From Genetic Algorithms towards Hybrid Optimization

Alexey Poklonskiy, Martin Berz, Kyoko Makino

Dec, 18 2006

MICHIGAN STATE UNIVERSITY

What is Genetic Algorithm?

- *Family:* heuristic, stochastic methods
- *Inspiration:* computational analogy of **adaptive systems** modelled on the principles of the **evolution** via natural **selection**
- *Idea:* employing a **population** of individuals that undergo selection in the presence of variation-inducing operators such as **mutation** and recombination (**crossover**). A **fitness function** is used to evaluate individuals, and reproductive success varies with fitness
- *Applicability:* might not find the best solution, but often come up with a **partially optimal** solution. So, **not a rigorous method**, but usually finds **good fit**

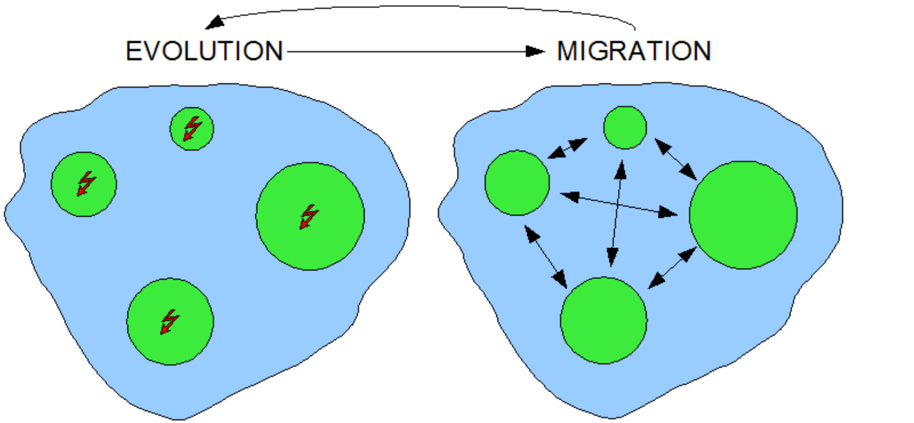
Algorithm

```
P(0) = random()
do
    For each m in P(t) compute Fitness(m)
    B(t) = best fitted individuals from P(t)
    while( Size(G(t)) < population size )
        Select parents p1,p2 from B(t)
        c = Crossover(p1,p2)
        Mutate(c) with certain probability
        Push(c, G(t))
    P(t+1) = G(t) OR select P(t+1) from B(t) and G(t)
while(satisfying solution not obtained OR
      number of steps less then threshold)
```

Variations

- Overlapping (steady-state GA) and non-overlapping (simple GA) populations
- Different representations, crossover and mutation operators
- Activator genes
- Alienated evolution (more than 2 parents)
- Random ”Evolutionary Bottlenecks”
- Parallelization
  - Evolutionary Divide and Conquer: explore problem subdivisions rather than the solutions
  - Master-slave (or global) parallel GAs (distributes the evaluation of the population among several processors)
  - Multiple-population GAs (coarse-grained or island model GAs)
- 2-level genetic algorithm

Island Model GA



Why

1. Relative simplicity of constrained optimization and technical implementation.
2. Good in avoiding local extrema.
3. Could possibly generate something new and better (artificial design).
4. Capable of both exploration (broad search) and exploitation (local search) of the search space.
5. When solving multi-objective problems, GA gives out many satisfactory solutions. Help in building Paretto front.
6. Very well suited for supporting the design and choice phases of decision making.

How and When

To gain benefits from Genetic Algorithms, one need to

1. **Effectively** encode solutions of a given problem to chromosomes in GA.
2. **Meaningfully** compare the relative performance (fitness) of solutions.

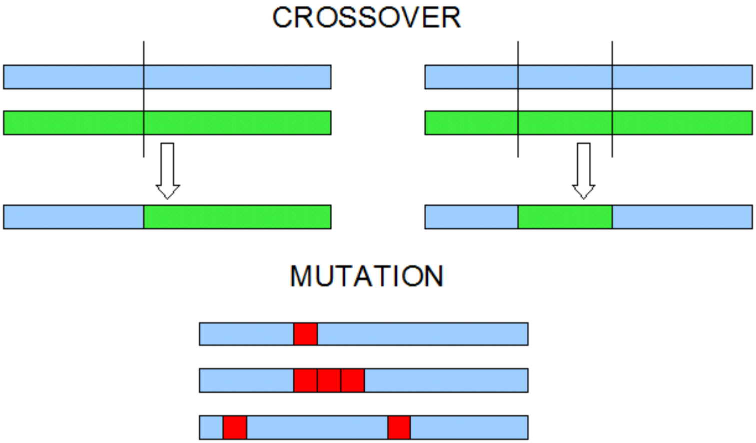
If those conditions are met GAs are useful and efficient when

1. The search space is large, complex or poorly understood.
2. Domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space.
3. No mathematical analysis is available.
4. Traditional search methods fail.

To apply GA for some problem

1. Select the particular GA.
2. Define a representation:
    - real number 1D, 2D, and 3D arrays
    - 1D, 2D, and 3D binary strings
    - lists
    - trees
3. Define the genetic operators.
    - Crossover
    - Mutation
4. Define the objective function.
5. Set the algorithm parameters (probabilities, rates, thresholds, flags).

Genetic operators: Crossover and Mutation



Critical factors

- Requires extensive fine-tuning
- It is possible to choose the 'right' representation but the 'wrong' genetic operator or vice versa
- Keeping balance between crossover and selection. If selection is very intense the population will converge very fast and there might not be enough time for good mixing to occur between members of the population. When this premature convergence occurs the GA may converge to a suboptimal population f the selection intensity is high, crossover might disrupt any good strings that may have already been found, but that have not had time to reproduce which may cause GA to not find optimal solution

MICHIGAN STATE UNIVERSITY

Examples of Successful Applications

- *Optimization:* wide variety of optimization tasks: numerical optimization, combinatorial optimization problems (TSP), circuit design, building schedules, video and sound quality optimization, optimal molecule configurations for particular systems like C60 (GARAGe).
- *Automatic Programming or Evolutionary Computing:* evolving computer programs for specific tasks, design of the other computational structures, e.g. cellular automates, sorting networks.
- *Machine and Robot Learning:* classification and prediction, designing neural networks, evolving rules for learning classifier systems and symbolic production systems, to design and control robots.
- *Economics:* modelling processes of innovation, the development of bidding strategies.
- *Ecology and Biology:* biological arms races, host-parasite co-evolutions, symbiosis and resource flow in nature, configuration applications, particularly physics applications of protein folding and protein/ligand docking (GARAGe).

MICHIGAN STATE UNIVERSITY

Example of Failure

To use a standard GA for TSP, the following problems have to be solved:

- Find good representation for tours
- Design appropriate fitness function which takes constraints into account

Straightforward solution:

- Permutation matrices

$$\mathrm{tour}(23541) = \mathrm{tour}(12354)$$

- Penalty-function method (non-permutation matrices represent unrealistic solutions).

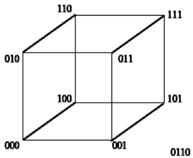
This approach generates too many invalid solutions and gives poor results.
Alternative solution:

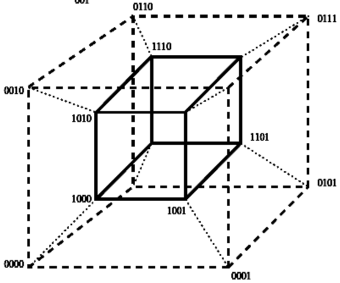
- new representations (Position Dependent Representations) and new genetic operators.
- ...

Why Do They Work?

John Holland, 1995, “Adaptation in Natural and Artificial Systems”: sampling hyperplane partitions in search space (being implemented properly)

Hypercubes

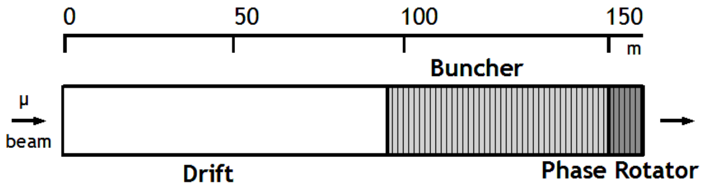

110
111
010
011
100
101
000
001


0110
0111
1110
0010
1010
1101
0101
1000
1001
0000
0001

Motivation

If there exists a specialized method of optimization for specific problem, then GA **may not be the best tool** for this application.
But it might be helpful to build **hybrid algorithm**.

- GA + GO = HO?
- beam dynamics optimization



MICHIGAN STATE UNIVERSITY

Rigorous GO + Non-rigorous GO

Rigorous GO has two main ingredients

1. heuristic updates of cutoffs based on searching promising regions
2. rigorous elimination of regions known to be above cutoff

- **Goal:** design and implement a GA that interlaces with rigorous GA for task 1
- **Most important:** dynamically restrict the population to all boxes not yet eliminated (Islands = Boxes)

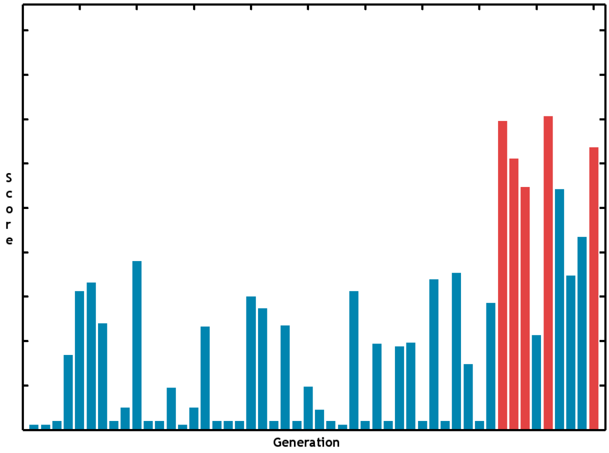
This is particularly easily formulated in a genetic optimizer

Part 1

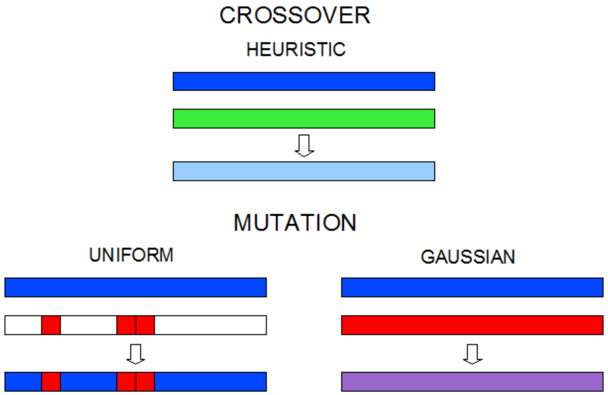
- **Representation:** vectors of real numbers
- **Fitness scaling:**
  - linear $\mathbf{f} \in \mathbb{R} \Rightarrow \hat{\mathbf{f}} \in \mathbb{R}^+ \Rightarrow \mathbf{fit} \in [0,1] : \sum \mathrm{fit}_i = 1$
  - rank $\mathbf{f} \in \mathbb{R} \Rightarrow \hat{\mathbf{f}} \in \mathbb{N} \Rightarrow \mathbf{fit} \in [0,1] : \sum \mathrm{fit}_i = 1$
- **Genetic operators:**
  - elitism
  - gaussian, uniform mutation
  - heuristic crossover

MICHIGAN STATE
UNIVERSITY

Elite members

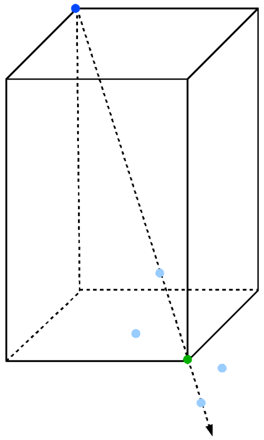


Genetic operators



Heuristic crossover

Heuristic crossover



Part 2

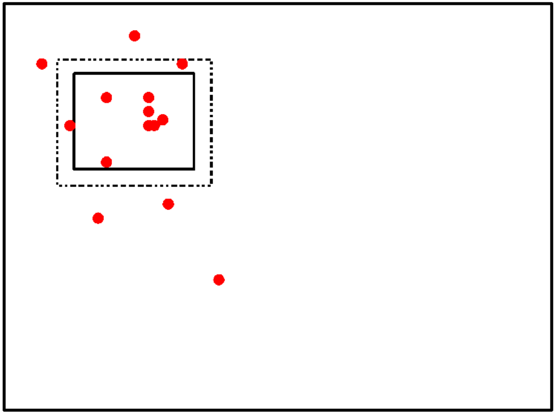
- **Selection:**
  - stochastic uniform
  - roulette
  - tournament
- **Stopping criteria:**
  - Maximum number of generations
  - Maximum number of stall generations + Tolerance
  - Pre-defined desired objective function value

Features

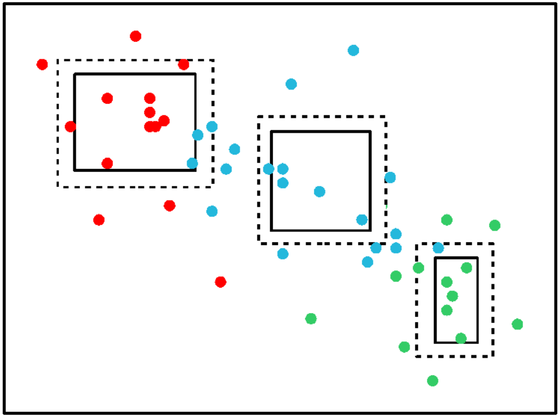
- Operates in the box(es), keep population in global box
- Interfaces with COSY-GO via reverse communication mode
- Fit into COSY FIT/ENDFIT syntax

```
FIT var1 var2 ... varN;
...
obj := ...
...
ENDFIT tolerance max_iterations algorithm obj;
```

Features

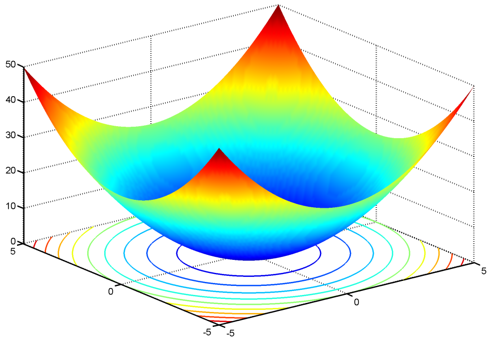


Features



Sphere function: definition

- *Definition:* $f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$
- *Search domain:* $x_i \in [-6, 6],\ i = 1, 2, \ldots, n$
- *Number of local minima:* no local minima, only global one
- *The global minimum:* $\mathbf{x}^* = (0, \ldots, 0),\ f(\mathbf{x}^*) = 0$

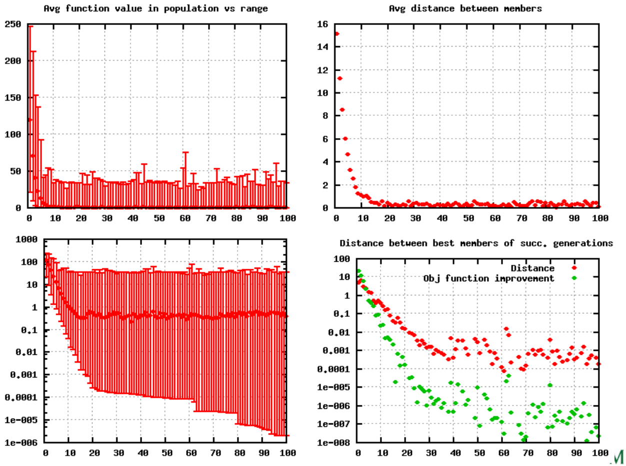


MICHIGAN STATE UNIVERSITY

Sphere function: algorithm parameters

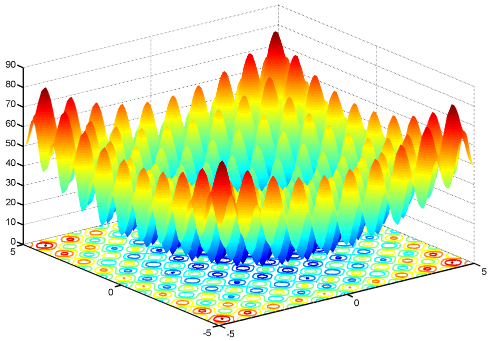
- N = 10
- Population size = 1000
- Initial population size = 0
- Reproduction params: Number of elite = 10, Mutation rate = 0.2
- Crossover params: Heuristic, Ratio = 0.8, Randomize On
- Fitness scaling: Rand
- Selection: Roulette
- Mutation params: Uniform, Gene Mutation Probability = 0.01
- Areal: $[-6.01250509, 6.01250509] \times N$, Killing On
- Max generations = 100
- **Best value** = 0.1984614290024165E-05
- **Time** = 0h 4m 2s

Optimization: optimization process



Rastrigin's function: definition

- *Definition:* $f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$
- *Search domain:* $x_i \in [-6, 6], \; i = 1, 2, \ldots, n$
- *Number of local minima:* several local minima
- *The global minimum:* $\mathbf{x}^* = (0, \ldots, 0), \; f(\mathbf{x}^*) = 0$

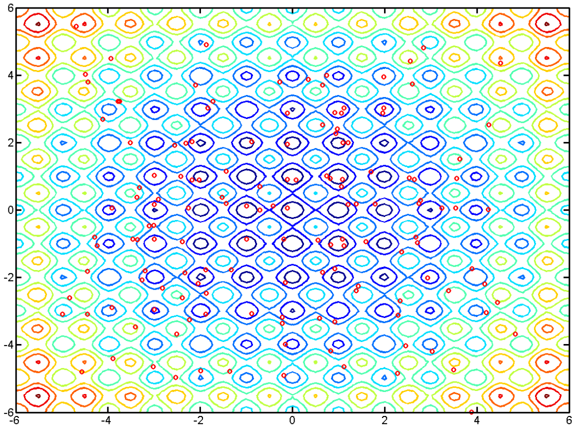


Rastrigin’s function: algorithm parameters

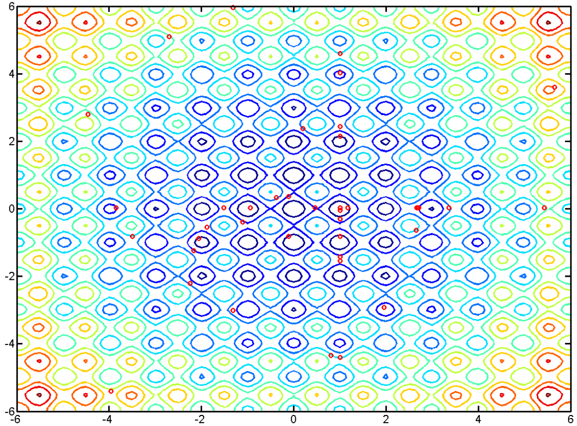
- N = 10
- Population size = 1000
- Initial population size = 0
- Reproduction params: Number of elite = 10, Mutation rate = 0.2
- Crossover params: Heuristic, Ratio = 0.8, Randomize On
- Fitness scaling: Rand
- Selection: Roulette
- Mutation params: Uniform, Gene Mutation Probability = 0.01
- Areal: $[-6.01250509, 6.01250509] \times N$, Killing On
- Max generations = 100
- **Best value** = 0.1001886961046239E-01
- **Time** = 0h 4m 43s

MICHIGAN STATE
UNIVERSITY

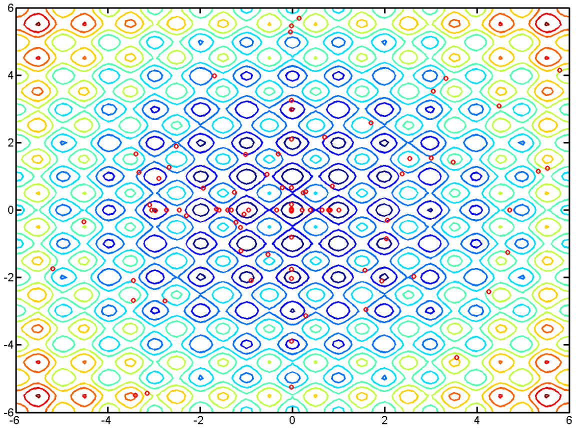
Rastrigin’s function, generation = 1



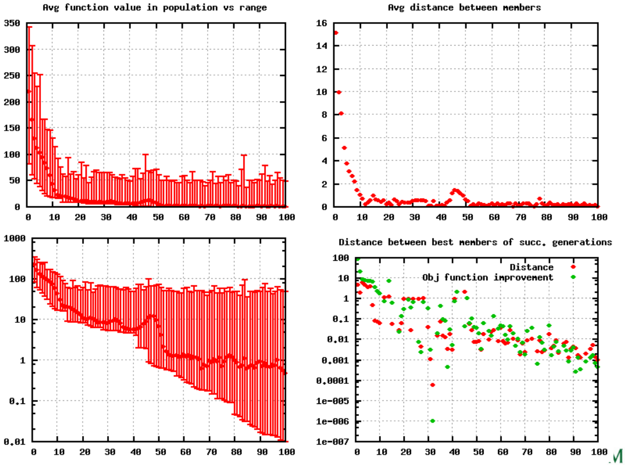
Rastrigin's function, generation = 10



Rastrigin's function, generation = 60



Rastrigin’s function: optimization process



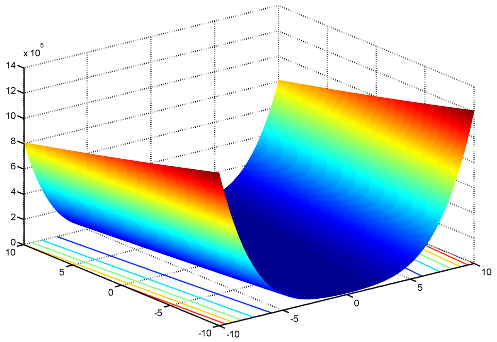
Rastrigin's function: different params

Different sets of parameters

Scaling	Elite	Mutation	Crossover	Result	Time
Rank	10	Unif(0.01)	Heur(0,8, 1)	0.196	0h 4m 27s
Rank	10	**Gauss(1,1)**	Heur(0,8, 1)	3.082	0h 4m 25s
Rank	10	Unif(0.01)	Heur(0,8, **0**)	0.100E-01	0h 4m 43s
Rank	10	Unif(**0.1**)	Heur(0,8, 1)	0.593E-02	0h 4m 30s
Rank	**0**	Unif(**0.1**)	Heur(0,8, 1)	0.125E-03	0h 4m 29s
Linear	**0**	Unif(**0.1**)	Heur(0,8, 1)	7.4327	0h 4m 1s

Rosenbrock's function: definition

- *Definition:* $f(\vec{x}) = \sum_{i=1}^{n-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right)$
- *Search domain:* $x_i \in [-5, 10],\ i = 1, 2, \ldots, n$
- *Number of local minima:* several local minima
- *The global minimum:* $\mathbf{x}^* = (1, \ldots, 1),\ f(\mathbf{x}^*) = 0$

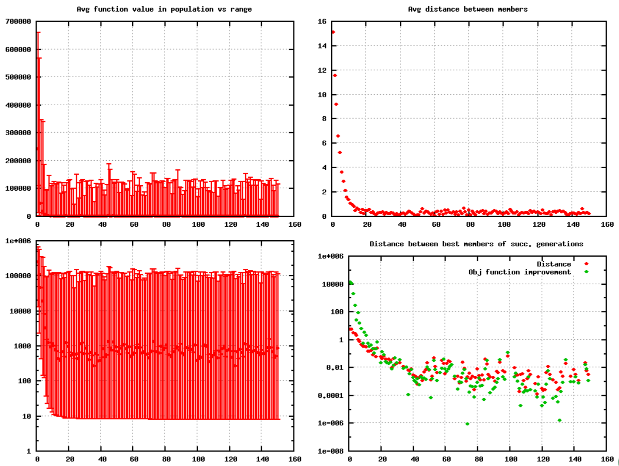


MICHIGAN STATE UNIVERSITY

Rosenbrock's function: algorithm parameters

- N = 10
- Population size = 1000
- Initial population size = 0
- Reproduction params: Number of elite = 10, Mutation rate = 0.2
- Crossover params: Heuristic, Ratio = 0.8, Randomize On
- Fitness scaling: Rand
- Selection: Roulette
- Mutation params: Uniform, Gene Mutation Probability = 0.01
- Areal: $[-6.01250509, 6.01250509] \times N$, Killing On
- Max generations = 150
- **Best value** = 7.940674306488130
- **Time** = 0h 6m 46s

Rosenbrock’s function: optimization process



MICHIGAN STATE UNIVERSITY

Conclusion

- GA framework for optimization of the real-valued functions is implemented in COSY Infinity and will be added as one of the available built-in optimizers
- GA work well in finding “good fits”
- GA can be combined efficiently to obtain cutoff updates for rigorous GO

MICHIGAN STATE
UNIVERSITY