A Synthetic review of Evolutionary algorithms and their applications in image analysis

Workshop on Machine-Learning-Assisted Image Formation

nice,

- Frederic Ros,
  PRISME Laboratory, Orleans University,
  Head of Orleans Technopole (digital incubator the LAB’O at Orléans)

- Serge Guillaume, UMR Itap, Irstea Montpellier
Agenda

• Evolutionary algorithms: general introduction
• Genetic algorithms: principle and basic algorithm
• Particle swarm optimization: principle and basic algorithm
• Multi objective Algorithms: few words
• Applications in image: several cases
• Conclusion
The problem

• Many interesting optimization problems are not trivial.
• The optimal solution cannot always be found in polynomial time.

-Multimodal
-The size of the search space grows exponentially!
The Optimization Problem

- **Exact algorithms**: brute force, branch and bound...optimal solutions if no limit on time and memory

- **Deterministic Methods** can fail because they could converge to local optimum

- **Heuristic algorithms**: ant colony, genetic algorithms...do not guarantee but they can give **good answers** relatively quickly

- **Evolutionary Algorithms** can however fail because they could converge to a sub-optimal solution

- Analogy: read a book in 1 month or 5 days
Solution: Parallel search concept

• Conduct searching in different areas simultaneously.
  • Population Based
  • Avoid unfortunate starting positions.

• Employ heuristic methods to effectively explore the space.
  • Focus on promising areas.
  • Also keep an eye on other regions.

• This is where EAs come into play!

EAs are Interesting for optimization and classification

• Can be used to solve many problems, and many kinds of problems, with minimal adjustments, without knowing optimum

• Are fast and easy to implement
“Dialects” Developing in Artificial Intelligence

• Fogel Owens (USA, 1965) **Evolutionary Programming** (fixed structure)
• Holland Genetic Algorithms (USA, 1973) **Genetic Algorithm**
• Rechenberg Schwefel (Germany, 1973) **Evolution Strategies**
• 90s Evolutionary Algorithms (EA)
• **EA Family:** **GA** (genetic algorithms), **PSO** (particle swarm optimization), **ACO** (ant colony), **BCO** (bee colony), **GWO** grey wolf (2014), **CSO** (cuckoo search) (2009), **GSA** (gravitational search algorithm) (2009) ...
• DE: Differential Evolution (new candidates: by combining existing ones)
• EDA: Estimation of Distribution Algorithm (heuristics, GA, sampling)
• Memetic algorithms (hybrid)
Some assertions

• **Culture and Cognition Summary**
  - Individuals searching for solutions learn from the experiences of others (individuals learn from their neighbors)
  - An observer of the population perceives phenomena of which the individuals are the parts (individuals that interact frequently become similar)
  - Culture affects the performance of individuals that comprise it (individuals gain benefit by imitating their neighbors)

• **So, what about intelligence?**
  - Social behavior increases the ability of an individual to adapt
  - There is a relationship between adaptability and intelligence
  - Intelligence arises from interactions among individuals
• Search is **directed** toward regions that are *likely* to have higher fitness values, through metaheuristics.

**Framework**

- **Problem Encoding**
  - Coding of solution
  - Objective function
  - Operators
  - Knowledge

- **Implementation**

**Differences to classical algorithms:** parallel search, straightforward to apply (direct fitness), able to apply self adaptation, use probabilistic rather than deterministic transition rules, several solutions can be provided.
The main inspiration of the ACO algorithm comes from stigmergy.

Stigmergy refers to the interaction and coordination of organisms in nature by modifying the environment.

The key points: no centralized control, probabilistic approach.
Genetic algorithms
Genetic algorithms

It is one of several evolutionary algorithms incorporating the idea of sexual reproduction, or «genetic recombination».

Background
Charles Darwin 1859: Writes Origin of Species and rocks the worlds of science and philosophy
Nils Aall Barricellu 1954: First emulates evolution on a computer
Ingo Rechenberg 1960s: Popularizes genetic algorithms as a tool for optimization
Holland 1975s: artificial systems
Genetic algorithms

It is inspired by the natural selection and mimics the biological evolution.

We start from a group of solutions (initial population). Those solutions are then combined to produce the offsprings - the next generation of (probably) better solutions.

New solutions are made from old ones using Crossover, Mutation and Selection just like Nature does.
Genetic algorithms

**Biggest advantage:** You do not need to know how to solve the problem. You just have to be able to evaluate the quality of the generated solution coming and through iteration you get good solution.

**Evolution and selection process:** Almost same for all kind of problem

**Fitness function and chromosome design:** problem specific
Basic components of genetic algorithms

• **Representation**
  - How to encode the parameters of the problem?
  - Binary Problems
    - 10001 00111 11001 ..... 
  - Continuous Problems (vector, matrix...)
    - 0.8 1.2 -0.3 2.1 ..... 
  - Hybrid: 001100 0.8 1.2 -0.3 2.1 ..... 
  - Matrices, graphs...

• **Population**
  - A set of individuals
  - GAs maintain and evolve a population of individuals.
  - Parallel Search → Global Optimization

• **Fitness function**
  - Gives a score to each state

• **Selection Strategy**
  - Which chromosomes should be involved in reproduction?
  - Which offspring should be able to survive?
  - Several approaches: roulette, tournament, generation gap approach, elitist...

• **Genetic Operators**
  - **Crossover:**
    - Exchange genetic materials between two chromosomes.
  - **Mutation:**
    - Randomly modify gene values at selected locations.
The fitness function produces the next generation of stages
The fitness function gives a score to each state
The probability of being chosen for reproduction is based on the fitness score.

\[ f(x,y) = x^2y + 5xy - 3y^2 \]

for what integer values of \( x \) and \( y \) is \( f(x,y) \) minimal?

Direct

Feature selection for classification Direct (based on classifier) Direct

Clustering: hybrid chromosome and cost function Direct

TPS problem: indirect
The chromosome is subjected to a random small modification without interaction with the others:

\[ y' = y (1 + \cdot N(0, \sigma)) \]

For binary representation:

```
1001001101001
```

For non-binary representation:

```
10010011001001
```

For other representations (matrix, graphs..), the operator depends on domain knowledge.
Crossover

\[ \text{ch'}_1 = a \cdot \text{ch}_1 + b \cdot \text{ch}_2 \]
\[ \text{ch'}_2 = b \cdot \text{ch}_1 + a \cdot \text{ch}_2 \]

Example for non binary representation

**Binary representation (classic)**
- For each pair to be mated, a crossover point (or more) is chosen at random from within the chromosome (binary).

- Offspring are created by exchanges between the parents at the crossover point.

**Uniform crossover** (ith allele)

\[ \text{ch'}_1(i) = \text{ch}_1(i) \text{ and } \text{ch'}_2(i) = \text{ch}_2(i) \quad \text{if } p_i > 0.5 \]
\[ \text{ch'}_1(i) = \text{ch}_2(i) \text{ and } \text{ch'}_2(i) = \text{ch}_1(i) \quad \text{if } p_i < 0.5 \]

\[ \begin{array}{c}
\text{parents} \\
10010011101001 \\
01110100101101 \\
\end{array} \quad \rightarrow \quad \begin{array}{c}
\text{children} \\
10010010101001 \\
01110011101101 \\
\end{array} \]

\[ p_i > 0.8 \]
Popular approaches:

**Baker’s method:** Use roulette wheel with \( n \) pointers spaced \( 1/n \) apart; use normalized fitness; spin wheel *once*.

**Tournament selection** - Select two individuals at random; the individual with the higher fitness is selected for the next population

**Generation gap approach:** Replace \( x \) percent that have worst fitness values (\( x \) is defined as the generation gap)

**Elitist strategy:** ensures that individual with highest fitness is copied into next generation (most GAs use this)
Genetic algorithms: parameters

**The crossover rate**
- The crossover operator is applied with a probability $P_c$.
- The higher is the rate the more novel chromosomes are introduced.

**The mutation rate**
- The mutation operator is applied with a probability $P_m$.
- If too big, the search is random, the evolution process is disrupted.
- If too small less diversification and then stagnation risk.

**The population size**
- If too high:
  Diversity grows but slow convergence rate, Convergence toward a local optimum diminishes.
- If too small:
  Risk of premature convergence (local minimum).

**Stopping criterion**
A minimum score, number of generations, time...

Exploration versus exploitation
A few more words

- **No so easy!**
  - So called “optimal” parameter values do not exist!
  - Vary from problems to problems.
  - Need some trials to find suitable parameter values.

- **Randomness**
  - Inherently stochastic algorithms
  - Independent trials are needed for performance evaluation.

- **Why does it work?**
  - Easy to understand & implement (No maths required!)
  - Very difficult to analyse mathematically.
  - Converge to global optimum with probability 1 (infinite population).

- **The addons!**
  - Multi – chromosomes: difficulty to « code » the problem
  - Niching / Sharing: facilitate the maintain of the diversity
  - Auto-adaptatif (Pmut, Pcross, population, chromosome, ...)
  - Hybridation: a local algorithm is used to promote a subset of candidate.
Basic Framework

**Initialization:** Generate a random population $P$ of $n$ chromosomes

**Evaluation:** Evaluate the fitness $f(x)$ of each chromosome

**Repeat until the stopping criteria are met:**

**Reproduction:** Repeat the following steps until all offspring are generated

  - **Parent Selection:** Select parents from $P$
  - **Crossover:** Apply crossover on the parents with probability $P_c$
  - **Mutation:** Apply mutation on offspring with probability $P_m$
  - **Evaluation:** Evaluate the newly generated offspring

**Offspring Selection:** Create a new population from offspring and $P$

**Output:** Return the best individual found

*Picture inspired from Dr. Bo Yuan*
Partial Swarm Optimization
The Particle Swarm Optimization Algorithm

PSO is initially developed by

James Kennedy
Social Psychologist

Russell C. Eberhart
Electrical Engineer

The DNA: combines self-experiences with social experiences
Intuition of PSO

Inspired from particle Swarm Optimization in MATLAB - Yarpiz Video
PSO Origin & Concept

In 1986, Craig Reynolds described this process in 3 simple behaviors of bird, fish..

Uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution.

Each particle in search space adjusts its “flying” according to its own flying experience as well as the flying experience of other particles.

Each particle modifies its position according to: its current position, current velocity, the distance between its current position, its best position, the distance between its current position and the best position, the best position.
PSO: concept and framework

Initialize the controlling parameters $\alpha, \beta, \gamma$
Initialize the population

do
  for each particle
    Calculate the fitness of the particle
    Update $p_{best}$ if required
    Update $g_{best}$ if required
  end for
  Update the inertia weight (option)
  for each particle
    Update the velocity ($v$)
    Update the position ($p$)
  end for
while the end condition is not satisfied

$v_{i}^{t+1} = \alpha v_{i}^{t} + \beta (p_{b_{i}}^{t} - p_{i}^{t}) + \gamma (g_{b}^{t} - p_{i}^{t})$
$p_{i}^{t+1} = p_{i}^{t} + v_{i}^{t+1}$

Inertia: Makes the particle move in the same direction and with the same velocity
Personal influence: Makes the particle return to a previous position, better than the current conservative
Social influence: Makes the particle follow the best neighbors’ direction
Algorithm Characteristics

**Advantages**
- Insensitive to scaling of design variables
- Simple implementation
- Easily parallelized for concurrent processing
- Derivative free
- Very few algorithm parameters
- Very efficient global search algorithm

**Disadvantages**
- Tendency to a fast and premature convergence in mid optimum points
- Slow convergence in refined search stage

**Adons**
- Adaptive PSO, Adaptive Mutation PSO
- Adaptive PSO Guided by Acceleration Information
- Attractive Repulsive Particle Swarm Optimization
- Chaotic PSO, fuzzy PSO...
- Cooperative Multiple PSO
- Dynamic and Adjustable PSO
- Extended Particle Swarms, hybrid
Few words about multi objective
Multi objectives: few words

- Extension of regular EA which maps multiple objective values to single fitness value
- Objectives typically conflict
- In a standard EA, an individual \( A \) is said to be better than an individual \( B \) if \( A \) has a higher fitness value than \( B \)
- In a MOEA, an individual \( A \) is said to dominate individual \( B \) iff:
  - \( A \) is no worse than \( B \) in all objectives
  - \( A \) is strictly better than \( B \) in at least one objective
Multi objectives

- **The process:** update the individuals for Pareto Improvement.

- An allocation is **Pareto Optimal** when no further Pareto Improvements can be made. This is often called a **Strong Pareto Optimum** (SPO).

- The non-dominated subset of the entire feasible search space $S$ is the globally **Pareto-optimal set**.

Source: wikipedia
Simple Multiobjective EAs

- **SEMO**: Each individual in the population is selected with the same probability (uniform selection)
- **FEMO**: Select individual with minimum number of mutation trials (fair selection)
- **GEMO**: Priority of convergence if there is progress (greedy selection)

- flip randomly chosen bit
- select individual from population
- insert into population if not dominated
- Remove dominated solutions from population

*Inspired from tutorial’ Eckart Zitzler on evolutionary optimization algorithms*
Simple Evolutionary Multiobjective Optimizer

1. Start with a random solution
2. Choose parent randomly (uniform)
3. Produce child by variating parent
4. If child is not dominated then
   • add to population
   • discard all dominated
5. Goto 2

Inspired from tutorial’ Eckart Zitzler on evolutionary optimization algorithms
Multi-Objective Optimization (EMOO) approaches state of the art

Non-Pareto Techniques
- Aggregating approaches
- VEGA (Vector Evaluated Genetic Algorithm)

Pareto Techniques
- Pure Pareto ranking
- NSGA (Non-dominated Sorted Genetic Algorithm-II, K. Deb): diversity-preserving strategy via crowding...
- MOGA (Multi-Objective Genetic Algorithm)
- Ant-colony based

Recent Approaches
- PAES (The Pareto Archived Evolution Strategy, Knuoles & Corne), maintains an archive population of non dominated solutions.
- SPEA (Strength Pareto Evolutionary Algorithm 2, Zitzler) use of an external population and clustering...

Bio-inspired Approaches
- PSO (particle swarm optimization)
- Ant-colony based

General idea: ensure a spread among the non dominated solutions while minimizing the distance to the optimal front

Pioneering techniques are around 15-20 years old, code sources of relevant techniques are available on the net
Evolutionary algorithms in practice
**Evolutionary algorithms in practice**

- **Optimization problems**: many uses in logistic (cf. Tsp problem..)

<table>
<thead>
<tr>
<th></th>
<th>Evolutionary Algorithms</th>
<th>Ad-hoc Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Slow *</td>
<td>Generally fast</td>
</tr>
<tr>
<td>Human work</td>
<td>Minimal</td>
<td>Long and exhaustive</td>
</tr>
<tr>
<td>Applicability</td>
<td>General</td>
<td>There are problems that cannot be solved analytically</td>
</tr>
<tr>
<td>Performance</td>
<td>Excellent</td>
<td>Depends</td>
</tr>
</tbody>
</table>

- **Select** a subset of features (remove irrelevant and redundant features...), instance (sampling).
- **Generate** parameters of a given algorithm (segmentation, clustering, Anns...)

**Domain fields**: optimal control & design, finance, logistic, chemical engineering.. but relatively few in image (genetic & pso)!
Image discrimination using texture parameters

Classical uses: parameter selection & optimization

Classification of osteoporosis using X-ray images

Objective: select texture parameters

\[ \text{Chromosome} = [0,0,1,1,1,..,0.2,..] \Rightarrow 2 \text{ parts, binary and continue} \]

Fitness function = classifier(knn, anns, SVM...)

Result: rate > 10% on test (using a basic GA)

Image tasks

Multi thresholding (direct application of Otsu approaches and derivates)

Noise reduction

Pick a window size of $n \times n$
Use a real-valued chromosome of length $n^2$
Use a blending crossover
Use a modest mutation operator
Apply mask to noisy images
Set fitness to the Euclidean distance between original images and filtered noisy images

*In this case, you get what you evolve for!*

Clustering & genetic algorithms

**Chromosome**

\[
\begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 & \vdots & 53.2 & 19.6 & 34.7 & 68.2 & 75.3
\end{bmatrix}
\]

**Fitness function**

\[
\min z = \sum_{i=1}^{3} \sum_{(x,y) \in R_i} \left[ \text{Rep}(R_i) - f'(x,y) \right]^2
\]

**Crossover**: Uniform crossover

**Mutation**: bit flip and \( y' = y (1 + \text{N}(0, \sigma)) \)

---

**Table**:

<table>
<thead>
<tr>
<th></th>
<th>Actual pixels</th>
<th>D-thresholding</th>
<th>CHNN</th>
<th>( k )-Means</th>
<th>Fuzzy ( c )-means</th>
<th>Via GA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Segmented pixels</td>
<td>Misclassified rate (%)</td>
<td>Segmented pixels</td>
<td>Misclassified rate (%)</td>
<td>Segmented pixels</td>
<td>Misclassified rate (%)</td>
</tr>
<tr>
<td>T5</td>
<td>3282</td>
<td>2106</td>
<td>35.83</td>
<td>2566</td>
<td>21.82</td>
<td>2594</td>
</tr>
<tr>
<td>T4</td>
<td>8745</td>
<td>9045</td>
<td>3.43</td>
<td>8810</td>
<td>0.70</td>
<td>8905</td>
</tr>
<tr>
<td>T3</td>
<td>9040</td>
<td>9004</td>
<td>0.40</td>
<td>9293</td>
<td>2.80</td>
<td>9158</td>
</tr>
<tr>
<td>T2</td>
<td>5590</td>
<td>7624</td>
<td>34.60</td>
<td>6631</td>
<td>18.62</td>
<td>6649</td>
</tr>
<tr>
<td>T1</td>
<td>38,879</td>
<td>37,857</td>
<td>2.6</td>
<td>38,236</td>
<td>1.65</td>
<td>38,230</td>
</tr>
<tr>
<td><strong>Average error</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes**:
- Competitive hopfield neural network
- Abdominal image
- Skull
Filter design (Finite impulse response)

Frequency space
Real Chromosome (vector, matrix of weights)
Find the components to match with the ideal frequential filter (MSE)

\[ E = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} [D(\omega_{1j}, \omega_{2k}) - F(\omega_{1j}, \omega_{2k})]^2 \]

\[ F(\omega_1, \omega_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} a(k_1, k_2) \cos(\omega_{1j}k_1) \cos(\omega_{2k}k_2) \]

\( D \) is the ideal frequency and \( F \) is the current response.

\( a(k_1, k_2) \) is the matrix component. Filter of \( N_1 \times N_2 \) dimensions.

\[ P_1 = \frac{(N_1-1)}{2} \quad P_2 = \frac{(N_2-1)}{2} \]

Fitness function: \( f = 1/(1+E) \)

Crossover

\[ \begin{align*}
B_{C}^{Chdd1} &= (B_i - B_j) \ast \lambda_{c1} + B_i \\
B_{C}^{Chdd2} &= (B_j - B_i) \ast \lambda_{c2} + B_j 
\end{align*} \]

Adaptative mutation

\[ P_{M_i} = \begin{cases} 
\frac{1 + \text{fit}_{\text{max}} - \text{fit}_i}{\text{fit}_{\text{max}} - \text{fit}_{\text{avg}}} \text{ fit}_i > \text{fit}_{\text{avg}} \\
\lambda_m \text{ fit}_i < \text{fit}_{\text{avg}} 
\end{cases} \]

Mutation

\[ B_{\text{New}}^{\text{add}} = B^{\text{add}} \ast (1 + \lambda_m) \]

where \( \lambda_m \) represents the level of modification and \( B \) defines each chromosome component.

Other uses: quadtree

Fitness function:

\[ f = \lambda \sum_{i=1}^{n} m_i \cdot (255 - \sigma_i) / \left( \sum_{i=1}^{n} m_i + \mu \cdot 255 \cdot (1 - (n/s)) \right) \]

- \( n \) (number of sub images), \( m_i \& \sigma_i \) (statistics), \( s \) chrom size
- \( \lambda, \mu \): parameters

Crossover
Segmentation with thermal images (active contours)

**Objective:** Segmentation

**Chromosome** = vector of discrete points from a Cassini model

Np+ 5 parameters: $C_x, C_y, H_x, H_y, \text{Teta (rst)}$

**Fitness function** = $ftn (i) = \lambda_1 \Delta_c + \lambda_2 \Delta_r$

$\lambda_1 + \lambda_2 = 1$

$\Delta_c$: contour information

$\Delta_r$: region information

---

**Chan et Vese**

**Branch and Mincut**

**Hybrid HGA2**

**Cassini model**

**Hybrid model HGA2**
Segmentation medical images

Active contours: prostate identification

- Ideal contours using a learning set of images

- Each contour has a unique shape and a pose (size, position and orientation).

- One modelises a medium shape and internal texture (including the variabilities).

- A population of chromosomes I(w,p) evolves until convergence (w: weight of the k textural descriptors, p: pose for the rst). The parameters w and p are generated in the possible space.

- The evaluation is done by matching the internal « texture » of each detected object with the medium « texture » found during the training phase.

Payel Gosh, Segmentation of medical images using genetic algorithms, 2006
Synthesis: art or science?

When to use EA?
• When space to be searched is large
• When the “best” solution is not necessarily required
• Approach to solving a problem not well-understood
• Problems with many parameters that need to be simultaneously optimized
• Problems that are difficult to describe mathematically

Drawbacks
• Difficult to find an encoding for the problem
• Difficult to define a valid fitness function
• May not return the global maximum, risk of premature convergence
• Exploration versus exploitation
• Can be time consuming.