From Genetic Algorithms towards Hybrid Optimization

Alexey Poklonskiy, Martin Berz, Kyoko Makino

Dec, 18 2006
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
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.
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 if 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
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).
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
  \[
  \text{tour}(23541) = \text{tour}(12354)
  \]
- 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
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
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 $f \in \mathbb{R} \Rightarrow \hat{f} \in \mathbb{R}^+ \Rightarrow \text{fit} \in [0, 1] : \sum \text{fit}_i = 1$
  - rank $f \in \mathbb{R} \Rightarrow \hat{f} \in \mathbb{N} \Rightarrow \text{fit} \in [0, 1] : \sum \text{fit}_i = 1$
- **Genetic operators:**
  - elitism
  - gaussian, uniform mutation
  - heuristic crossover
Elite members
Genetic operators

Crossover
- Heuristic

Mutation
- Uniform
- Gaussian
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:** \( f(x) = \sum_{i=1}^{n} x_i^2 \)
- **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:** \( x^* = (0, \ldots, 0), \ f(x^*) = 0 \)
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:** \( f(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i)) \)
- **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
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

<table>
<thead>
<tr>
<th>Scaling</th>
<th>Elite</th>
<th>Mutation</th>
<th>Crossover</th>
<th>Result</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>10</td>
<td>Unif(0.01)</td>
<td>Heur(0,8, 1)</td>
<td>0.196</td>
<td>0h 4m 27s</td>
</tr>
<tr>
<td>Rank</td>
<td>10</td>
<td>Gauss(1,1)</td>
<td>Heur(0,8, 1)</td>
<td>3.082</td>
<td>0h 4m 25s</td>
</tr>
<tr>
<td>Rank</td>
<td>10</td>
<td>Unif(0.01)</td>
<td>Heur(0,8, 0)</td>
<td>0.100E-01</td>
<td>0h 4m 43s</td>
</tr>
<tr>
<td>Rank</td>
<td>10</td>
<td>Unif(0.1)</td>
<td>Heur(0,8, 1)</td>
<td>0.593E-02</td>
<td>0h 4m 30s</td>
</tr>
<tr>
<td>Rank</td>
<td>0</td>
<td>Unif(0.1)</td>
<td>Heur(0,8, 1)</td>
<td>0.125E-03</td>
<td>0h 4m 29s</td>
</tr>
<tr>
<td>Linear</td>
<td>0</td>
<td>Unif(0.1)</td>
<td>Heur(0,8, 1)</td>
<td>7.4327</td>
<td>0h 4m 1s</td>
</tr>
</tbody>
</table>
Rosenbrock’s function: definition

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