Outline

1. Evolutionary Algorithms

2. Swarm Intelligence and Ant Colony Optimization
   Background
   Ant Colony Optimization: the Metaheuristic

Solution representation

- neat separation between solution encode or representation (genotype) from actual variables (phenotype)
- a solution \( s \in S \) is represented by a string that is: the genotype set is made of strings of length \( l \) whose elements are symbols from an alphabet \( A \) such that there exists a map:
  \[
  c : A^l \rightarrow S
  \]
  - the elements of strings are the genes
  - the values of elements can take are the alleles
- the search space is then \( X \subseteq A^l \)
- if the strings are member of a population they are called chromosomes and their recombination crossover
- strings are evaluated by \( f(c(x)) = g(x) \) which gives them a fitness
  \( \Rightarrow \) binary representation is appealing but not always good (e.g., in constrained problems binary crossovers might not be good)

Example

\[
\begin{array}{cccccccc}
1001010 & 1101000 & 0110100 & 1010010 & 1000100 \\
0101110 & 0111101 & 0101101 & 1100000 & 1010101 \\
01011 & 10101001001101 & 010100101000010 \\
01101 & 100111110101101 & 011010001010101
\end{array}
\]

Which Produces the Offspring

\[
\begin{array}{cccccccc}
010111101101110101101001010101 & 011010010100001010100101000010 \\
0101010011011101011010001010101 & 010100110101010101001010000010
\end{array}
\]

Conjectures on the goodness of EA

- schema: subset of \( A^l \) where strings have a set of variables fixed. Ex.: \( 1 \ast \ast 1 \)
  - exploit intrinsic parallelism of schemata
  - Schema Theorem:
    \[
    E[N(S, t + 1)] \geq \frac{F(S(t))}{F(S)} N(s, t)[1 - e(S, t)]
    \]
  - a method for solving all problems \( \Rightarrow \) disproved by the No Free Lunch Theorems
  - building block hypothesis

Selection

- Main idea: selection should be related to fitness
  - Fitness proportionate selection (Roulette-wheel method)
    \[
    p_i = \frac{f_i}{\sum f_j}
    \]
  - Tournament selection: a set of chromosomes is chosen and compared and the best chromosome chosen.
  - Rank based and selection pressure
Recombination (Crossover)
- Binary or assignment representations
  - one-point, two-point, m-point (preference to positional bias w.r.t. distributional bias)
  - uniform cross over (through a mask controlled by a Bernoulli parameter \( p \))
- Non-linear representations
  - (Permutations) Partially mapped crossover
  - (Permutations) mask based
- More commonly ad hoc crossovers are used as this appears to be a crucial feature of success
- Two off-springs are generally generated
- Crossover rate controls the application of the crossover. May be adaptive: high at the start and low when convergence

Mutation
- Goal: Introduce relatively small perturbations in candidate solutions in current population + offspring obtained from recombination.
- Typically, perturbations are applied stochastically and independently to each candidate solution; amount of perturbation is controlled by mutation rate.
- Mutation rate controls the application of bit-wise mutations. May be adaptive: low at the start and high when convergence
- Possible implementation through Poisson variable which determines the \( m \) genes which are likely to change allele.
- Can also use subsidiary selection function to determine subset of candidate solutions to which mutation is applied.
- The role of mutation (as compared to recombination) in high-performance evolutionary algorithms has been often underestimated

New Population
- Determines population for next cycle (generation) of the algorithm by selecting individual candidate solutions from current population + new candidate solutions obtained from recombination, mutation (+ subsidiary perturbative search). \( \lambda, \mu \) \( (\lambda + \mu) \)
- Goal: Obtain population of high-quality solutions while maintaining population diversity.
- Selection is based on evaluation function (fitness) of candidate solutions such that better candidate solutions have a higher chance of ‘surviving’ the selection process.
- It is often beneficial to use elitist selection strategies, which ensure that the best candidate solutions are always selected.
- Most commonly used: steady state in which only one new chromosome is generated at each iteration
- Diversity is checked and duplicates avoided

Example: A memetic algorithm for TSP
- Search space: set of Hamiltonian cycles
  Note: tours can be represented as permutations of vertex indexes.
- Initialization: by randomized greedy heuristic (partial tour of \( n/4 \) vertices constructed randomly before completing with greedy).
- Recombination: greedy recombination operator GX applied to \( n/2 \) pairs of tours chosen randomly:
  1) copy common edges (param. \( p_v \))
  2) add new short edges (param. \( p_e \))
  3) copy edges from parents ordered by increasing length (param. \( p_e \))
  4) complete using randomized greedy.
- Subsidiary perturbative search: LK variant.
- Mutation: apply double-bridge to tours chosen uniformly at random.
- Selection: Selects the \( \mu \) best tours from current population of \( \mu + \lambda \) tours (=simple elitist selection mechanism).
- Restart operator: whenever average bond distance in the population falls below 10.

Example: crossovers for binary representations

Subsidiary perturbative search
- Often useful and necessary for obtaining high-quality candidate solutions.
- Typically consists of selecting some or all individuals in the given population and applying an iterative improvement procedure to each element of this set independently.

Types of evolutionary algorithms
- Genetic Algorithms (GAs) [Holland, 1975; Goldberg, 1989]:
  - have been applied to a very broad range of (mostly discrete) combinatorial problems;
  - often encode candidate solutions as bit strings of fixed length, which is now known to be disadvantageous for combinatorial problems such as the TSP.
- Evolution Strategies [Rechenberg, 1973; Schwefel, 1981]:
  - originally developed for (continuous) numerical optimization problems;
  - operate on more natural representations of candidate solutions;
  - use self-adaptation of perturbation strength achieved by mutation;
  - typically use elitist deterministic selection.
- Evolutionary Programming [Fogel et al., 1966]:
  - similar to Evolution Strategies (developed independently),
  - but typically does not make use of recombination and uses stochastic selection based on tournament mechanisms.
  - often seek to adapt the program to the problem rather than the solutions.
The Biological Inspiration

Double-bridge experiment [Goss, Aron, Deneubourg, Pasteels, 1989]

- If the experiment is repeated a number of times, it is observed that each of the two bridges is used in about 50% of the cases.
- About 100% the ants select the shorter bridge.

Swarms and it happens through pheromone. If the experiment is repeated a number of times, it is observed that each of the two bridges is used in about 50% of the cases.

Theoretical studies

- Through Markov chains modelling some versions of evolutionary algorithms can be made to converge with probability 1 to the best possible solutions in the limit [Fogel, 1992; Rudolph, 1994].
- Convergence rates on mathematically tractable functions or with local approximations [Bäck and Hoffmeister, 2004; Beyer, 2001].
- "No Free Lunch Theorem" [Wolpert and Macready, 1997]. On average, within some assumptions, blind random search is as good at finding the minimum of all functions as hill climbing.

However,

- These theoretical findings are not very practical.
- EAs are made to produce useful solutions rather than perfect solutions.

Research Goals

- Analyzing classes of optimization problems and determining the best components for evolutionary algorithms.
- Applying evolutionary algorithms to problems that are dynamically changing.
- Gaining theoretical insights for the choice of components.

Swarm Intelligence

Definition: Swarm Intelligence

Swarm intelligence deals with systems composed of many individuals that coordinate using decentralized control and self-organization. In particular, it focuses on the collective behaviors that emerges from the local interactions of the individuals with each other and with their environment and without the presence of a coordinator.

Examples:

- Natural swarm intelligence
  - colonies of ants and termites
  - schools of fish
  - flocks of birds
  - herds of land animals

- Artificial swarm intelligence
  - artificial life (boids)
  - robotic systems
  - computer programs for tackling optimization and data analysis problems.

Self-organization

Four basic ingredients:

- Multiple interactions
- Randomness
- Positive feedback (reinforcement)
- Negative feedback (evaporating, forgetting)

Communication is necessary

- Two types of communication:
  - Direct: antennation, trophallaxis (food or liquid exchange), mandibular contact, visual contact, chemical contact, etc.
  - Indirect: two individuals interact indirectly when one of them modifies the environment and the other responds to the new environment at a later time. This is called stigmergy and it happens through pheromone.

Outline

1. Evolutionary Algorithms
2. Swarm Intelligence and Ant Colony Optimization
   Background
   Ant Colony Optimization: the Metaheuristic

The Biological Inspiration

Double-bridge experiment [Goss, Aron, Deneubourg, Pasteels, 1989]

- If the experiment is repeated a number of times, it is observed that each of the two bridges is used in about 50% of the cases.
- About 100% the ants select the shorter bridge.

No Free Lunch Theorem

NFL: No Free Lunch

All search algorithms are equivalent when composed over all possible discrete functions. Wolpert, Macready (1995)

Consider any algorithm $A$, applied to function $f_j$.

On($A_i, f_j$) outputs the order in which $A_i$ visits the elements in the codomain of $f_j$. Resampling is ignored. For every pair of algorithms $A_k$ and $A_i$, and for any function $f_j$, there exist a function $f$ such that

$$On(A_k, f_j) \neq On(A_i, f_j)$$

Consider a “BestFirst” versus a “WorstFirst” local search with restarts. For every $j$ there exists an $l$ such that

$$On(BestFirst, f_j) \neq On(WorstFirst, f_j)$$

Orthogonal to the natural / artificial distinction.
Mathematical Model

[Goss et al. (1989)] developed a model of the observed behavior:

Assuming that, at a given moment in time,

- $m_1$ ants have used the first bridge
- $m_2$ ants have used the second bridge,

The probability $p_i$ for an ant to choose the first bridge is:

$$p_i = \frac{|m_1 + k|^h}{|m_1 + k|^h + |m_2 + k|^h},$$

where parameters $k$ and $h$ are to be fitted to the experimental data.

From Real Ants to Artificial Ants

Our Design Choices

- Ants are given a memory of visited nodes
- Ants build solutions probabilistically without updating pheromone trails
- Ants deterministically backward retrace the forward path to update pheromone
- Ants deposit a quantity of pheromone function of the quality of the solution they generated

From Real Ants to Artificial Ants

Ants’ Probabilistic Transition Rule

$$p_{ijd}^k(t) = \frac{[\tau_{ijd}(t)]^\alpha}{\sum_{h \in H} [\tau_{ihd}(t)]^\alpha} \cdot \eta_{ijd}(t)$$

- $\tau_{ijd}$ is the amount of pheromone trail on edge $(i,j,d)$
- $H$ is the set of feasible nodes ant $k$ positioned on node $i$ can move to
- $\eta_{ijd}$ is a heuristic evaluation of link $(i,j,d)$ which introduces problem specific information

From Real Ants to Artificial Ants

Ants’ Pheromone Trail: Deposition and Evaporation

$$\tau_{ijd}(t + 1) \leftarrow (1 - \rho) \cdot \tau_{ijd}(t) + \Delta \tau_{ijd}(t)$$

where the $(i,j)$’s are the links visited by ant $k$, and

$$\Delta \tau_{ijd}(t) = \text{quality}^k \cdot \Delta \text{time it took ant } k \text{ to build the path from } i \text{ to } d \text{ via } j.$$

From Real Ants to Artificial Ants

Using Pheromone and Memory to Choose the Next Node

$$p_{ijd}^k(t) = f(\tau_{ijd}(t))$$

From Real Ants to Artificial Ants

Ants’ Probabilistic Transition Rule (Revised)

$$p_{ijd}^k(t) = \frac{[\tau_{ijd}(t)]^\alpha \cdot [\eta_{ijd}(t)]^\beta}{\sum_{h \in H} [\tau_{ihd}(t)]^\alpha \cdot [\eta_{ihd}(t)]^\beta}$$

- $\tau_{ijd}$ is the amount of pheromone trail on edge $(i,j,d)$
- $\eta_{ijd}$ is the heuristic evaluation of link $(i,j,d)$
- $H$ is the set of feasible nodes ant $k$ positioned on node $i$ can move to
The Simple Ant Colony Optimization Algorithm

- Ants are launched at regular instants from each node to randomly chosen destinations.
- Ants build their paths probabilistically with a probability function of: (i) artificial pheromone values, and (ii) heuristic values.
- Ants memorize visited nodes and costs incurred.
- Once reached their destination nodes, ants retrace their paths backwards, and update the pheromone trails.

The pheromone trail is the stigmergic variable.

Ant Colony Optimization

Example: A simple ACO algorithm for the TSP

- **Construction graph**
  - To each edge $ij$ in $G$ associate pheromone trails $\tau_{ij}$ and heuristic values $\eta_{ij} := \frac{1}{C}$.
  - Initialize pheromones.
  - **Constructive search:**
    \[ p_{ij} = \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in \Lambda_i} \tau_{il}^\alpha \cdot \eta_{il}^\beta} \]
  - Update pheromone trail levels
    \[ \tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \text{Reward} \]

Note:

- In each cycle, each ant creates one candidate solution using a constructive search procedure.
- Ants build solutions by performing randomized walks on a construction graph $G = (V, E)$ where $V$ are solution components and $G$ is fully connected.
- All pheromone trails are initialized to the same value, $\tau_0$.
- Pheromone update typically comprises uniform decrease of all trail levels (evaporation) and increase of some trail levels based on candidate solutions obtained from construction + perturbative search.
- Subsidiary perturbative search is (often) applied to individual candidate solutions.
- Termination criterion can include conditions on make-up of current population, e.g., variation in solution quality or distance between individual candidate solutions.

Why Does it Work?

Three important components:

- **TIME:** a shorter path receives pheromone quicker (this is often called: “differential length effect”)
- **QUALITY:** a shorter path receives more pheromone.
- **COMBINATORICS:** a shorter path receives pheromone more frequently because it is likely to have a lower number of decision points.

Artificial versus Real Ants:

Artificial ants:

- Live in a discrete world.
- Deposit pheromone in a problem dependent way.
- Can have extra capabilities: Local search, lookahead, backtracking.
- Exploit an internal state (memory).
- Deposit an amount of pheromone function of the solution quality.
- Can use local heuristic.

Ant Colony Optimization Metaheuristic

- Population-based method in which artificial ants iteratively construct candidate solutions.
- Solution construction is probabilistically biased by pheromone trail information, heuristic information and partial candidate solution of each ant.
- Pheromone trails are modified during the search process to reflect collective experience.

Example: A simple ACO algorithm for the TSP (1)

- Search space and solution set as usual (all Hamiltonian cycles in given graph $G$).
- Associate pheromone trails $\tau_{ij}$ with each edge $(i, j)$ in $G$.
- Use heuristic values $\eta_{ij} := \frac{1}{C}$ (better: $\eta_{ij} := \frac{C_{VN}}{C_{ij}}$).
- Initialize all weights to a small value $\tau_0$ ($\tau_0 = 1$).
- **Constructive search:** Each ant starts with randomly chosen vertex and iteratively extends partial round trip $n^k$ by selecting vertex not contained in $n^k$ with probability
  \[ p_{ij} = \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in \Lambda_i} \tau_{il}^\alpha \cdot \eta_{il}^\beta} \]
  where $\alpha$ and $\beta$ are parameters.
Example: A simple ACO algorithm for the TSP (2)

- Subsidiary perturbative search: Perform iterative improvement based on standard 2-exchange neighborhood on each candidate solution in population (until local minimum is reached).
- Update pheromone trail levels according to
  \[ \tau_{ij} := (1 - \rho) \cdot \tau_{ij} + \sum_{s' \in \text{sp}'} \Delta_{ij}(s') \]
  where \( \Delta_{ij}(s') := \frac{1}{g(s')} \) (better \( \Delta_{ij}(s') = C_{NN}m \cdot g(s') \)) if edge \((i, j)\) is contained in the cycle represented by \(s'\), and 0 otherwise.

Motivation: Edges belonging to highest-quality candidate solutions and/or that have been used by many ants should be preferably used in subsequent constructions.

- Termination: After fixed number of cycles (= construction + perturbative search phases).

ACO Variants

- Ant System (AS) [Dorigo et al., 1991]
- Elitist AS [Dorigo et al., 1991; 1996]
  - The iteration best solution adds more pheromone
- Rank-Based AS [Bullnheimer et al., 1997]
  - Only best ranked ants can add pheromone
  - Pheromone added is proportional to rank
- Max-Min AS [Stützle & Hoos, 1997]
  - Approximate Nondeterministic Tree Search ANTS [Maniezzo, 1999]
  - Hypercube AS [Blum, Roli and Dorigo, 2001]

ACO: Theoretical results

- Through Markov chains modelling some versions of ACO can be made to converge with probability 1 to the best possible solutions in the limit [Gutjahr, 2000; Stützle and Dorigo, 2002]

Outline

3. Problems
- Set Covering

Set Covering Problem

Input: a finite set \(X\) and a family \(\mathcal{F}\) of subsets of \(X\) such that every element of \(X\) belongs to at least one subset in \(\mathcal{F}\):

\( X = \bigcup_{S \in \mathcal{F}} S \)

Task: Find a minimum cost subset \(C\) of \(\mathcal{F}\) whose members cover all \(X\):

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} c_j x_j \\
\text{such that} & \quad X = \bigcup_{S \in C} S \quad (1)
\end{align*}
\]

Any \(C\) satisfying (1) is said to cover \(X\)

Covering, Partitioning, Packing

Set Covering

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} c_j x_j \\
\text{such that} & \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \\
& \quad x_j \in \{0, 1\}
\end{align*}
\]

Set Partitioning

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} c_j x_j \\
\text{such that} & \quad \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall i \\
& \quad x_j \in \{0, 1\}
\end{align*}
\]

Set Packing

\[
\begin{align*}
\max & \quad \sum_{j=1}^{n} c_j x_j \\
\text{such that} & \quad \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall i \\
& \quad x_j \in \{0, 1\}
\end{align*}
\]