Swarm Intelligence: An Introduction

Nathan Bell

Introduction to Swarm Intelligence

This presentation will cover the following topics:

- What is Swarm Intelligence?
- Origins of Swarm Intelligence
- Core Concepts
- Applications and SI based algorithms

What is Swarm Intelligence?

What is “Swarm Intelligence” (SI)?

- `http://www.youtube.com/watch?v=jEGV4ZSP22A`
- “The collective behaviour of a decentralized, self-organized system”
- What is “Decentralization”?
- What is “Self-Organization”?

What is Swarm Intelligence?

What does this definition mean to us?

- System consisting of a population of “agents”
- Simple Interactions between agents
- Leading to complex high-level behaviour

Why is Swarm Intelligence Interesting?

Why is SI interesting to us?

- Framework for decentralized and scalable problem solving
- Unique perspective for addressing many problems

Why is Swarm Intelligence Interesting?

SI naturally lends itself to alternative computational models:

- Distributed models
- Massively parallel models
- Flexible, dynamic models
- Robotics
- etc...

Why is Swarm Intelligence Interesting?

Useful in simulating many real-life systems

- Behaviour of crowds
- Traffic simulation
- Spread of disease
- etc...

Origins, Observations of Nature

Where did the idea of SI originate?

- Inspired by the success of swarming creatures in nature
- In particular: ants, termites and bees

Ant Colonies

What's so interesting about ants?

- One of the most successful species on the planet
- Colonies can range from tens, to millions, of ants
- Individual ants are extremely basic creatures
- Colony displays a complex structure and behaviour
- How do they achieve this success?

Colony Behaviour

A basic look at ant colony behaviour:

- Coordinated among specialized workers
- Labour tasks include:
  - Defence
  - Food Collection
  - Brood Care
  - Nest Cleaning
  - Nest Construction

Colony Behaviour

Ants achieve complex behaviours:

- Division of labour, adaptive task allocation
- Path finding and optimization
- Clustering and sorting
- Structure formation
- Recruitment for foraging and collective transport of food

Colony Behaviour

How does the colony address these tasks?

- Ants, as individuals, are extremely basic
- No single ant could plan these complex tasks
- These tasks are easily completed by the ant colony
- How is this possible?

Colony Behaviour

Colony behaviour:

- Labour within ant colonies is decentralized
- No central planning or control
- Local interactions between ants
- Laying of chemical pheromones
- Complex behaviour arises
- A swarm intelligence is displayed

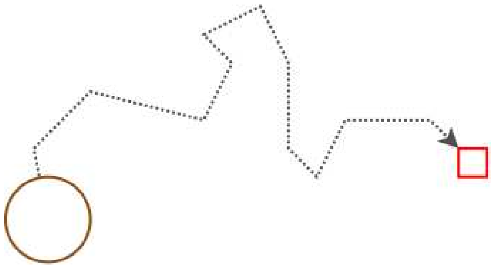
Ant Foraging

Ant Foraging:

- Ants effectively locate and exploit food sources
- Food sources are discovered by random exploration
- Ants discover efficient paths to known food sources
- Ants have no high-level knowledge of the situation
- How do ants form these efficient paths?

Ant Foraging

- The key is pheromone
- While searching for food, ant movement is largely random

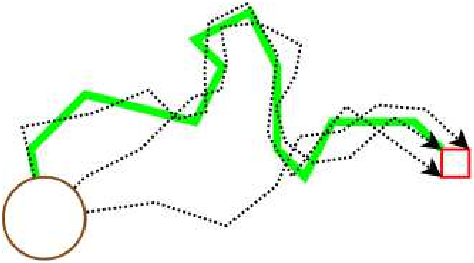


Ant Foraging

- Ants will return to the colony with food
- Ants with food will lay a trail of pheromone along their path

Ant Foraging

- Ants are sensitive to pheromone
- The pheromone deposited attracts other ants
- Ants encountering a pheromone trail will tend to follow it
- These ants are more likely to reach the food source



Ant Foraging

Over Time:

- More ants discover the food source
- More pheromone is deposited
- Existing trails to the food source are strengthened

Trail Optimization

At the high level:

- Ants will seem to choose more optimal paths
- How is this achieved?
- Combination of:
    - Randomness of movement
    - Nature of pheromone

Trail Optimization

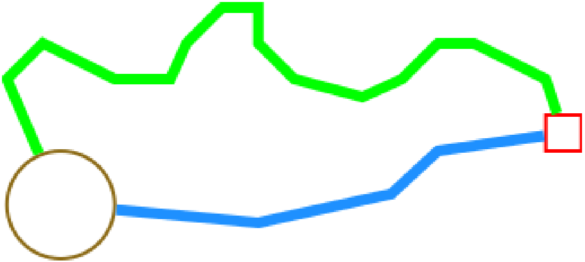
Properties of pheromone:

- Pheromones are chemicals subject to:
  - Evaporation
  - Diffusion
  - Other environmental effects
- Pheromone weakens over time
- Pheromone trails will dissipate and spread out

Trail Optimization

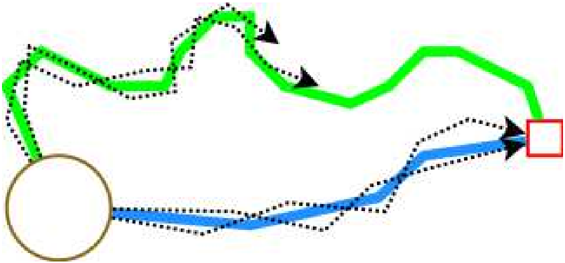
Example:

- A food source has been discovered by two ants
- One ant happens to take a more efficient path



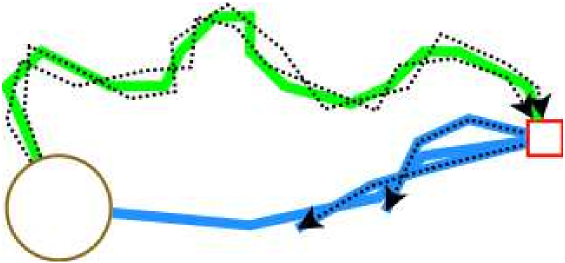
Trail Optimization

- Each trail initially attracts a roughly equal number of ants
- Ants on the more efficient path arrive earlier



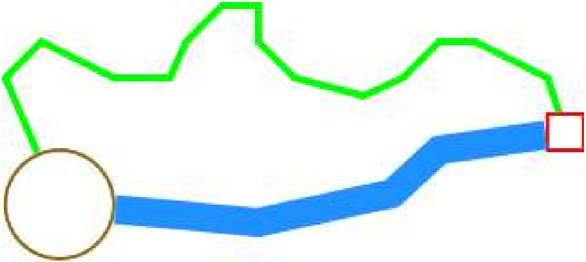
Trail Optimization

- Ants return, depositing additional pheromone
- Pheromone trail is strengthened on return



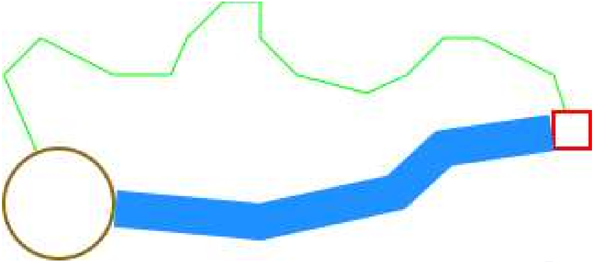
Trail Optimization

- Ants following the more efficient trail have shorter trips
- Trips are completed with a higher frequency
- Higher frequency leads to stronger pheromone



Trail Optimization

- Ants will favour the stronger pheromone trail
- Less efficient trail slowly dissipates
- The colony has chosen the more efficient path
- No knowledge of the global situation!



Core Concepts and Principles

What differentiates Swarm Intelligence from other Population-based methods?

- We must understand the core components:
  - Decentralization
  - Self-Organization
  - Emergent Behaviour

Decentralization and Self-Organization

Decentralization:

- Population of roughly homogeneous agents
- Control is fully distributed among the population
- No central “brain” controlling the agents
- Each agent has roughly the same level of influence

Decentralization and Self-Organization

Self-Organization:

- Agents each act according to their own behaviour
- Agents communicate and interact
- Communications can include:
  - Direct contact
  - Local exchange of information
  - Local broadcasts
  - Stigmergy
  - etc...
- Global behaviour arises from the interactions of the agents

Emergent Behaviour

“Emergent Behaviour” is the heart of SI

- A High-level behaviour of a population based system
- Must arise from the local interactions within the population
- Self-organization of a decentralized population
- Two categories: Weak and Strong

Emergent Behaviour

Weak emergent behaviour of a population:

- *can* be reduced to the behaviour of a single individual

Strong emergent behaviour of a population:

- can *not* be reduced to the behaviour of a single individual

Emergent Behaviour

Weak Emergent Behaviour:

- Extremely common
- Can be easily predicted by looking at a single individual
- Often simple to engineer
- For example, if individuals simply “move forward”, the population will, as a whole, move forward

Emergent Behaviour

Strong Emergent Behaviour:

- Heart of SI and very interesting phenomena itself
- Hard to predict from the behaviour of an individual
- May seem to “transcend” the capabilities of the individual
- “The whole is greater than the sum of its parts”
- It is often quite difficult to engineer
- For example, path optimization of foraging ants

SI and Artificial Life

SI and Artificial Life:

- SI was first explored in the field of artificial life
- Simulations of swarming creatures
- “Boids” algorithm
- Motivation: learning more about emergent behaviours

SI based Algorithms

SI for problem solving:

- Relatively new field
- Active field of research
- SI algorithms have been used to address many problems
- Most commonly, optimization problems

Ant Colony Optimization

Ant Colony Optimization (ACO):

- Meta-heuristic method for combinatorial optimization
- One of the first SI algorithms for optimization
- Modelled after the foraging behaviour of ants
- ACO finds good paths through graphs
- Applicable to a wide variety of problems

How ACO works

How does ACO work?

- ACO simulates the way ants communicate via pheromone
- Iteratively generates paths through graphs
- Pheromone “deposited” on graph edges
- Pheromone levels effect edge choices
- Good paths will emerge over time

How ACO works

Generic ACO Process:

- Generic ACO consists of a two phase loop
  - Edge selection phase
  - Pheromone update phase
- Loop iterates until reaching some termination criteria

Generic Ant Colony Optimization

Generic edge selection phase:

- Population of ants placed into the graph
- Ants move randomly through the graph
- Movement influenced by edge weights and pheromone
- Phase ends when all ants have created some kind of path

Generic Ant Colony Optimization

Generic ant movement:

- For current node x with set of neighbours Y
- Choose edge $xy, y \in Y$ with probability p_{xy}

$$p_{xy} = \frac{(\tau_{xy})(\eta_{xy})}{\sum_{y_i \in Y}(\tau_{xy_i})(\eta_{xy_i})}$$

- η_{xy} - Contribution of edge weight
- τ_{xy} - Contribution of pheromone

Generic Ant Colony Optimization

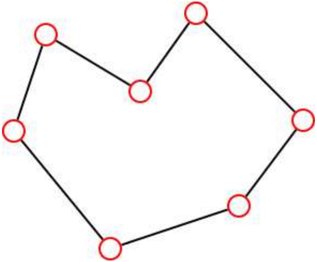
Generic pheromone update phase:

- Existing pheromone levels are reduced via "evaporation"
  - $\tau_{xy} = (1 - \rho)\tau_{xy}$
  - ρ - Evaporation coefficient parameter
- Ants return along the path taken through the graph
- At each edge, ants deposit pheromone
- Pheromone deposited according to: $\Delta\tau^k$
  - Path of each ant is evaluated using heuristic
  - $\Delta\tau^k \propto$ heuristic path value of ant k

Ant Colony Optimization for Travelling Salesman

Travelling Salesman Problem (TSP):

- Given a set of cities
- Visit each city exactly once
- Return to the origin
- Find the tour with shortest total length



Ant Colony Optimization for Travelling Salesman

Travelling Salesman Problem (TSP):

- NP-Hard
- Modelled as a complete graph
- Each node represents a city
- Edges have weight equal to the distance between cities
- ACO is applied to find *good* solutions in polynomial time

Ant Colony Optimization for Travelling Salesman

ACO for TSP:

- Convert edge distance values into attractiveness value η
- For each pair of cities x, y:
  - $\eta_{xy} = P / d_{xy}$
  - P - Initial attractiveness parameter
  - d_{xy} - Distance between cities x and y
- Each ant will form a valid tour during edge selection
- Pheromone deposited according to tour values

Ant Colony Optimization for Travelling Salesman

Edge selection for TSP:

- Each ant randomly assigned some city as their origin
- Ants travel through the graph visiting all cities
- Ants blacklist cities they have previously visited
- Blacklisted cities do not contribute to p_{xy} values
- After visiting all cities, ants return to their origin

Ant Colony Optimization for Travelling Salesman

Selecting the next city in the tour:

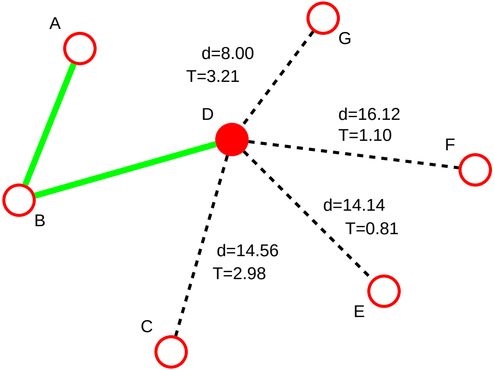
- Current city x
- Set of *unvisited* cities Y
- Move to city $y \in Y$ with probability p_{xy}

$$p_{xy} = \frac{(\tau_{xy})(n_{xy})}{\sum_{y_i \in Y}(\tau_{xy_i})(n_{xy_i})}$$

- If Y is empty, move to origin then stop

Ant Colony Optimization for Travelling Salesman

For example, consider the following situation:



Ant Colony Optimization for Travelling Salesman

X	d	η	τ	p
A	− - −	− - −	− - −	− - −
B	− - −	− - −	− - −	− - −
C	14.56		2.98	
D	− - −	− - −	− - −	− - −
E	14.14		0.81	
F	16.12		1.10	
G	8.00		3.21	

Calculate η for each move:

- $\eta_{xy} = P/d_{xy}$
- P = 100 for example

Ant Colony Optimization for Travelling Salesman

X	d	η	τ	p
A	– - –	– - –	– - –	– - –
B	– - –	– - –	– - –	– - –
C	14.56	6.87	2.98	
D	– - –	– - –	– - –	– - –
E	14.14	7.07	0.81	
F	16.12	6.20	1.10	
G	8.00	12.5	3.21	

Calculate p for each move:

- $p_{xy} = \frac{(\tau_{xy})(\eta_{xy})}{\sum_{y_i \in Y}(\tau_{xy_i})(\eta_{xy_i})}$

Ant Colony Optimization for Travelling Salesman

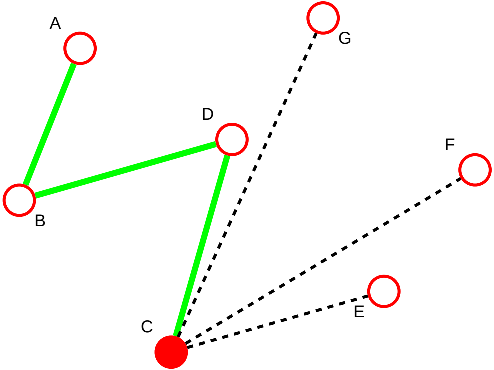
X	d	η	τ	p
A	– - –	– - –	– - –	0
B	– - –	– - –	– - –	0
C	14.56	6.87	2.98	0.28
D	– - –	– - –	– - –	0
E	14.14	7.07	0.81	0.08
F	16.12	6.20	1.10	0.09
G	8.00	12.5	3.21	0.55

Choose the next move randomly according to p:

- C is chosen

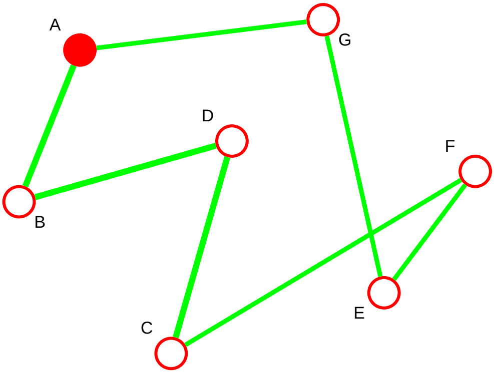
Ant Colony Optimization for Travelling Salesman

Move to C and repeat:



Ant Colony Optimization for Travelling Salesman

When all cities have been visited, a valid tour has been created



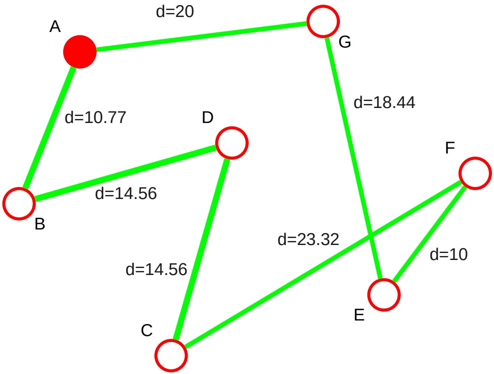
Ant Colony Optimization for Travelling Salesman

Pheromone Update Phase:

- Each ant calculates the cost of their tour
- Ants travel along their tours depositing pheromone
- For each ant k:
    - $\Delta\tau^k = Q/L_k$
        - L_k - Value of ant k's tour
        - Q - Pheromone attractiveness parameter
- For each edge xy:
    - $\tau_{xy} = (1 - \rho)\tau_{xy} + \sum_k \Delta\tau^k$

Ant Colony Optimization for Travelling Salesman

Consider this tour:



Ant Colony Optimization for Travelling Salesman

Pheromone deposit is calculated:

$$\begin{aligned} L &= d_{AB} + d_{BD} + d_{DC} + d_{CF} + d_{FE} + d_{EG} + d_{GA} \\ &= 10.77 + 14.56 + 14.56 + 23.32 + 10 + 18.44 + 20 \\ &= 111.65 \end{aligned}$$

$$Q = 100 \text{ for example}$$

$$\begin{aligned} \Delta\tau &= Q/L \\ &= 100/111.65 \\ &= 0.89 \end{aligned}$$

Ant Colony Optimization for Travelling Salesman

Algorithm:

- For each edge xy:
  - Initialize $\eta_{xy} = P/d_{xy}$
  - Initialize $\tau_{xy} = 0$
- Run simulation loop until reaching termination criteria
  - Edge Selection
  - Pheromone Update
- Return best tour found as output

Ant Colony Optimization for Travelling Salesman

Behaviour in early iterations:

- Little to no pheromone
- Ants generate tours in a greedy way
- Closest cities chosen with high probability
- High variety in TSP tours generated

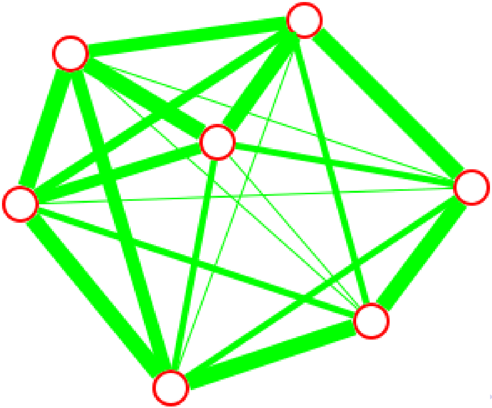
Ant Colony Optimization for Travelling Salesman

Behaviour in later iterations:

- Pheromone levels build up on popular edges
- Edges used by “good” tours will have more pheromone
- Pheromone levels begin to have greater influence
- Ants generate similar tours using these “good” edges

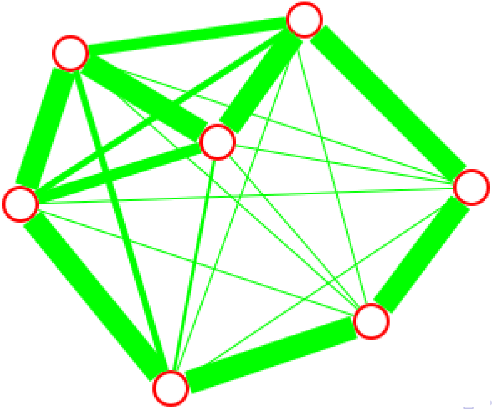
Ant Colony Optimization for Travelling Salesman

Early iterations:



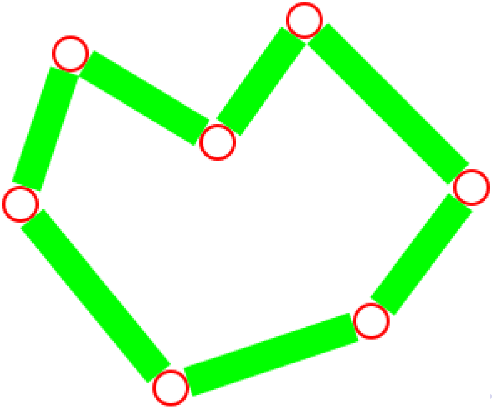
Ant Colony Optimization for Travelling Salesman

Later iterations:



Ant Colony Optimization for Travelling Salesman

Convergence:



ACO for Classification

- ACO finds applications in a wide variety of fields
- For example: data mining
- Rule-based classification

Rule Based Classification

Given an object with a set of attributes:

- Assign a predefined class to the object
  - Based on a set of rules
  - Find a rule matching the attributes
  - Assign a class using this rule

Rule Based Classification

Classification Rule:

- A rule assigns a class with the form:
  - IF < conditions > THEN < class >
    - IF A and B THEN 0
    - IF (A and C) or D THEN 1
    - etc...
- A condition has the form:
  - Attribute, Operator, Value
    - Colour = Blue
    - Width < 2
    - etc...

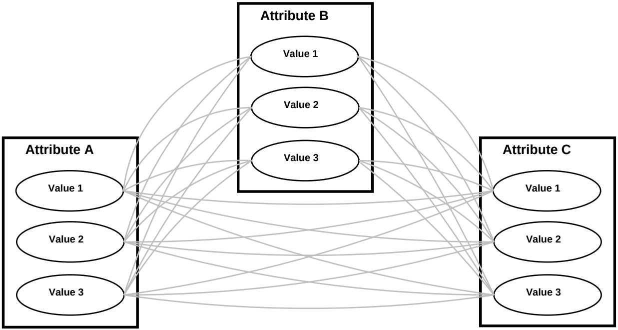
ACO for Rule Based Classification

ACO can discover rules from a data set:

- These rules are of the form:
  - IF < condition AND condition AND ... > THEN < class >
- These conditions are restricted to “categorical” attributes
  - Attribute = Value

Rules as Paths

Rules can be interpreted as paths through a graph:

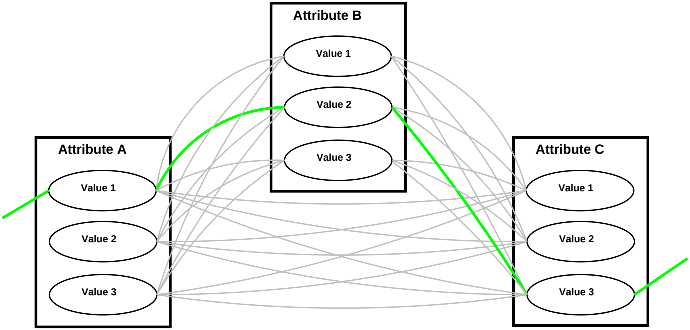


Rules as Paths

- Vertices represent conditions:
  - Attribute-value pair
  - “Attribute = Value”
- Edges represent “AND” relations between conditions

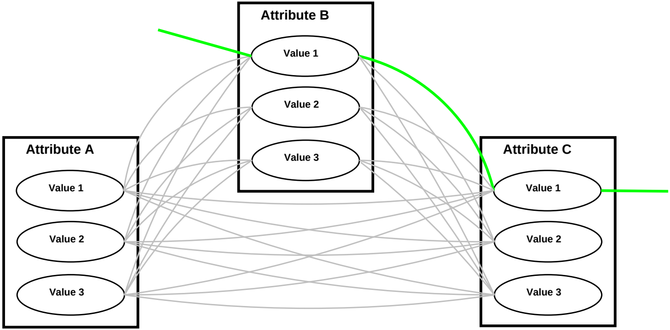
Rules as Paths

IF A=1 AND B=2 AND C=3:



Rules as Paths

IF B=1 AND C=1:



Applying ACO

To apply ACO to this graph representation, we require:

- Initial edge weights
- Heuristic for depositing pheromone

Initial Edge Weights

Edge weights are assigned using the training set:

- Edges leading to an Attribute-Value pair are assigned weightings based on that Attribute-Value pair
- Weights are determined according to the normalized Shannon Entropy of that Attribute-Value pair within the training set

Initial Edge Weights

Shannon Entropy:

$$H(W|A_a = V_{av}) = -\sum_{w \in W} (P(w|A_a = V_{av}) * \log_2 P(w|A_a = V_{av}))$$

- W: Set of possible classes
- A_a: Attribute $a \in A$
- V_{av}: Value $v \in V_a$

Represents the information gained by observing a given Attribute-Value pair according to the training set

Edge Weights

For edges leading to value $v \in V_a$ of attribute $a \in A$:

$$\eta = \frac{\log_2 |W| - H(W|A_a = V_{av})}{\sum_{i \in A} \sum_{j \in V_i} \left(\log_2 |W| - H(W|A_i = V_{ij})\right)}$$

A normalized and inverted Shannon Entropy

Pheromone Heuristic

New rules are evaluated using the “quality” measure:

$$Q = \frac{TP}{TP + FN} * \frac{TN}{FR + TN}$$

- TP : True Positives
- TN : True Negatives
- FP : False Positives
- FN : False Negatives

ACO Rule Generation Algorithm

Until termination:

- Initialize edge weights using the uncovered training set
- Until convergence:
  - Generate a new rule as a path
  - Deposit and evaporate pheromone
- Add the best-found rule to the final set of rules
- Discard training cases covered by the new rule

Output: A list of classification rules

Convergence and Termination

Convergence occurs when either:

- The same rule is generated a specified number of times
- The maximum number of ants per iteration is reached

Termination occurs when:

- The number of uncovered cases reaches a threshold

Result

ACO has successfully been applied to generate a set of classification rules from a training set.
These rules can now be applied to perform classification.

Particle Swarm Optimization

Particle Swarm Optimization (PSO):

- SI algorithm for real-valued, black-box optimization
- Discovered accidentally
- Originally a simulation of “social flocking” behaviour
- Population made of “particles” in the solution space
- Collision free “bird flocking” behaviour

Real-valued Black-box Optimization

Real-valued optimization problems:

- Find the optimal input to some objective function
- Each parameter is a real number

Solution space:

- Parameters define a “solution space”
- Each parameter corresponds to a dimension
- A point in this space represents a function input

Black-box optimization problems:

- Objective function provided as a “black-box”
- Nothing is known about the function
- Must learn about the function through evaluations

How Particle Swarm Optimization Works

Particles:

- Particles exist as points in the solution space
- These points represent input values to objective function
- Assigned values by evaluating the objective function
- “Fly” through solution space
- Communicate via broadcast

How Particle Swarm Optimization Works

Each particle is simply made up of:

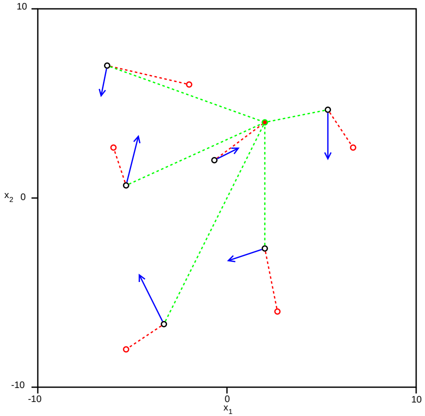
- Position $\vec{X}$
- Velocity $\vec{V}$
- Personal Best Point $\vec{P_i}$

How Particle Swarm Optimization Works

Particle behaviour:

- Particles move according to their velocity
- Velocity is updated through accelerations
- Inertial weighting is applied to prevent explosive speeds
- Acceleration towards the particle's best found point
- Acceleration towards the swarm's best found point

How Particle Swarm Optimization Works



How Particle Swarm Optimization Works

Acceleration towards personal best:

- Self influence
- Influence of the particle's own knowledge

Acceleration towards global best:

- Social influence
- Influence of the swarm's collective knowledge
- Global best points are communicated via broadcast

How Particle Swarm Optimization Works

Accelerations:

- Randomly weighted according to parameters
  - ϕ_p - Personal influence parameter
  - ϕ_g - Social influence parameter
- Separate random weighting per dimension
  - More "explorative"
- Constant deceleration is applied
  - ω - Inertial weighting parameter
  - Allows stronger accelerations without explosive speeds
  - Promotes convergence

Effects of Parameters

Low personal influence, high social influence:

- Faster convergence
- More susceptible to local optima
- Exploitation > Exploration

Effects of Parameters

High personal influence, low social influence:

- Slower convergence
- Less susceptible to local optima
- Exploitation < Exploration

Effects of Parameters

NO personal influence, high social influence:

- “Social only” swarm
- All particles immediately converge to the best found point
- Behaviour degrades to simple repeated local search

Effects of Parameters

High personal influence, *NO* social influence:

- “Self only” swarm
- Each particle acts completely independently
- Behaviour degrades to multiple local search
- No “swarm intelligence” present

Effects of Parameters

Parameter values:

- Matter of exploration vs. exploitation
- Exploration required for more “difficult” problems
- Exploitation required for convergence and efficiency

Effects of Parameters

Parameters:

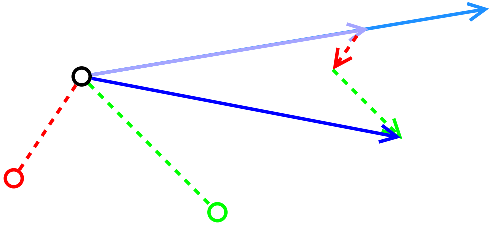
- Typically, both ϕ_p and ϕ_g are set to 2
- ω set to around 0.7
- Must be tweaked for each problem to get best results
- This tweaking is often unintuitive
- Requires knowledge of the solution space

Particle Swarm Optimization Algorithm

Velocity and position update equations:

- For each dimension d:

$$V_d = \omega V_d + U[0, \phi_p](P_{i_d} - X_d) + U[0, \phi_g](P_{g_d} - X_d)$$
$$X_d = X_d + V_d$$



Example Particle Update

Consider a particle with:

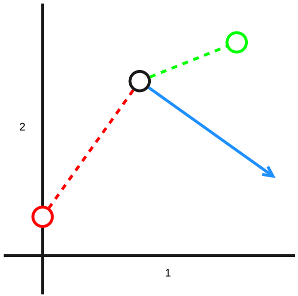
- $X = [5, 9]$
- $V = [7, -5]$
- $P = [0, 2]$

And global best found point:

- $P_g = [10, 11]$

With PSO Parameters:

- $\omega = 0.7$
- $\phi_p = 2, \phi_g = 2$

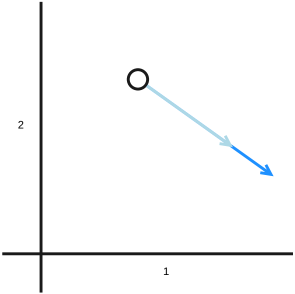


Example Particle Update

Inertial weighting is applied:

$$\begin{aligned} V_1 &= \omega V_1 \\ &= 0.7 * 7 \\ &= 4.9 \end{aligned}$$

$$\begin{aligned} V_2 &= \omega V_2 \\ &= 0.7 * -5 \\ &= -3.5 \end{aligned}$$

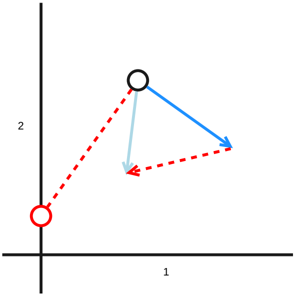


Example Particle Update

Random acceleration towards P:

$$\begin{aligned} V_1 &= V_1 + U[0, \phi_p](P_1 - X_1) \\ &= 4.9 + U[0, 2](0 - 5) \\ &= 4.9 + (1.1 * -5) \\ &= -0.6 \end{aligned}$$

$$\begin{aligned} V_2 &= V_2 + U[0, \phi_p](P_2 - X_2) \\ &= -3.5 + U[0, 2](2 - 9) \\ &= -3.5 + (0.2 * -7) \\ &= -4.9 \end{aligned}$$

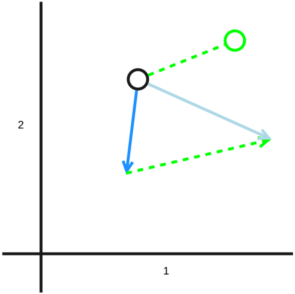


Example Particle Update

Random acceleration towards P_g:

$$\begin{aligned} V_1 &= V_1 + U[0, \phi_g](P_1 - X_1) \\ &= -0.6 + U[0, 2](10 - 5) \\ &= 4.9 + (0.4 * 5) \\ &= 6.9 \end{aligned}$$

$$\begin{aligned} V_2 &= V_2 + U[0, \phi_g](P_2 - X_2) \\ &= -4.9 + U[0, 2](11 - 9) \\ &= -4.9 + (0.9 * 2) \\ &= -3.1 \end{aligned}$$

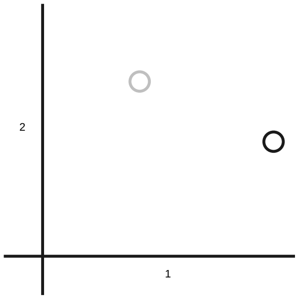


Example Particle Update

Update position:

$$
\begin{aligned}
X_1 &= X_1 + V_1 \\
&= 5 + 6.9 \\
&= 11.9
\end{aligned}
$$

$$
\begin{aligned}
X_2 &= X_2 + V_2 \\
&= 9 - 3.1 \\
&= 5.9
\end{aligned}
$$



Particle Swarm Optimization Algorithm

Algorithm:

- Initialize particles:
  - Random position
  - Random velocity
  - Evaluate initial positions
- While termination criteria not met:
  - Update P_g via communication
  - For each particle i:
    - Update velocity
    - Update position
    - Evaluate new position
    - Update P_i

Swarm Behaviour

Behaviour in early stages:

- Particles will fly through the solution space
- Overshoot the global and personal best points
- Arc and loop around these points
- Coarse-grained search
- Swarm finds “good” areas
- Particles slow over time, swarm begins to converge

Swarm Behaviour

Behaviour in later stages:

- Stagnation of global best point leads to convergence
- Particles gather around this point, slowing over time
- Fine-grained search of the “good” area
- Swarm finds the best point within the “good” area

Benefits of Particle Swarm Optimization

Benefits of PSO:

- Very simple to implement
- No costly computations
- Easily extendable to problems of any dimension
- Effective on difficult, noisy problems
- Can be applied to any black-box real-valued function
- Can even be used in meta-optimization
- PSO finding optimal parameters for PSO on some problem

Example Application

PSO can be applied to solve a problem if:

- A solution to the problem can be represented as a number of real-valued parameters
- A solution's quality can be represented by a single value
- A function is provided which evaluates any given solution

As an example, consider the problem of maximizing the coverage of a number of broadcast towers.

Simple Broadcast Coverage Problem

Given a set number of broadcast towers:

- Assign each tower an x, y coordinate
- Assign each an amount of power

In order to:

- *Maximize* coverage
- *Minimize* power costs

Simple Broadcast Coverage Problem

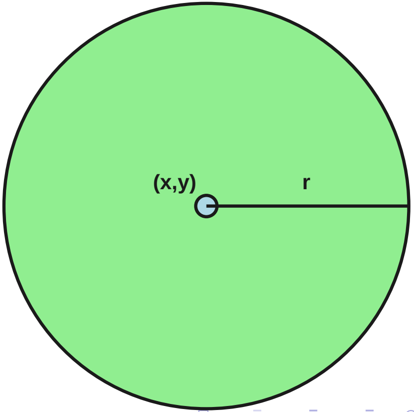
In order to apply PSO we must:

- Form a real-valued solution space
- Represent the problem as a function which:
  - Assigns values to points in the solution space
  - Evaluates the quality of a given solution point

Tower Representation

Each tower has 3 parameters

- x coordinate
- y coordinate
- power r



Solution Space

Multiple towers can be respresented:

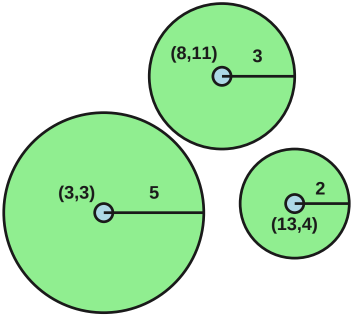
- $[Tower_1, Tower_2, ..., Tower_n]$
- $[x_1, y_1, r_1, x_2, y_2, r_2, ..., x_n, y_n, r_n]$
- A solution space with $3n$ dimensions

A particle's position in this space describes:

- The positions and power values of n towers
- A candidate solution to the tower coverage problem

Example Point

For example, $X = [3, 3, 5, 8, 11, 3, 13, 4, 2]$ corresponds to:



Evaluating a Solution

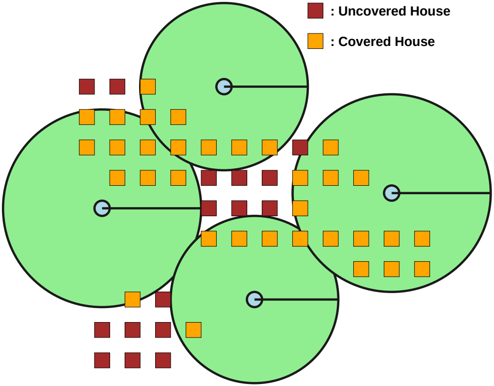
PSO requires a function to evaluate these solutions:

- Total coverage of the towers
- Penalize for power costs

Coverage

For simplicity:

- Houses within broadcast range
- Each house is worth some set value
- For example:
  - 10 per house



Power Cost

For simplicity, the cost of a tower is:

- A flat initial cost, for example 100
  - If $r <= 0$, the tower is considered unused
  - No initial cost in this case
- Power cost scaling exponentially with r
  - For example: r^2

Cost of n towers $= \sum_{i=0}^{n} \begin{cases} 0, & r_i <= 0 \\ 100 + r_i^2, & r_i > 0 \end{cases}$

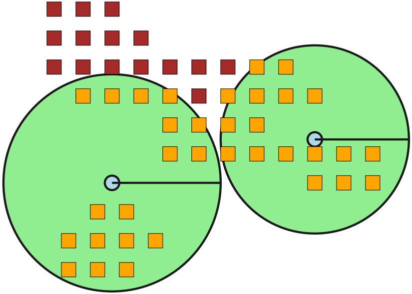
Final Evaluation Function

The final evaluation function is then:

$$\begin{aligned} f(...) &= HouseCoverage - PowerCosts \\ &= 10 * |Covered| - \sum_{i=0}^{n} \begin{cases} 0, & r_i <= 0 \\ 100 + r_i^2, & r_i > 0 \end{cases} \end{aligned}$$

Example

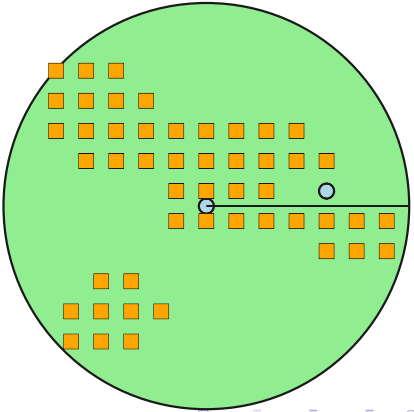
$$
\begin{aligned}
f(7,7,7,21,10,6) \\
&= 10 * |Covered| \\
&\quad - \sum_{i=0}^{n} \begin{cases} 0, & r_i <= 0 \\ 100 + r_i^2, & r_i > 0 \end{cases} \\
&= 10 * 34 \\
&\quad - \left(7^2 + 100\right) \\
&\quad - \left(6^2 + 100\right) \\
&= 55
\end{aligned}
$$



Example

$$f(13, 10, 13, 21, 11, 0)$$

$$
\begin{aligned}
=&10 * |Covered| \\
&- \sum_{i=0}^{n} \begin{cases} 0, & r_i <= 0 \\ 100 + r_i^2, & r_i > 0 \end{cases} \\
=&10 * 49 \\
&- \left(13^2 + 100\right) \\
&- (0) \\
=&121
\end{aligned}
$$



Applying PSO

Applying PSO:

- Plug in the evaluation function
- Provide appropriate parameter ranges

PSO will search for the optimal solution:

- According to the provided evaluation function
  - Accuracy of this function is essential
  - PSO will exploit errors in this function

Thank You

Questions?

New Work

My research:

- Particle Field Optimization (PFO)
- Based on the PSO algorithm
- Abstraction of the “Bare-Bones” PSO model

Exploring:

- New high-level concept
- New behaviour
- New avenues for development

Bare-Bones PSO

Bare-Bones Particle Swarm (BBPSO):

- Abstraction of the core PSO
- Simplifies the particle update process
- Removal of velocity and acceleration
- Retains roughly the same behaviour

Bare-Bones PSO

From PSO to BBPSO:

- Observations of a single particle during swarm stagnation
- Each dimension shows a distinct bell-curve histogram
- Can a similar histogram be achieved in a simpler way?

Bare-Bones PSO

New method of position update:

- Update particles according to Gaussian distribution
- Distribution constructed to mimic histogram bell-curve
- Velocity and accelerations discarded entirely
- Particle movement abstracted to a random sampling

Bare-Bones PSO

BBPSO particle update:

- For each particle i:
  - $P_m = \frac{P_i + P_g}{2}$
  - For each dimension d:
    - $X_{i_d} = \mathcal{N}(P_{m_d}, |P_{i_d} - P_{g_d}|)$

Bare-Bones PSO

Result of BBPSO:

- Roughly the same high-level behaviour
- Roughly the same level of performance
- Particles no longer “fly” through the space
- Much simpler particle behaviour
- More predictable, easier to analyze

Abstracting the BBPSO Algorithm

Abstracting the BBPSO:

- It is possible to further abstract the BBPSO
- Recall the components of the canonical PSO particle:
    - Position
    - Velocity
    - Personal best found point
    - Communicated knowledge of global best
- Each component is required to update the particle

Abstracting the BBPSO Algorithm

Abstracting the BBPSO:

- The BBPSO algorithm removes the velocity component
- Position is updated by random sampling
- Sampled distribution is independent of current position
- Communication of global bests depend on personal bests
- Current position is used only to update the personal best

Abstracting the BBPSO Algorithm

Particles without positions:

- Particles can be updated without storing a position
- Sample the random distribution
- Evaluate new point
- Update the best found point if needed
- Discard the new point

Abstracting the BBPSO Algorithm

Effects of removing particle position:

- Identical behaviour to BBPSO
- Different concepts
- Particles no longer exist as explicit points
- Each particle defined by its best found point
- Particle's construct and sample a random distribution
- Probability of that particle existing at a given point

New Model

Moving forward with this concept:

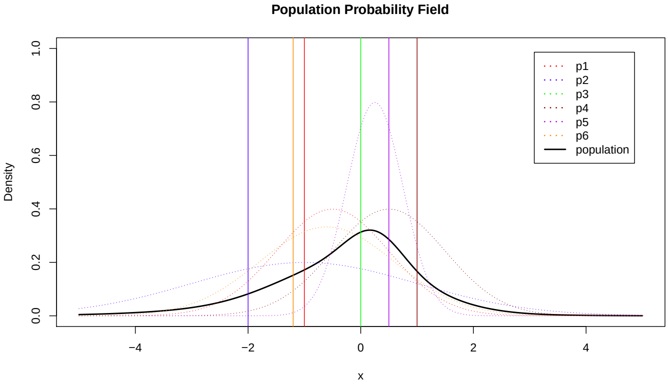
- Individuals are no longer candidate solutions
- Each represents a probability field
- Simple Gaussian distributions
- No longer “Particles”
- Instead, “Particle Fields”

New Model

At the population level:

- Population forms a complex probability field
- Sum of simple individual fields
- Probability field of all possible particle locations

New Model


Population Probability Field
Density
x
p1
p2
p3
p4
p5
p6
population
0.0
0.2
0.4
0.6
0.8
1.0
−4
−2
0
2
4

New Model

Population probability field:

- Probability field updated as individuals are updated
- Shows how PSO explores the solution space
- Demonstrates how PSO “learns”
- Similarities to a limited Bayesian learning process

New Model

Generating candidate solutions:

- Solutions have been separated from particles
- Maintain a population of candidate solutions
- Solutions generated by sampling complex population distribution
- For each candidate solution:
  - Randomly select a particle field individual
  - Generate position from that individual's distribution

New Model

Updating the particle field individuals:

- Each solution chooses an individual during generation
- Individual’s are updated using associated solutions
- Similar to the standard PSO method
- Adapted to handle any number of associated solutions

New Model

Moving forward again:

- Still roughly the same behaviour as the BBPSO
- Slightly more random
- Biggest difference is the high-level concept
- New concept provides new avenues for development

New Model

Modifying the population distribution:

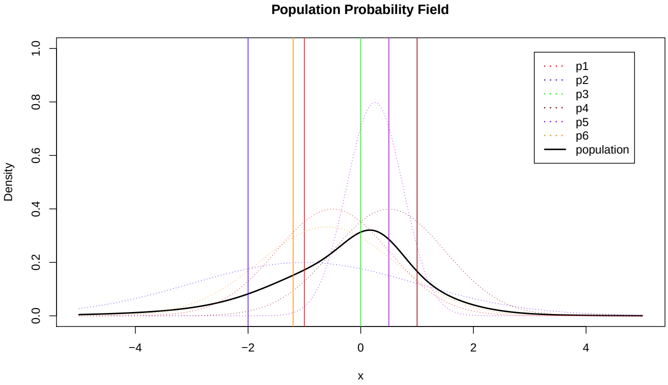
- Now possible to modify the population probability field
- Consider complex population distribution
- Sum of simple individual distributions
- Apply simple weighting scheme
- Drastic change to population distribution
- Focus search to better areas?

Weighting Schemes

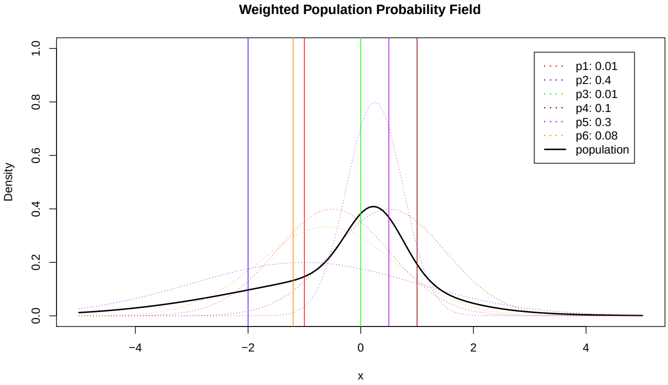
Effects of weighting schemes:

- Introduce new behaviour
- Changes the population probability field
- Changes how the swarm “learns”
- Incorporate different information into the search

Effects of Weighting



Effects of Weighting



Relative Population Sizes

Modifying relative population sizes:

- Model now consists of two separated populations
- Possible to modify population sizes independently
- Relative sizes effects the high level behaviour
- More particle field individuals:
  - May be more exploratory
- Less particle field individuals:
  - Faster convergence

Final PFO Algorithm

Final PFO Algorithm:

- Combining all these changes we have a distinct algorithm
- New behaviour and perspective
- New avenues for future development

PFO Simulation Step

For each candidate solution to be generated:

- Sample complex distribution defined by PF population
- Randomly choose a particle field individual
- According to weighting values
- Sample that individual's simple distribution

For each PF individual:

- Choose best associated solution
- Update personal best point if needed
- Calculate new weighting value

Results

Preliminary experimental results:

- Weighting scheme for individuals:
  - Average value of candidate solutions generated by each
  - Better information than just the best found?
- Tests done with different ratios of population sizes
- Compared to BBPSO
- Better performance on all test problems

Thank You

Questions?