## Genetic Algorithms (GA)

**Outline**

- Introduction
- Basic structure of Genetic Algorithms
- Basic properties of Genetic Algorithms
- Genetic operators
- A simple optimization example
- Main difficulties of Genetic Algorithms
- Case-study



## Introduction into Genetic Algorithms

- Search technique based on the principle of evolution
- Invented in the early 70's by John Holland
- GAs use two basic processes from evolution
  - inheritance (passing of features from one generation to the next)
  - competition (survival of the fittest)
- Goal: Evolution through alternation in generations towards better and better regions of the search space

## Basic Structure of Genetic Algorithms

**Data structure D**

$P(t) = \{a_1^t, a_2^t, ..., a_n^t\}$

t: generation counter

n: number of individuals

$a_i^t$: fixed-length binary strings

(i=1,...,n)

| 100...1 | 010...0 | ... | 110...1 |
|---|---|---|---|
| $p_1$ | $p_2$ | ... | $p_m$ |

$p_j$: parameter

(j=1,...,m)

**Algorithm A**

```
begin
   t:= 0
   initialize P(t)
   evaluate P(t)
   while (not termination-condition) do
   begin
        t:= t+1
        select P(t) from P(t-1)   {selection}
        recombine P(t)   {crossover, mutation}
        evaluate P(t)
   end
end
```

Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)

3

## Basic Properties of Genetic Algorithms

- GAs manipulate a **coding** of the optimized parameters
- GAs search from a **population**, not a single point
- GAs use **only goal function values** to guide the search process (blind search)
- GAs use **stochastic operators** instead of deterministic rules

Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)

4

## Genetic Operators

**Selection**

- task: select n individuals from the previous population
- realization alternative: Roulette Wheel Selection
  - fitness proportional selection
  - an individual $a_i$ is selected with probability

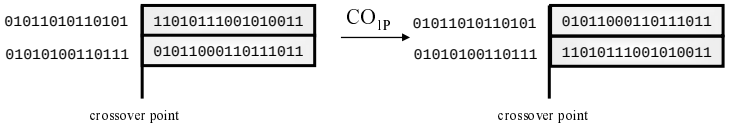
$$p_s(a_i) = \frac{F(a_i)}{\sum_{j=1}^{n} F(a_j)}$$

- properties: not extinctive
  - each individual has principally the chance to be selected

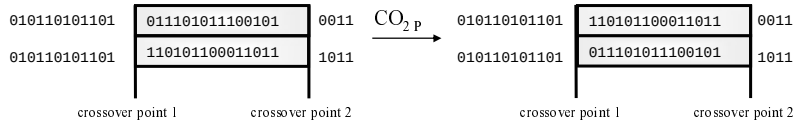


## Genetic Operators

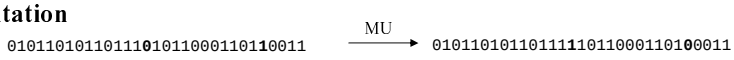
- **Crossover**
  - 1-point-crossover




  - 2-point-crossover

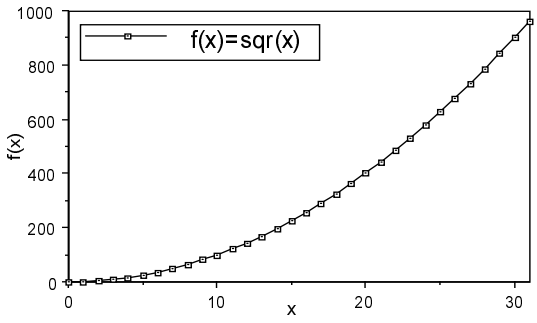



- **Mutation**



## A Simple Optimization Example

- **Optimization of** $f(x)=x^2$, with $x \in [0,31]$
- **Problem representation**
  - encoding of the variable x as a binary vector
  - [0, 31] → [00000, 11111]




Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)
7

## A Genetic Algorithm by Hand

| string No. | initial population | x value | fitness $f(x)=x^2$ | % of total fitness | selection probability |
|---|---|---|---|---|---|
| 1 | 01101 | 13 | 169 | 14,4 | 0,144 |
| 2 | 11000 | 24 | 576 | 49,2 | 0,492 |
| 3 | 01000 | 8 | 64 | 5,5 | 0,055 |
| 4 | 10011 | 19 | 361 | 30,9 | 0,309 |

| after selection | mate | crossover point | mutation | new population | fitness $f(x)=x^2$ |
|---|---|---|---|---|---|
| 0110 \| 1 | 2 | 4 | - | 01100 | 144 |
| 1100 \| 0 | 1 | 4 | 3 | 11101 | 841 |
| 11 \| 000 | 4 | 2 | - | 11011 | 729 |
| 10 \| 011 | 2 | 2 | - | 10000 | 256 |

Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)
8

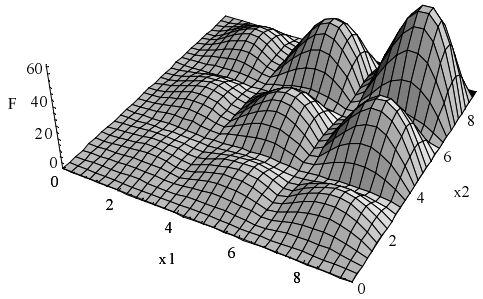
## Main Difficulties of Genetic Algorithms

- Adjustment of the GA control parameters
  - *population size*
  - *crossover probability*
  - *mutation probability*
- Specification of the termination condition
- Representation of the problem solutions



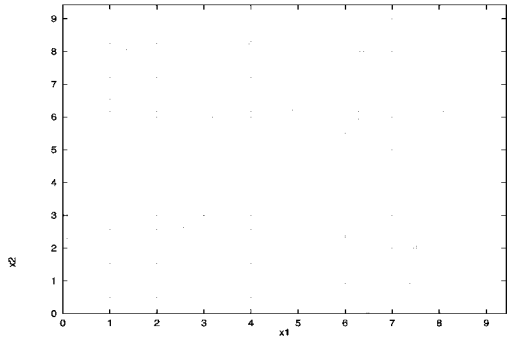
## Case-Study

$$F(\vec{x}) = \left| \prod_{i=1}^{2} x_i \cdot \sin(x_i) \right|$$



# Genetic Algorithm

Population P(0)

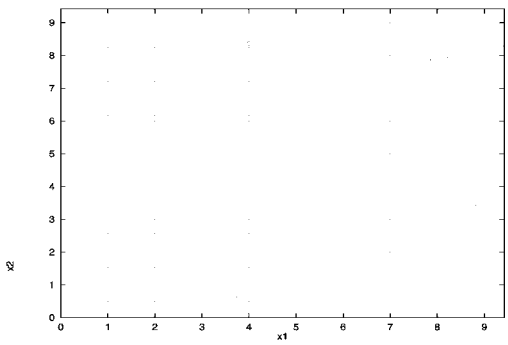



**population-based optimization**



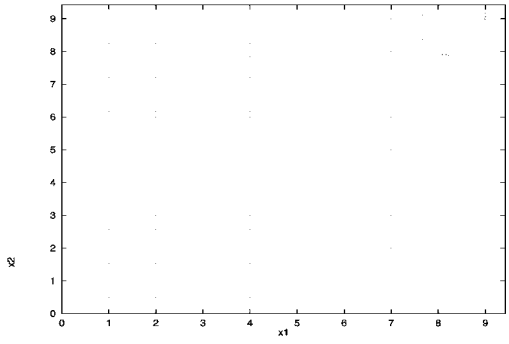
# Genetic Algorithm

Population P(1)



# Genetic Algorithm

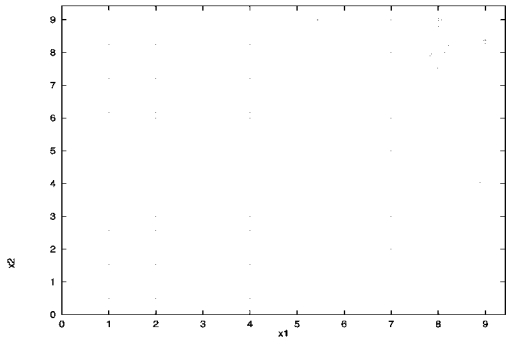
Population P(2)





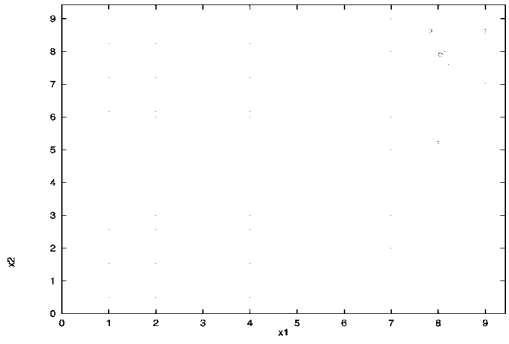

# Genetic Algorithm

Population P(3)



# Genetic Algorithm

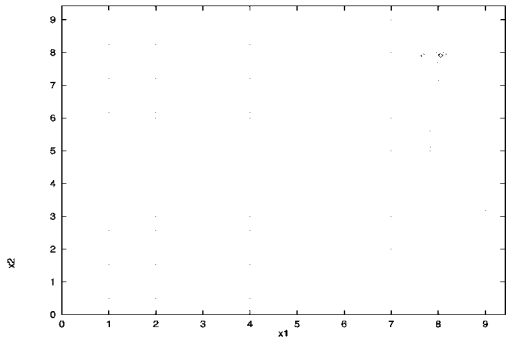
Population P(4)





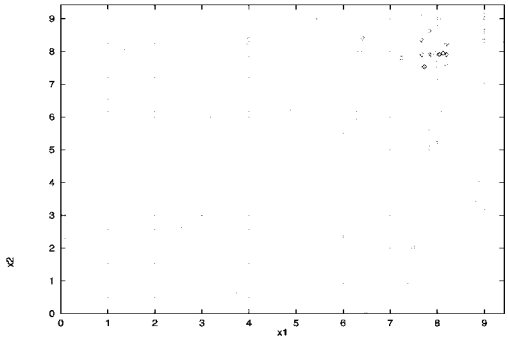

# Genetic Algorithm

Population P(5)



## Genetic Algorithm

Complete optimization trajectory (P(0) - P(5))






## Additional Information

- **Literature**
  - Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning; Addison-Wesley, 1989.
  - Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs; Springer, 1992.
  - Bäck, Th.: Evolutionary Algorithms in Theory and Practice; Oxford University Press, New York, 1996.
  - Mazumder, P.; Rudnick, E.: Genetic Algorithms for VLSI Design, Layout & Test Automation; Prentice Hall, 1999.
- **Information on the Web**
  - http://www-illigal.ge.uiuc.edu/illigal.home.html
  - http://www.aic.nrl.navy.mil/galist/
  - http://www.aracnet.com/~wwir/NovaGenetica/

## Simulated Annealing

**Outline**

- Introduction
- Basic structure of the Simulated Annealing algorithm
- Properties of Simulated Annealing
- Difficulties of Simulated Annealing
- A realization alternative
- Case-study



## Introduction into Simulated Annealing

- In the early 80's Kirkpatrick, Gelatt, and Vecchi and independently Cerny introduced the concepts of annealing in combinatorial optimization.
- In condensed matter physics annealing is known as a thermal process for obtaining low energy states of a solid in a heat bath.
- The concept of Simulated Annealing is based on a strong analogy between the physical annealing process of solids and the problem of solving large combinatorial optimization problems.

## Introduction into Simulated Annealing

In order to reach a low energy state the following two steps are carried out during the physical annealing process:

- Increase the temperature of the heat bath to a maximum value at which the solid melts. In this liquid phase the particles of the solid arrange themselves randomly.
- Decrease carefully the temperature of the heat bath until the particles arrange themselves in the ground state of the solid.



## Introduction into Simulated Annealing

- The goal of the physical annealing process is to reach the ground state of the solid where the particles are arranged in a highly structured lattice and the energy of the system is minimal.
- The ground state of the solid is obtained only if the maximum temperature is sufficiently high and the cooling is done sufficiently low.
- Otherwise the solid will be frozen into a meta-stable state rather than into the ground state.

## Introduction into Simulated Annealing

- Methods for simulation of the physical annealing process can be directly applied to solve combinatorial optimization problems.
- The analogy between a physical many-particle system and a combinatorial optimization problem is based on the following equivalences:
  - solutions in a combinatorial optimization problem are equivalent to states of a physical system,
  - the cost of a solution is equivalent to the energy of a state.



## Basic Structure of the Simulated Annealing Algorithm

```
procedure Simulated_Annealing;    {Minimization}
begin
    Choose_a_starting_solution (i_start ∈ L);
    initialize(T_0, M_0);
    k:=0;
    i:=i_start;
    repeat
        for m:=1 to M_k do
            begin
                generate(j ∈ L_i);
                if F(j) ≤ F(i) then i:=j
                else
                if exp( (F(i) - F(j)) / T_k ) > random (0,1) then i:=j;
            end;                                              } inner loop
        k:=k+1;
        Calculate_M(M_k);
        Calculate_Temperature(T_k);
    until termination criterion;                              } outer loop
end;
```

## Basic Structure of the Simulated Annealing Algorithm

- The Simulated Annealing algorithm represents an iterative process consisting of an inner and an outer loop.
- Within the outer loop the temperature T is cooled down.
- At temperature level $T_k$ the inner loop performs $M_k$ transitions.



## Basic Structure of the Simulated Annealing Algorithm

- A transition is a combined action resulting in the transformation of a current solution into a subsequent one.
- The action consists of the following two steps:
  - outgoing from the current solution $i \in L$ generation of a neighboring solution $j \in L_i$
  - application of the acceptance criterion

$$P_T\{\text{accept } j\} = \begin{cases} 1 & \text{, if } F(j) \le F(i) \\ \exp\left(\dfrac{F(i) - F(j)}{T}\right) & \text{, if } F(j) > F(i) \end{cases}$$

## Basic Structure of the Simulated Annealing Algorithm

- The acceptance criterion corresponds to the Metropolis criterion

$$P_T\{\text{accept } j\} = \begin{cases} 1 & \text{, if } E_j \le E_i \\ \exp\left(\dfrac{E_i - E_j}{k_B T}\right) & \text{, if } E_j > E_i \end{cases}$$

  which is applied as the acceptance criterion when a physical annealing process is simulated.

- $E_i$ and $E_j$ denote the energy of the solid in state i and j, $k_B$ is a physical constant (Boltzmann constant), and T is the physical temperature.

Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)
27

## Properties of Simulated Annealing

- In contrary to conventional local Hill-Climbing algorithms, which only accept better solutions, Simulated Annealing also to a limited extent accepts deteriorations in cost.
- This is exactly the property that enables Simulated Annealing to escape from local extreme points.
- Initially, at high temperature values, large deteriorations are accepted.
- As the temperature decreases, only smaller deteriorations are accepted.
- Finally, as the temperature approaches 0, no deteriorations are accepted at all and SA behaves like a local optimization algorithm.

Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)
28

## Properties of Simulated Annealing

- Simulated Annealing can be viewed as a generalization of local search.
- As a result Simulated Annealing inherits all the favorable features of local search, i.e. simplicity as well as general applicability.
- Simulated Annealing requires only goal functions values to guide the optimization process.
- For that reason Simulated Annealing is also a direct optimization method.



## Difficulties of Simulated Annealing

- Adjustment of the control parameters $T_k$ and $M_k$
  - suitable initial values
  - calculation of the stepwise reduction of $T_k$ and $M_k$ (annealing schedule)
- Representation of the problem solutions
- Definition of the neighborhood structure $\eta$

## A Realization Alternative for Simulated Annealing

- **Representation of the problem solutions**

  fixed-length binary strings (see Genetic Algorithms)

- **Calculation of $T_k$**

  $T_k := (T_{k-1}-s) \cdot m$, with $k \in N^+$; $s,m \in R$

- **Calculation of $M_k$**

  $M_k := \text{round}(M_{k-1} \cdot \exp(k \cdot \ln(p)))$, with $k \in N^+$; $p \in R$



## A Realization Alternative for Simulated Annealing

**Some realization alternatives for neighbourhood structures**

- neighbourhood structure 1
  - Bit i of the fixed-length binary string is flipped with probability $\exp(-2/i) \cdot (1-1/\text{sqrt}(\text{sqrt}(T_k+1)))$. With a decrease of temperature this kind of neighbourhood structure effects, that increasingly less significant bits of the fixed-length binary string are flipped whereas significant ones keep unchanged.
- neighbourhood structure 2
  - A randomly chosen bit of the fixed-length binary string is flipped. A second bit is flipped with probability 0,5. If the second bit was flipped, another bit is chosen and flipped again with probability 0,5. This procedure stops, if a chosen bit remains unchanged.

## A Realization Alternative for Simulated Annealing

- neighbourhood structure 3
  - Each bit of the fixed-length binary string is flipped with an adjusted mutation probability. This neighbourhood structure works exactly the same as mutation in Genetic Algorithms.

## Additional Information

**Literature**

Aarts, Emile; Korst, Jan: Simulated Annealing and Boltzmann Machines; John Wiley & Sons, 1989.



## The Pattern Search Algorithm of Hooke and Jeeves

**Outline**

- Introduction
- The Pattern Search Algorithm
- Convergence characteristics
- Efficiency
- Advantages
- Disadvantages
- Case-study

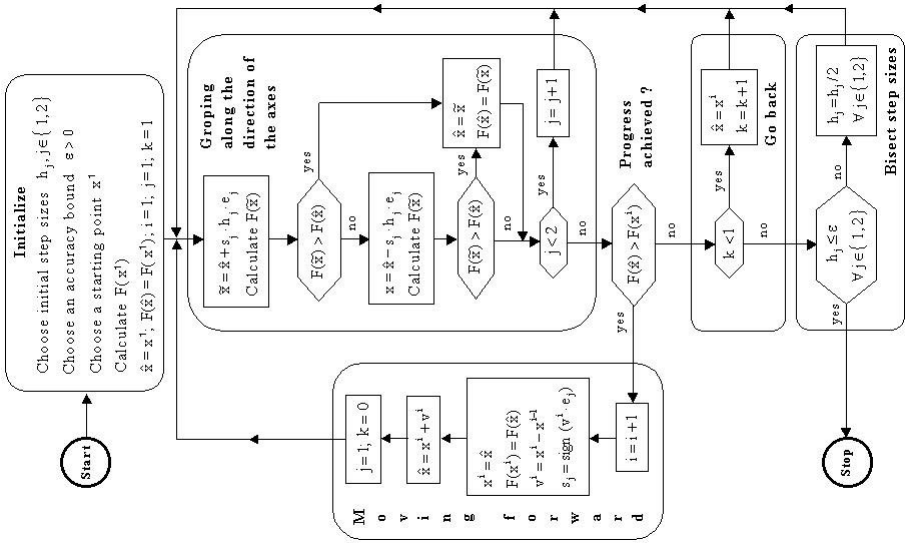
## Introduction

- The Pattern Search algorithm represents a deterministic local optimization method, that was developed by Hooke and Jeeves in the early 60's.
- It is also a direct search method, that exclusively uses goal function values to guide the optimization process.
- The direction of increasing (respectively decreasing) values of the goal function is calculated according to deterministic rules without any calculation of gradients.
- Today it is a wide-spread local optimization method because of its excellent performance characteristics.



## The Pattern Search Algorithm

*(maximization)*



## The Pattern Search Algorithm

**Legend**

i: number of already performed extrapolations

j: coordinate axes ($1 \rightarrow x_1$, $2 \rightarrow x_2$)

k: already moved forward? ($0 \rightarrow$yes, $1 \rightarrow$no)

$\tilde{x}$ : one of the possible test points after an extrapolation

$\hat{x}$ : test point, which is better than the base point after an extrapolation

$x^i$ the best point so far

s: sign vector

h: step size vector

e: vector (1,1)

$v^i$: search direction vector

ε: accuracy bound



## The Pattern Search Algorithm

- For initialization of the Pattern Search algorithm the user has to specify
  - a starting point $x^1$
  - initial step sizes $h_j$, $j \in \{1,...,n\}$, $n \in N$
  - an accuracy bound $\varepsilon \in R$, which represents the termination condition for the Pattern Search algorithm
- The Pattern Search algorithm consists of two different parts:
  - groping along the direction of the axes (exploration)
  - moving forward through extrapolation

## Convergence Characteristics

- The Pattern Search algorithm converges in case of continuous and differentiable goal functions with a strictly convex (minimization) or strictly concave (maximization) function surface.
- Even if these requirements are not completely fulfilled the Pattern Search algorithm usually is still able to compute ε-accurate results.
- In some special cases however the limitation of the groping steps to the direction of the coordinate axes can cause premature termination.



## Efficiency of the Pattern Search Algorithm

The efficiency decisively depends on the choice of the initial step sizes $h_j$, $j \in \{1,...,n\}$:

- When the step sizes are chosen too small the algorithm requires many iterations to appropriately increase the lenght of the search direction vector $v^i$.
- When the step sizes are chosen too large they have to be reduced before the algorithm can advance towards the optimum.

## Advantages of the Pattern Search Algorithm

The Pattern Search algorithm can be viewed as a discretized analogue of gradient-based search techniques with the following advantages:

- excellent convergence characteristics
- low memory capacity is required
- only basic mathematical calculations are used
- numerical stability, because goal function values are exclusively used in comparison operations
- less sensitive against inaccurate calculations compared to gradient-based search methods

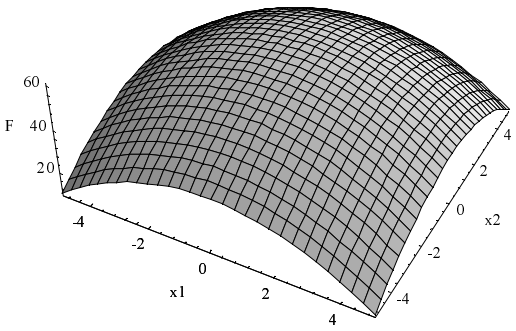


## Disadvantages of the Pattern Search Algorithm

- The limitation of the groping steps to the direction of the coordinate axes can cause premature termination.
- The choice of the starting point and the initial step sizes strictly determine, which optimum will be found. It may be a local or a global optimum.
- If the initial step sizes are chosen too small (or too large), the algorithm requires many iteration cycles, until an optimum is found.
- The Pattern Search algorithm has big problems with intense stochastic inaccuracies of the goal function. Especially when the initial step sizes are chosen too small the probability for premature convergence towards a sub-optimal region is very high.

## Case-Study

$$F(\vec{x}) = 60 - \sum_{i=1}^{2} x_i^2$$

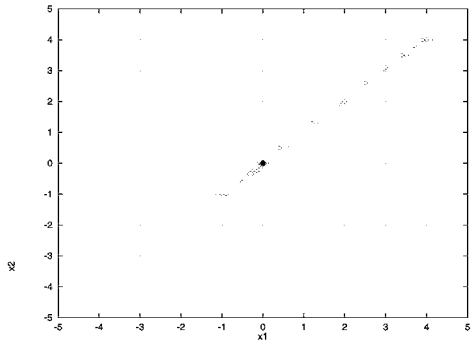



Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)

43

## Optimization Trajectory

initial step sizes $\vec{h}_{start} = (0.1, 0.1)$




$\vec{x}_{start} = (4,4)$

$\varepsilon = 0.01$

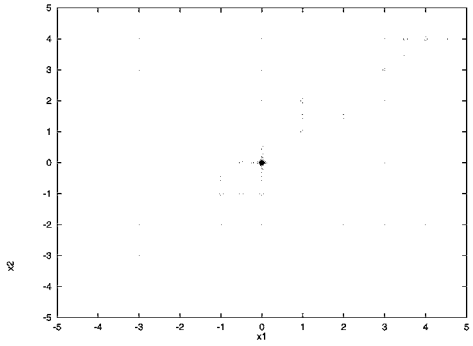
point-to-point optimization

Vorlesung „Simulationstechnik"
(Dr. M. Syrjakow)

44

## Optimization Trajectory

initial step sizes $\vec{h}_{start} = (0.5,0.5)$



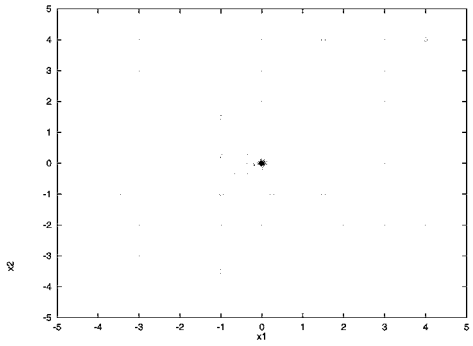

$\vec{x}_{start} = (4,4)$

$\varepsilon = 0.01$



## Optimization Trajectory

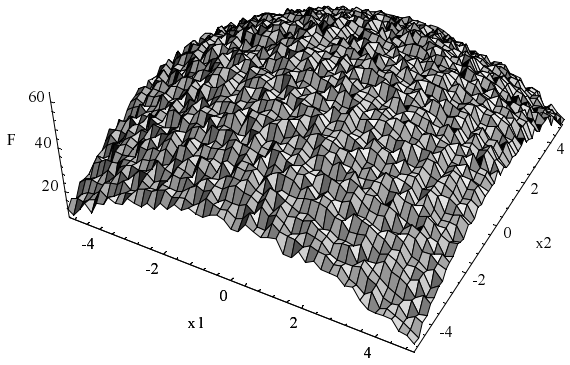
initial step sizes $\vec{h}_{start} = (2.5,2.5)$




$\vec{x}_{start} = (4,4)$

$\varepsilon = 0.01$

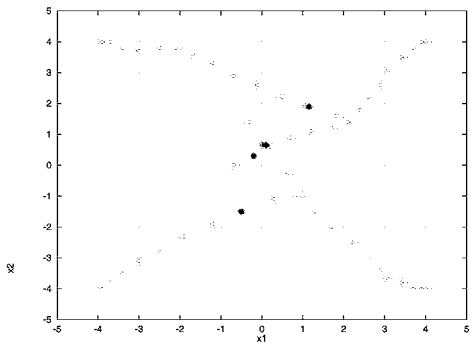
## Goal Function Overlayed with Stochastic Inaccuracies






## Optimization Trajectory

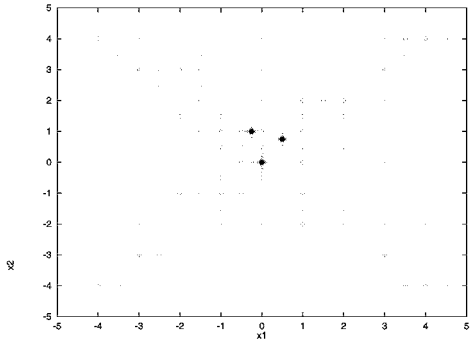
initial step sizes $\vec{h}_{start} = (0.1, 0.1)$




$\varepsilon = 0.01$

## Optimization Trajectory

initial step sizes $\vec{h}_{start} = (0.5, 0.5)$




$\varepsilon = 0.01$



## Additional Information

**Literature**

Hooke, R.A.; Jeeves, T.A.: Direct Search Solution for Numerical and Statistical Problems; Journal ACM 8, pp. 212-221, 1961.

## Other Direct Optimization Methods

- Evolution Strategies
  - closely related to Genetic Algorithms
  - literature: Schwefel, Hans-Paul: Evolution and Optimum Seeking; Wiley & Sons, 1995.
- Threshold Accepting, Deluge Algorithm, Record-to-Record Travel
  - based on Simulated Annealing
  - literature: Dueck, G.: New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel; in Journal of Computational Physics 104 (1993), pp. 86-92.
- Tabu Search
  - important links to evolutionary and genetic methods
  - literature: Glover, F.; Laguna, M.: Tabu Search; Kluwer Academic Publishers, Boston, 1997.