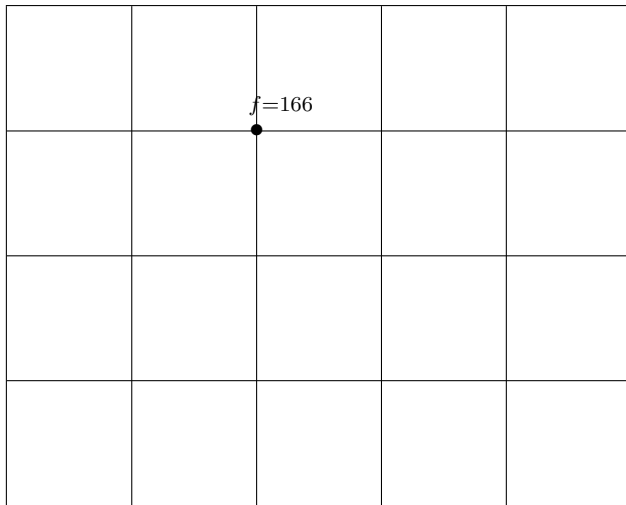
Dynamic Coordinate Search

$$x_0 = (2, 2)^\top, \Delta_0 = 1$$



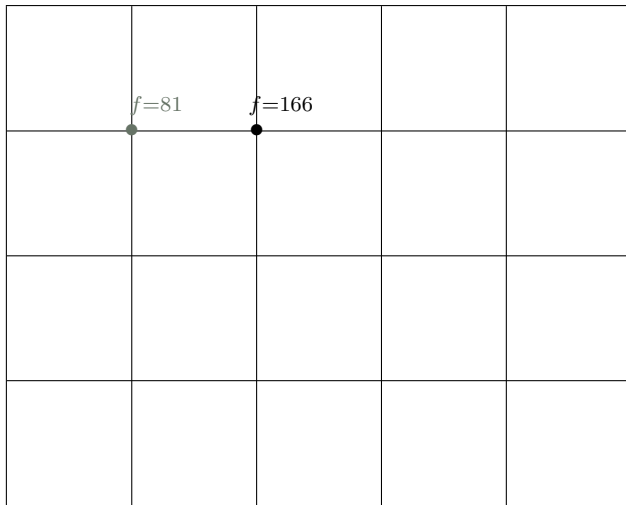
Dynamic Coordinate Search

$$x_1 = (1, 2)^\top, \Delta_1 = 1$$



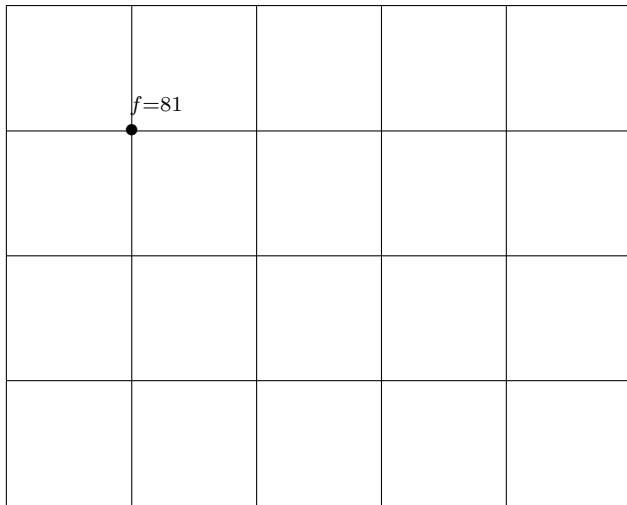
Dynamic Coordinate Search

$$x_1 = (1, 2)^\top, \Delta_1 = 1$$



Dynamic Coordinate Search

$$x_2 = (0, 2)^\top, \Delta_2 = 1$$

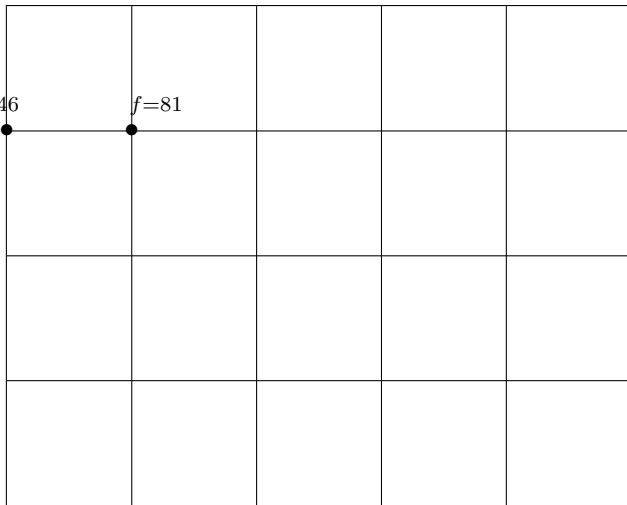


Dynamic Coordinate Search

$$x_2 = (0, 2)^\top, \Delta_2 = 1$$

$$f = 2646$$

$$f = 81$$



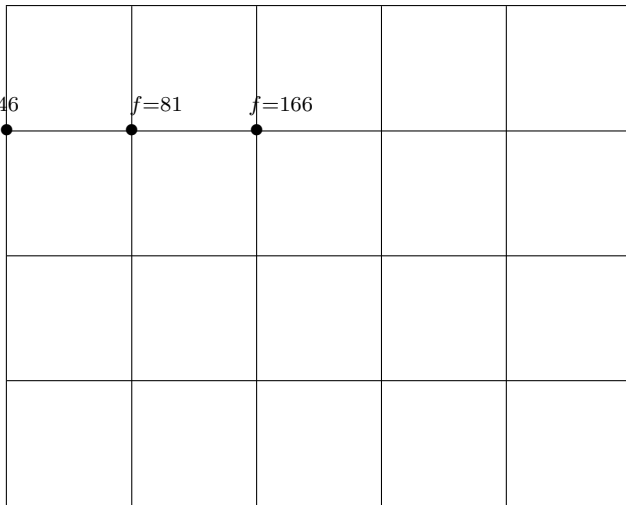
Dynamic Coordinate Search

$$x_2 = (0, 2)^\top, \Delta_2 = 1$$

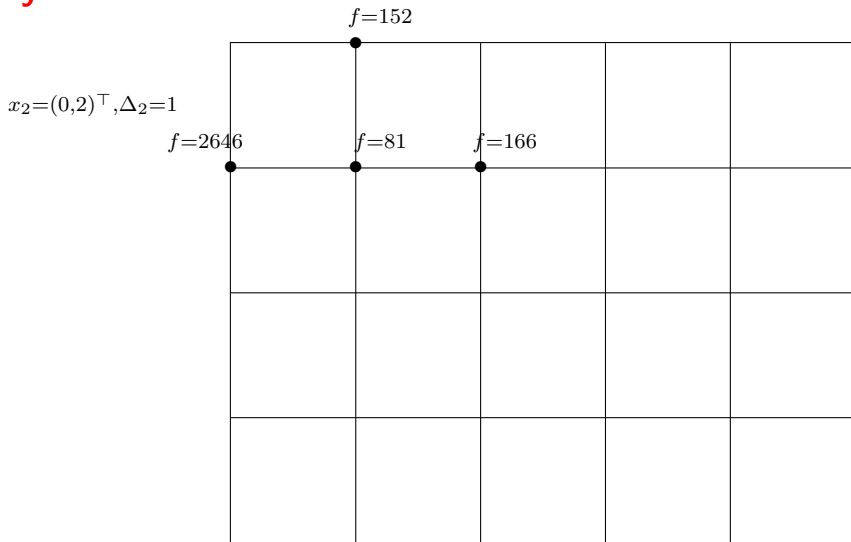
$$f = 2646$$

$$f = 81$$

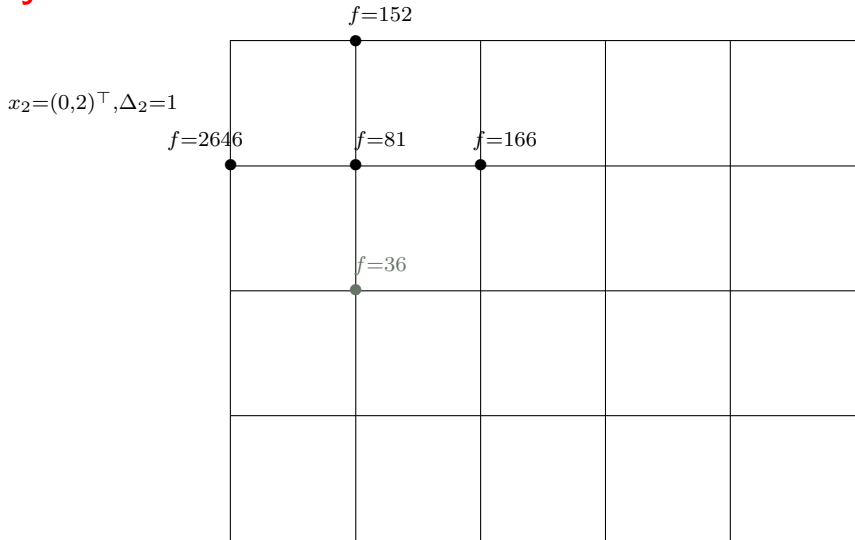
$$f = 166$$



Dynamic Coordinate Search

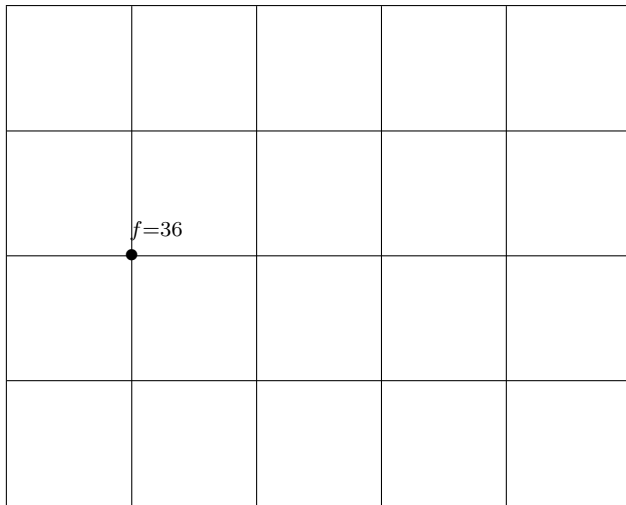


Dynamic Coordinate Search



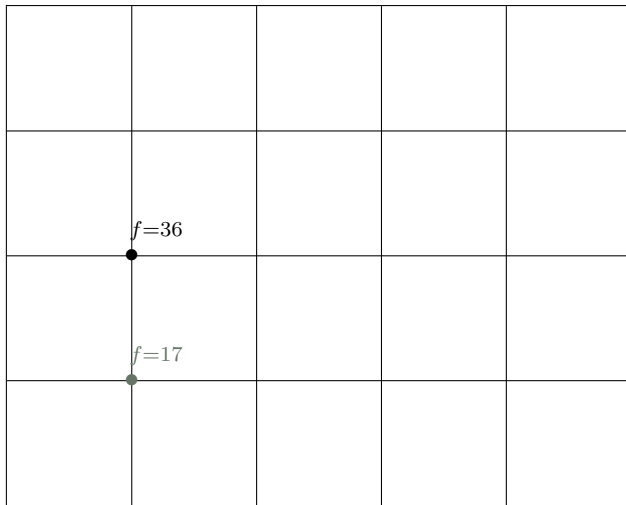
Dynamic Coordinate Search

$$x_3 = (0, 1)^\top, \Delta_3 = 1$$



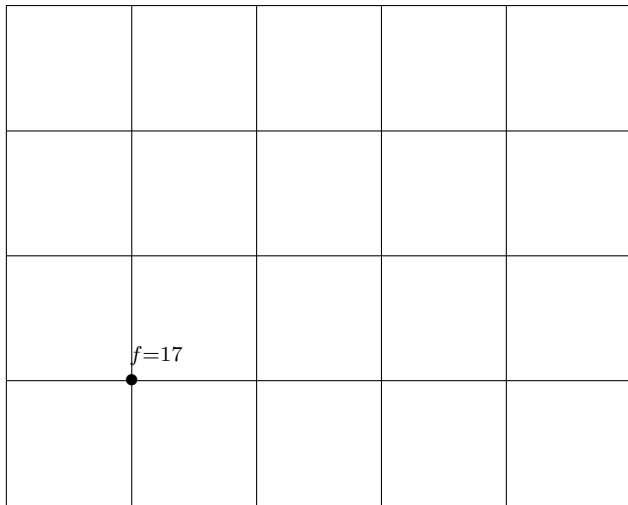
Dynamic Coordinate Search

$$x_3 = (0, 1)^\top, \Delta_3 = 1$$



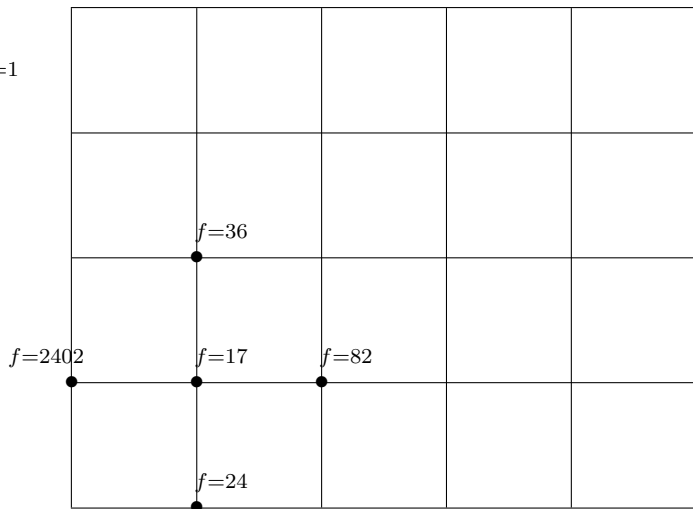
Dynamic Coordinate Search

$$x_4 = (0, 0)^\top, \Delta_4 = 1$$



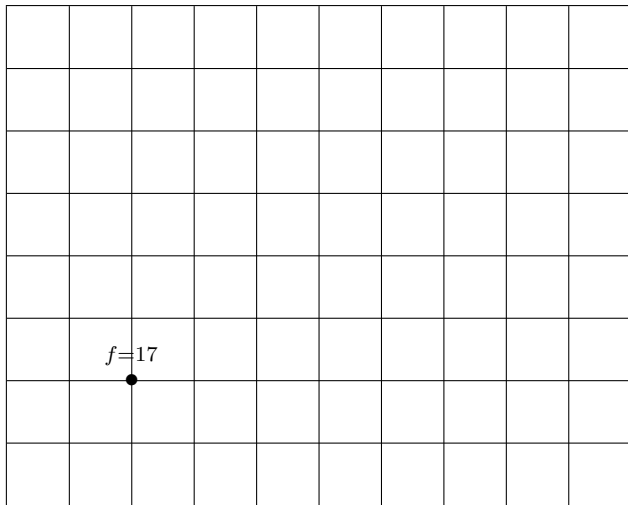
Dynamic Coordinate Search

$$x_4 = (0,0)^\top, \Delta_4 = 1$$



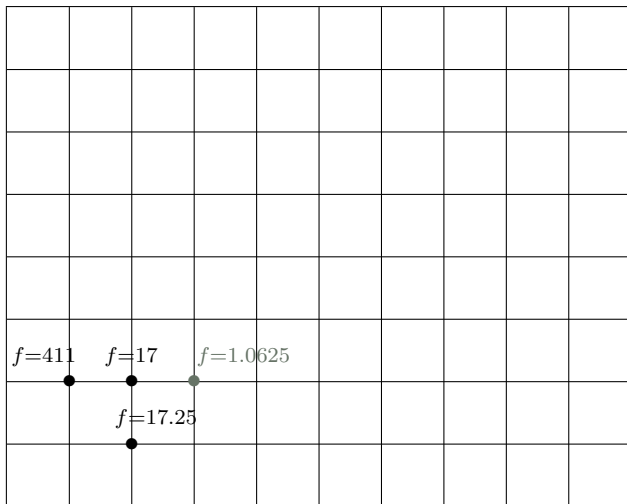
Dynamic Coordinate Search

$$x_5 = (0, 0)^T, \Delta_5 = \frac{1}{2}$$



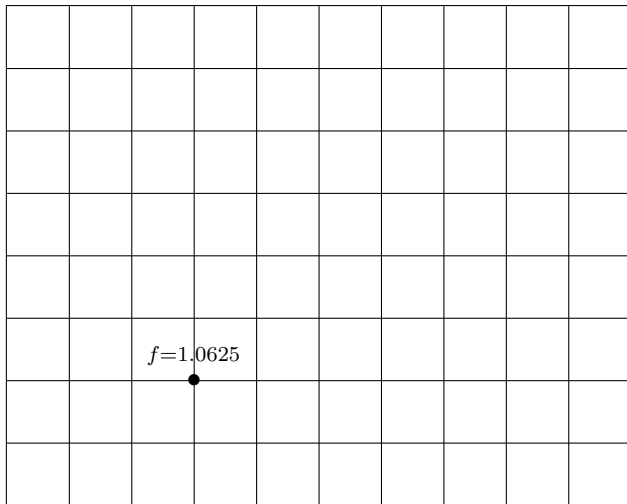
Dynamic Coordinate Search

$$x_5 = (0, 0)^T, \Delta_5 = \frac{1}{2}$$



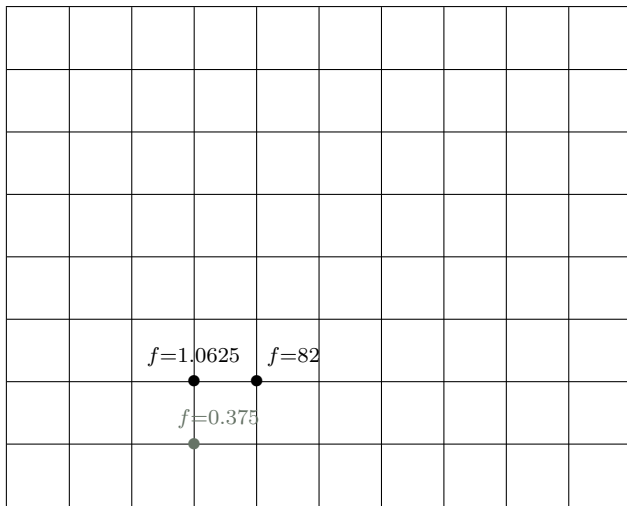
Dynamic Coordinate Search

$$x_6 = (0.5, 0)^T, \Delta_6 = \frac{1}{2}$$



Dynamic Coordinate Search

$$x_6 = (0.5, 0)^\top, \Delta_6 = \frac{1}{2}$$



Coordinate Search: 3 strategies

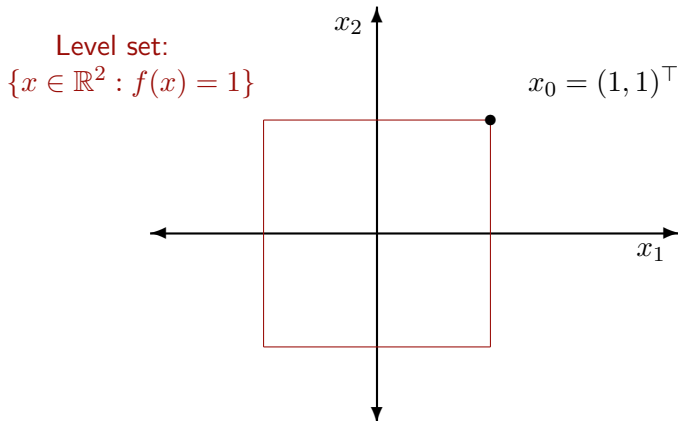
Basic (complete)		Opportunistic		Dynamic	
x^\top	$f(x)$	x^\top	$f(x)$	x^\top	$f(x)$
After 20 evaluations of f					
(0,0)	17	(.5,0)	1.0625	(.5,-.5)	.375
After 50 evaluations of f					
(.375, -.375)	1.8E-2	(.375, -.312)	5.7E-3	(.375, -.344)	3.1E-3

Coordinate Search: Convergence analysis

- ▶ Special case of GPS for which convergence was proved by [Torczon, 1997]
- ▶ If the sequence of iterates $\{x_k\}$ is bounded, then
 - ▶ $\lim_k \Delta_k = 0$
 - ▶ There exists $\hat{x} \in \mathbb{R}^n$, the limit of a subsequence of mesh local optimizers, whose size is infinitely small
 - ▶ If f is continuously differentiable at \hat{x} , then $\nabla f(\hat{x}) = 0$

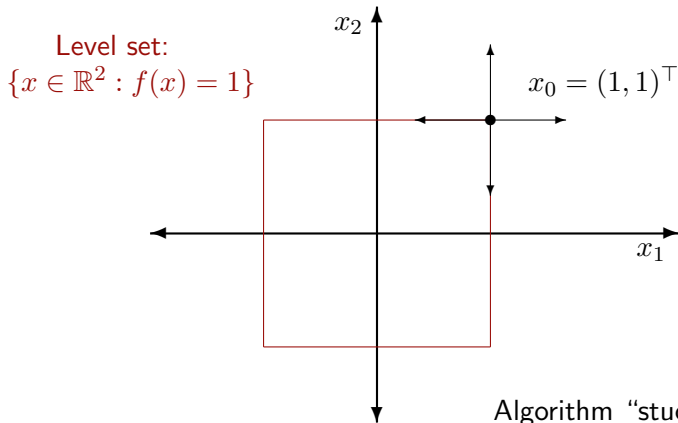
Coordinate Search: Failure

$$f(x) = \|x\|_\infty \text{ from } x_0 = (1, 1)^\top:$$



Coordinate Search: Failure

$$f(x) = \|x\|_\infty \text{ from } x_0 = (1, 1)^\top:$$



Latin Hypercube Sampling

The Coordinate Search method

The Nelder-Mead method

References

The Nelder-Mead method

- ▶ The **Nelder-Mead** (NM) algorithm, or the *other Simplex* method [Nelder and Mead, 1965]
- ▶ No link with the Simplex method from Dantzig for linear programming
- ▶ 1965: Good timing with the emergence of blackbox optimization problems
- ▶ Most used and cited method for unconstrained optimization

Nelder-Mead: Idea

- ▶ **Simplex** in dimension n : set of $n + 1$ points
- ▶ At each iteration, such a simplex is defined:

$$Y = \{y^0, y^1, \dots, y^n\}$$

with $f(y^0) \leq f(y^1) \leq \dots \leq f(y^n)$

- ▶ The centroid y^c of the first n points is defined as

$$y^c = \sum_{i=0}^{n-1} y^i / n$$

- ▶ Depending on the success of the iteration, the shape and the size of the simplex are updated

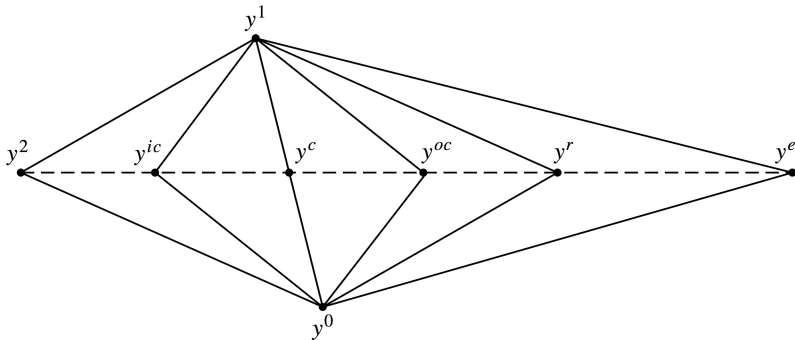
Nelder-Mead: Transformations (1/2)

- ▶ Five different transformations:
 - ▶ Reflection
 - ▶ Expansion
 - ▶ Outside contraction
 - ▶ Inside contraction
 - ▶ Shrink
- ▶ Except for the shrink, the worst vertex of the simplex is replaced by one of the reflection, expansion, or contraction points

Nelder-Mead: Transformations (2/2)

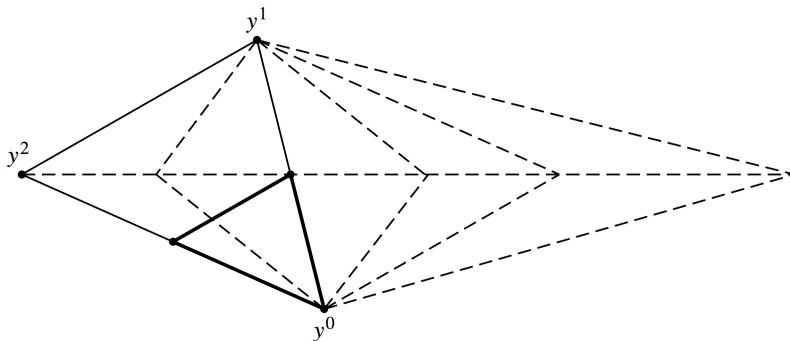
- ▶ **Reflection** point: $y^r = y^c + \delta^r(y^c - y^n)$
- ▶ **Expansion** point: $y^e = y^c + \delta^e(y^c - y^n)$
- ▶ **Outside contraction** point: $y^{oc} = y^c + \delta^{oc}(y^c - y^n)$
- ▶ **Inside contraction** point: $y^{ic} = y^c + \delta^{ic}(y^c - y^n)$
- ▶ **Shrink** points: $y^{s_i} = y^0 + \gamma^s(y^i - y^0)$ for $i = 1, 2, \dots, n$
- ▶ $0 < \gamma^s < 1$, $-1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e$
- ▶ Standard choices: $\delta^{ic} = -0.5$, $\gamma^s = \delta^{oc} = 0.5$, $\delta^r = 1$, $\delta^e = 2$

Nelder-Mead: Reflection, expansion, contractions



Taken from [Conn et al., 2009]

Nelder-Mead: Shrink



Taken from [Conn et al., 2009]

Nelder-Mead: Algorithm (1/4)

[0] Initializations

evaluate f at $Y_0 = \{y_0^0, y_0^1, \dots, y_0^n\}$

$0 < \gamma^s < 1$, $-1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e$

$k \leftarrow 0$

[1] Iteration k

[1.1] Order

order the points in Y_k such that

$f^0 = f(y_k^0) \leq f^1 = f(y_k^1) \leq \dots \leq f^n = f(y_k^n)$

[1.2] Reflect

construct y^c and y^r and evaluate $f^r = f(y^r)$

if ($f^0 \leq f^r < f^{n-1}$)

$Y_{k+1} \leftarrow \{y_k^0, y_k^1, \dots, y_k^{n-1}, y^r\}$

$k \leftarrow k + 1$

goto [1]

...

Nelder-Mead: Algorithm (2/4)

[1] Iteration k

...

[1.3] **Expand** if ($f^r < f^0$)

Construct y^e and evaluate $f^e = f(y^e)$

if ($f^e \leq f^r$)

 | $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \dots, y_k^{n-1}, y^e\}$

otherwise

 | $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \dots, y_k^{n-1}, y^r\}$

$k \leftarrow k + 1$

goto [1]

...

Nelder-Mead: Algorithm (3/4)

[1] Iteration k

...

[1.4] Contract ($f^r \geq f^{n-1}$)

Outside contraction if ($f^r < f^n$)

construct y^{oc} and evaluate $f^{oc} = f(y^{oc})$

if ($f^{oc} \leq f^r$)

$Y_{k+1} \leftarrow \{y_k^0, y_k^1, \dots, y_k^{n-1}, y^{oc}\}$

$k \leftarrow k + 1$, goto [1]

otherwise goto [1.5]

Inside contraction if ($f^r \geq f^n$)

construct y^{ic} and evaluate $f^{ic} = f(y^{ic})$

if ($f^{ic} < f^n$)

$Y_{k+1} \leftarrow \{y_k^0, y_k^1, \dots, y_k^{n-1}, y^{ic}\}$

$k \leftarrow k + 1$, goto [1]

...

Nelder-Mead: Algorithm (4/4)

[1] Iteration k

...

[1.5] Shrink ($f^r \geq f^{n-1}$)

construct the n points y^{si} , $i = 1, 2, \dots, n$

evaluate f at these n points

$$Y_{k+1} \leftarrow \{y_k^0, y^{s1}, y^{s2}, \dots, y^{sn}\}$$

$$k \leftarrow k + 1$$

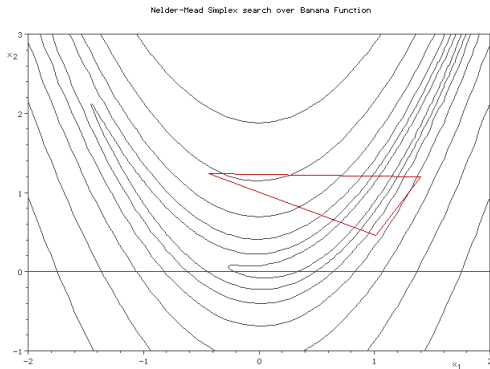
goto [1]

Nelder-Mead: Other details

- ▶ Stopping criteria:
 - ▶ Minimal diameter size
 - ▶ Number of evaluations
 - ▶ Number of shrinks
 - ▶ etc.

- ▶ Number of evaluations at each iteration:
 - ▶ 1 if the iteration is a reflection
 - ▶ 2 if the iteration is an expansion or a contraction
 - ▶ $n + 2$ if the iteration is a shrink

Nelder-Mead: Execution example



See the animation [NM_animation.pdf](#) taken from [Wikipedia](#), on the [Rosenbrock](#) function $f(x, y) = (1 - x)^2 + 100(y - x^2)^2$

Nelder-Mead: Convergence (1/2)

- ▶ It is a **heuristic**. No convergence theory. But at the time, the method proved to be very efficient compared to what was available before
- ▶ From [Swann, 1972]: *“Although the methods described above have been developed heuristically and no proofs of convergence have been derived for them, in practice they have generally proved to be robust and reliable”*
- ▶ With time, deficiencies appeared, including the famous McKinnon family of strictly convex functions, for which NM executes an infinite sequence of inside contractions and fails to converge [McKinnon, 1998]

Nelder-Mead: Convergence (2/2)

- ▶ Modified versions of the NM method are in use nowadays. They correct its known deficiencies with:
 - ▶ Use of a **sufficient decrease** instead of a **simple decrease**
 - ▶ Restart when the simplex becomes ill-conditioned
 - ▶ Lexicographic rules to break the ties when ordering
- ▶ However NM remains an “*old*” method and modern algorithms should be preferred
- ▶ But NM is still very popular amongst engineers
- ▶ NM is still analyzed to understand why it works so well in some instances

Latin Hypercube Sampling

The Coordinate Search method

The Nelder-Mead method

References

References I



Audet, C. and Dennis, Jr., J. (2006).

Mesh adaptive direct search algorithms for constrained optimization.
SIAM Journal on Optimization, 17(1):188–217.
(MADS).



Conn, A., Scheinberg, K., and Vicente, L. (2009).

Introduction to Derivative-Free Optimization.
MOS-SIAM Series on Optimization. SIAM, Philadelphia.
(NM description, Section 8.1).



Fermi, E. and Metropolis, N. (1952).

Numerical solution of a minimum problem.
Los Alamos Unclassified Report LA-1492, Los Alamos National Laboratory, Los Alamos, USA.
(Coordinate Search).



Hooke, R. and Jeeves, T. (1961).

“Direct Search” Solution of Numerical and Statistical Problems.
Journal of the Association for Computing Machinery, 8(2):212–229.
(H&J).



Kolda, T., Lewis, R., and Torczon, V. (2003).

Optimization by direct search: New perspectives on some classical and modern methods.
SIAM Review, 45(3):385–482.
(GSS).

References II



McKay, M., Beckman, R., and Conover, W. (1979).

A comparison of three methods for selecting values of input variables in the analysis of output from a computer code.

Technometrics, 21(2):239–245.

(Latin Hypercube Sampling).



McKinnon, K. (1998).

Convergence of the Nelder-Mead simplex method to a nonstationary point.

SIAM Journal on Optimization, 9(1):148–158.

(NM failure).



Nelder, J. and Mead, R. (1965).

A simplex method for function minimization.

The Computer Journal, 7(4):308–313.

(NM).



Swann, W. (1972).

Direct Search Methods.

In Murray, W., editor, *Numerical Methods for Unconstrained Optimization*, pages 13–28. Academic Press, London and New York.

(Remark on NM).

References III



Torczon, V. (1997).

On the convergence of pattern search algorithms.

SIAM Journal on Optimization, 7(1):1–25.

(GPS).



Wright, M. (2012).

Nelder, Mead, and the Other Simplex Method.

In Grötschel, M., editor, *Documenta Mathematica Extra Volume: Optimization Stories*, pages 271–276.

Journal der Deutschen Mathematiker-Vereinigung Gegründet 1996, Berlin.

(NM history).