Metaheuristics for Optimization

E-G. Talbi

http://paradiseo.gforge.inria.fr
Introduction (1)

- High-dimensional and complex optimization problems in many areas of industrial concern → Telecommunication, Computational biology, Transportation and Logistics, Design, ...
- Problems of increasing size (combinatorial explosion)
  - Getting near-optimal solutions in a tractable time
- Using approached methods isn’t sufficient
  - Metaheuristics approach
  - Hybridization features
  - Large scale parallelism (Cluster & GRID Computing)
Popular NP-hard problems

- Permutation problems: traveling salesman, scheduling, ...
- Assignment: QAP, GAP, ...
- Grouping: partitioning, clustering, graph coloring ...
- Routing: VRP, CTP, ...
- Knapsack and packing
- and many more, etc
Tackling an academic COP. The Traveling Salesman Problem

- “Given a collection of N cities and the distance between each pair of them, the TSP aims at finding the shortest route visiting all of the cities”
- Symmetric TSP: \( \frac{(N-1)!}{2} \) candidate solutions
- Example

```
 1  8  10  4  3
V₀
/   /   /   /   /
|   |   |   |   |
V₁  V₂  V₃  V₄
8  4  10  6
6  6  6  6
6  6  9  4
3  5  4  4
```

Length: 26
Tackling an academic COP. The Traveling Salesman Problem

usa13509
Tackling a real-world COP. Design of cellular radio networks

- Financial context (cost of the network)
  - Number of sites
  - Quality of Service

- Network design
  - Positioning sites
  - Fixing a set of parameters for each antenna

- Very highly combinatorial (NP-hard)
A practical hard problem

- Three main features
  - A high number of potential configurations
    \( \rightarrow \equiv 4.8 \times 10^{2558}, 8.4 \times 10^{6494} \) and \( 5.5 \times 10^{8541} \) candidate solutions (on different instances Arno 1.0, 3.0 et 3.1)
  - A CPU cost evaluation function (trigonometric functions, sorting algorithms, ...).
  - Need of a large amount of memory
    \( \rightarrow 512 \text{ Mo.}, 1 \text{ Go.} \) and \( 2 \text{ Go.} \).
Introduction (2)

- Combinatorial Optimization Problems (COPs) in practice
  - Diversity
  - Continual *evolution* of the *modeling* (regards needs, objectives, constraints, ...)
  - Need to experiment many solving methods, techniques of hybridization, parameters, ...
Motivations

- A framework for the design of parallel hybrid metaheuristics dedicated to the mono/multi-objective resolution of COPs
  - Identifying abstract/specific features of both metaheuristics and main models of parallelization and hybridization
  - Insuring transparence of parallelism
    - Easily deploying on sequential architectures, parallel/distributed platforms and meta-computing grids
  - Validating the framework by tackling hard and real applications (modeling and solving)
Taxonomy (optimization methods)

- **Exact methods**: optimality and exploitation on small instances
- **Heuristics**: near-optimal solutions on large-size problem instances
Classification of metaheuristics

- Nature inspired vs. non nature inspired
- Population-based vs. single point search
- Dynamic vs. static objective function
- One vs. various neighborhood structures
- Memory usage vs. memory less methods
- Iterative vs. Greedy
- ...

Paradiseo
Single solution metaheuristics are *exploitation* oriented
Population-based metaheuristics are *exploration* oriented
Taxonomy (Population-based Metaheuristics)

Metaheuristics

Population

- Evol. algorithms
- Scatter search
- Ant colony

- Evol. programming
- Evol. strategies
- Genetic algorithms
- Genetic programming
History

• L. Fogel 1962 (San Diego, CA): *Evolutionary Programming*

• J. Holland 1962 (Ann Arbor, MI): *Genetic Algorithms*

• I. Rechenberg & H.-P. Schwefel 1965 (Berlin, Germany): *Evolution Strategies*

• J. Koza 1989 (Palo Alto, CA): *Genetic Programming*
The metaphor

<table>
<thead>
<tr>
<th>Evolution</th>
<th>Problem solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual</td>
<td>Candidate Solution</td>
</tr>
<tr>
<td>Fitness</td>
<td>Quality</td>
</tr>
<tr>
<td>Environment</td>
<td>Problem</td>
</tr>
</tbody>
</table>

- Based on the evolution of a population of individuals
- Evolution features
  - Variation operators (crossover, mutation) to increase diversity,
  - Selection of parents, replacement by offspring to decrease diversity
The ingredients

mutations

recombination

reproduction

selection

$t$

$t + 1$

mutation

recombination
Evolutionary cycle

1. Selection
2. Parents
3. Crossover
4. Mutation
5. Offspring
6. Replacement
7. Population
Evolutionary Algorithm procedure

- $t \leftarrow 0$
- $P \leftarrow \text{GenerateInitialPopulation}()$
- evaluate ($P_t$)
- do
  - Select $P_{t+1}$ from $P_t$
  - Transform ($P_t$)
  - Evaluate ($P_t$)
  - $P_{t+1} \leftarrow \text{Replace} (P_t, P_{t+1})$
  - $t \leftarrow t + 1$
- While termination conditions not met
Domains of application

- Numerical, Combinatorial Optimisation
- System Modeling and Identification
- Planning and Control
- Engineering Design
- Data Mining
- Machine Learning
- Artificial Life
- ...

Paradiseo
Performances

- Acceptable performance at acceptable costs on a wide range of problems
- Intrinsic parallelism (robustness, fault tolerance)
- Superior to other techniques on complex problems with
  - lots of data, many free parameters
  - complex relationships between parameters
  - many (local) optima
Advantages

- No presumptions w.r.t. problem space
- Widely applicable
- Low development & application costs
- Easy to incorporate other methods
- Solutions are interpretable (unlike NN)
- Can be run interactively, accommodate user proposed solutions
- Provide many alternative solutions
- Robust regards any change of the environment (data, objectives, etc)
- Co-evolution, parallelism and distribution …
Disadvantages

- No guarantee for optimal solution within finite time (in general)
- May need parameter tuning
- Often computationally expensive, i.e. slow
Genetic Algorithms

- Developed: USA in the 1970’s
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators
# Genetic Algorithms (SGA)

<table>
<thead>
<tr>
<th>Representation</th>
<th>Binary strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination</td>
<td>N-point or uniform</td>
</tr>
<tr>
<td>Mutation</td>
<td>Bitwise bit-flipping with fixed probability</td>
</tr>
<tr>
<td>Parent selection</td>
<td>Fitness-Proportionate</td>
</tr>
<tr>
<td>Survivor selection</td>
<td>All children replace parents</td>
</tr>
<tr>
<td>Speciality</td>
<td>Emphasis on crossover</td>
</tr>
</tbody>
</table>
Evolution Strategies

- Developed: Germany in the 1970’s
- Early names: I. Rechenberg, H.-P. Schwefel
- Typically applied to:
  - numerical optimisation
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation
  - relatively much theory
- Special:
  - self-adaptation of (mutation) parameters standard
# Evolution Strategies

<table>
<thead>
<tr>
<th>Representation</th>
<th>Real-valued vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination</td>
<td>Discrete or intermediary</td>
</tr>
<tr>
<td>Mutation</td>
<td>Gaussian perturbation</td>
</tr>
<tr>
<td>Parent selection</td>
<td>Uniform random</td>
</tr>
<tr>
<td>Survivor selection</td>
<td>$(\mu,\lambda)$ or $(\mu+\lambda)$</td>
</tr>
<tr>
<td>Specialty</td>
<td>Self-adaptation of mutation step sizes</td>
</tr>
</tbody>
</table>
Evolutionary Programming

- Developed: USA in the 1960’s
- Early names: D. Fogel
- Typically applied to:
  - traditional EP: machine learning tasks by finite state machines
  - contemporary EP: (numerical) optimization
- Attributed features:
  - very open framework: any representation and mutation op’s OK
  - crossbred with ES (contemporary EP)
  - consequently: hard to say what “standard” EP is
- Special:
  - no recombination
  - self-adaptation of parameters standard (contemporary EP)
# Evolutionary Programming

<table>
<thead>
<tr>
<th>Representation</th>
<th>Real-valued vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination</td>
<td>None</td>
</tr>
<tr>
<td>Mutation</td>
<td>Gaussian perturbation</td>
</tr>
<tr>
<td>Parent selection</td>
<td>Deterministic</td>
</tr>
<tr>
<td>Survivor selection</td>
<td>Probabilistic ($\mu+\mu$)</td>
</tr>
<tr>
<td>Specialty</td>
<td>Self-adaptation of mutation step sizes (in meta-EP)</td>
</tr>
</tbody>
</table>
Genetic Programming

- Developed: USA in the 1990’s
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (prediction, classification...)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes: trees, graphs
  - mutation possible but not necessary (disputed!)
## Genetic Programming

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation</strong></td>
<td><strong>Tree structures</strong></td>
</tr>
<tr>
<td><strong>Recombination</strong></td>
<td><strong>Exchange of subtrees</strong></td>
</tr>
<tr>
<td><strong>Mutation</strong></td>
<td><strong>Random change in trees</strong></td>
</tr>
<tr>
<td><strong>Parent selection</strong></td>
<td><strong>Fitness proportional</strong></td>
</tr>
<tr>
<td><strong>Survivor selection</strong></td>
<td><strong>Generational replacement</strong></td>
</tr>
</tbody>
</table>
Books

Ant colony Optimization (ACO)
Ant colonies

- Artificial ants: Dorigo (1992)
- Imitate the cooperative behavior of ant colonies to solve optimization problems
- Use very simple communication mechanism: pheromone
  - Olfactive and volatile substance
  - Evolution: evaporation, reinforcement

A nature-inspired process

- During the trip, a pheromone is left on the ground.
- The quantity left depends on the amount of food found.
- The path is chosen accordingly to the quantity of pheromones.
- The pheromone has a decreasing action over time.
Ant colony procedure

- Do
  - Schedule activities
    - AntBasedSolution Construction
    - Pheromone Update
    - Daemons Actions
  - While termination conditions not met
Application to Combinatorial Optimization Problems

- **Traveling Salesman Problem**: T. Stützle, 1997
- **Vehicle Routing Problem**: Hartl et al., 1997
- **Graph Coloring**: Costa et Hertz, 1997
- **Frequency Assignment Problem**: Dicaro et al., 1997
- **Quadratic Assignment Problem**: Gambardella, Taillard et Dorigo, 1997
- **Set Covering**: E-G. Talbi et al., 1997
- **Graph Partitioning**: Kuntz et Snyers, 1994
Particle Swarm Optimization (PSO)
Particle Swarm

- Population based stochastic metaheuristic
- Dr. Eberhart and Dr. Kennedy (1995)
- Inspired by social behavior of bird flocking or fish schooling
- Similarities with genetic algorithm

A nature-inspired process

- Particles fly through the problem space
- Flight = add a velocity to the current position
- Social adaptation of knowledge
- Particles follow the current optimum particles («follow the bird which is nearest to the food »)
PSO procedure

- Do
  - Evaluate the velocities
  - Flight
  - Update the bests
- While termination conditions not met
Swarm construction

- Initialize positions $P$:
  \[ P_i = \text{random} \]

- Initialize the first best position $P_{\text{best}}$ of each particle:
  \[ P_{i\text{ best}} = P_i \]
  (standard strategy)

- Initialize the global best $P_{\text{gbest}}$ particle:
  \[ P_{\text{gbest}} = \text{best}(P_i) \]
  (standard strategy)
Make the particles flying

- Evaluate the velocities:
  \[ V_i = V_i + c_1 \times (P_{i\text{best}} - P_i) + c_2 \times (P_{gbest} - P_i) \]
  - local direction
  - global direction

  \( c_1 \) and \( c_2 = \) learning factors

- Perform the flight
  \[ P_i = P_i + V_i \]
**Topology**

- Determines how the solution spread through the population
- Local, global, neighbourhood best?
- Affects the rate of convergence
- Advanced parallel search

*Mean degree, Clustering, Heterogeneity*
Update the particle’s best

- Update the best fitness value of each particle:
  - *If* $P_i$ *better than* $P_i \text{best}$
    
    $$P_i \text{best} = P_i$$

- Update the global best:
  - *If* $P_i$ *better than* $P_{gbest}$
    
    $$P_{gbest} = P_i$$
Application to Combinatorial Optimization Problems

- Transport: Venter, 2004
- Planning: Onwubolu, G. C. and Clerc, 2004
Scatter search and path relinking
Scatter Search and Path Relinking

- Scatter Search and its generalized form Path Relinking provide unifying principles for joining (or recombining) solutions based on generalized path constructions in Euclidean or neighborhood spaces.


Main operators

- **Diversification Generation Method**: Generate a collection of diverse trial solutions
- **Improvement Method**: Transform a trial solution into one or more enhanced trial solutions
- **Reference Set Update Method**: Build and maintain a reference set - "best" solutions found (quality, diversity)
- **Subset Generation Method**: Operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions
- **Solution Combination Method**: Transform a given subset of solutions produced by the Subset Generation Method
Overview

- Diversification Generation Method
- Repeat until $|P| = P\text{Size}$
- Improvement Method
- Subset Generation Method
- Solution Combination Method
- Reference Set Update Method
- Stop if $\text{MaxIter}$ reached
- No more new solutions
- Improvement Method
- Diversification Generation Method
- Reference Set

$P$
Scatter Search procedure

- Create diversified population
- Generate Reference Set
- Do
  - Do
    - Subset Generation Method
    - Solution Combination Method
    - Improvement Method
  - Until first termination criterion is not reached
  - Reference Set Update Method
- Until other termination criterion is not reached
Estimation of Distribution Algorithm (EDA)
Estimation of Distribution Algorithm

- Based on the use of (unsupervised) density estimators/generative statistical models
- Idea is to convert the optimization problem into a search over probability distributions
- The probabilistic model is in some sense an explicit model of (currently) promising regions of the search space
EDA pseudo-code

- Initialize a probability model $Q(x)$
- Do
  - Create a population of points by sampling from $Q(x)$
  - Evaluate the objective function for each point
  - Update $Q(x)$ using selected population and $f()$ values
- While termination criterion not reached
EDA simplest probability model

- Population-based incremental Learning (PBIL)
  - Initial distribution $D=(0.5, \ldots, 0.5)$
  - Boucle:
    - Generation of the population
    - $X_i = 1$ if $r<D_i$ ($r$ uniform in $[0,1]$)
    - $X_i = 0$ else
    - Evaluate and sort the population
    - Update the distribution
      \begin{equation}
      D = (1 - \alpha)D + \alpha \cdot X_{\text{best}}
      \end{equation}

S. Baluja, R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. ICML'95
Other probability models

- Mutual Information Maximization for Input Clustering (MIMIC) regards pairwise dependances
  

- Bayesian Optimization Algorithm (BOA) for multivariate dependances
  
EDA: Applications

- Field of EDA is quite young. Much effort is focused on methodology rather than performance

- First applications
  - Knapsack problem
  - Job Shop Scheduling