IM5110701 Metaheuristics

Dr. Hendri Sutrisno

Institute of Statistical Science

Academia Sinica

Chapter 6: Swarm Intelligence II and Hybrid Method

Sean Luke, Essentials of Metaheuristics, Second Edition

Initial work

- Marco Dorigo's Ant Colony Optimization (ACO) is an approach to combinatorial optimization which gets out of the issue of Tweaking by making it optional
- ACO simply assembles candidate solutions by selecting components which compete with one another for attention
- The first version of ACO was called Ant Systems
- Designed for combinatorial optimization problem, such as the Travelling Salesman Problem

The photo is taken from his personal website:
https://iridia.ulb.ac.be/~mdorigo/HomePageDorigo/

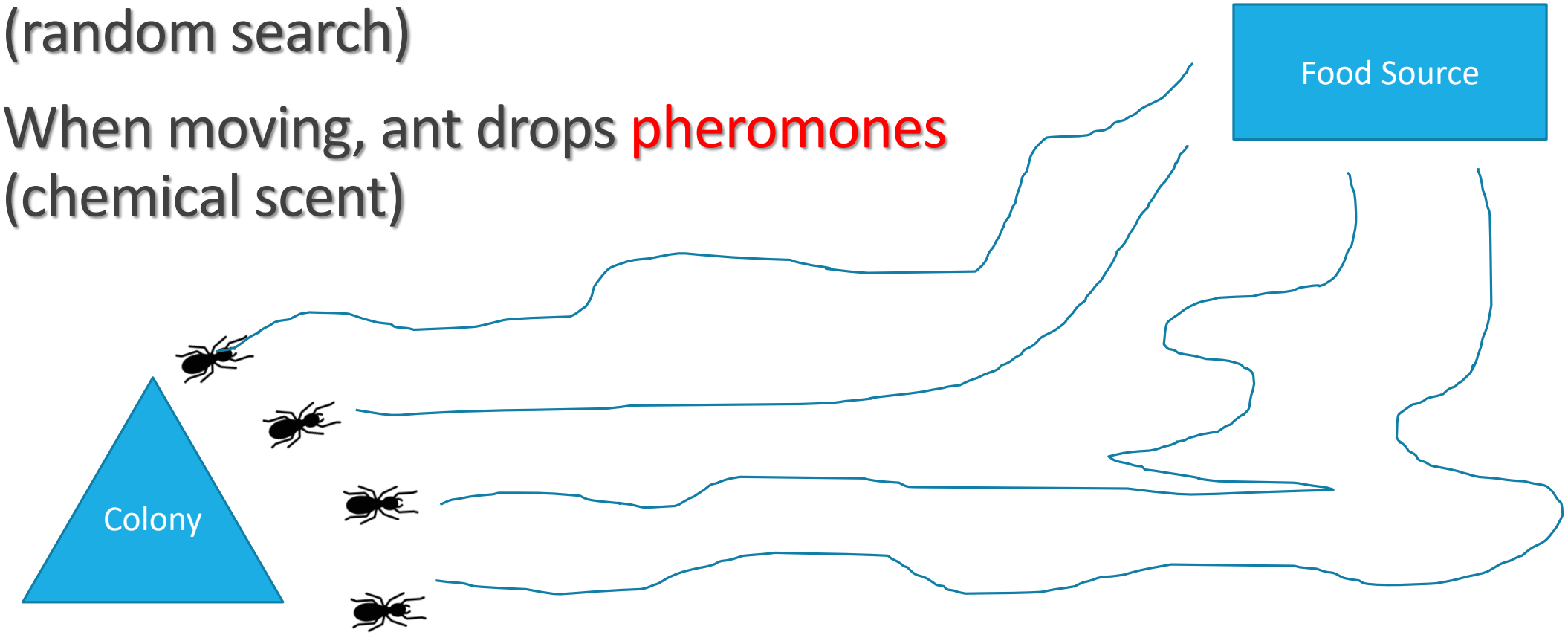
Dorigo, M. (1992). Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*.

Motivation

- Ant colonies, and more generally social insect societies, are distributed systems that, in spite of the simplicity of their individuals, present a highly structured social organization
- Ant colonies can accomplish complex tasks that far exceed the capabilities of a single ant. Ant colonies have a highly coordinated behavior
- Ants coordinate their activities via stigmergy, a form of indirect communication mediated by the modifications in the environment

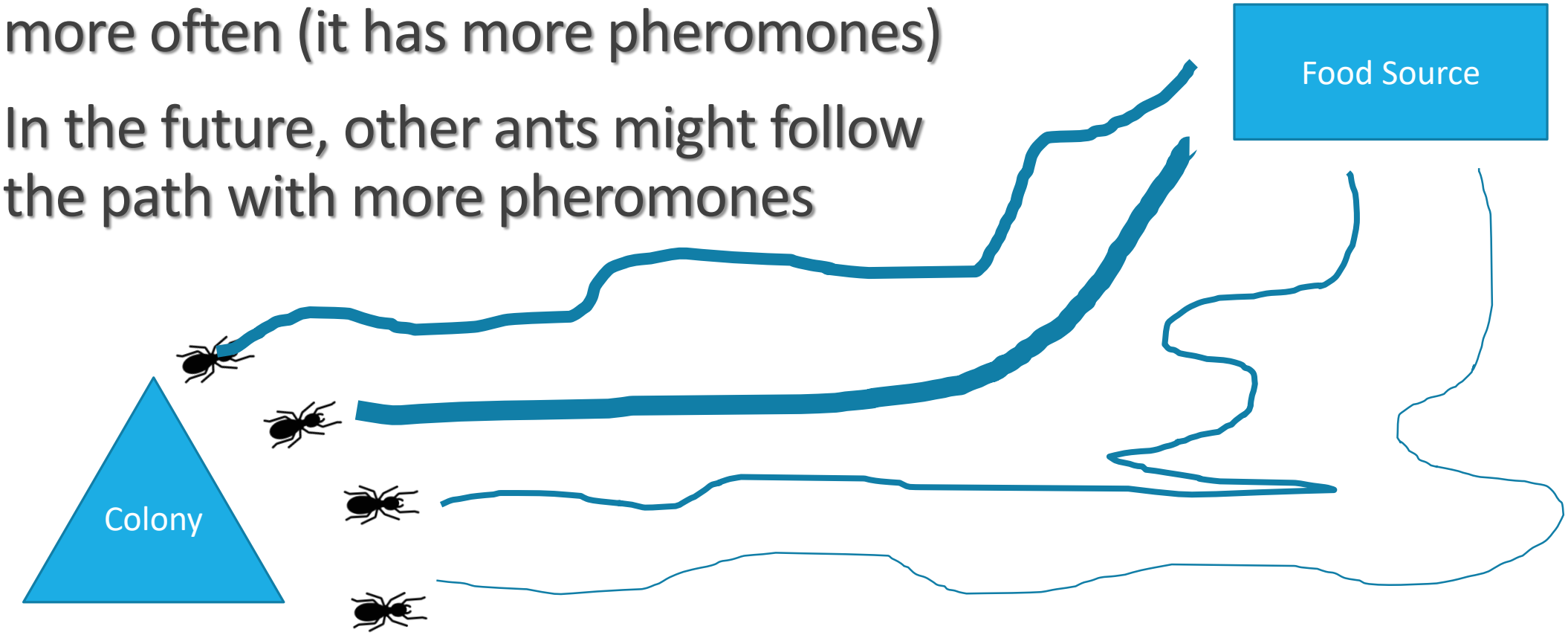
Analogy: 1. Random Search

- Initially, explore multiple feasible paths (random search)
- When moving, ant drops pheromones (chemical scent)



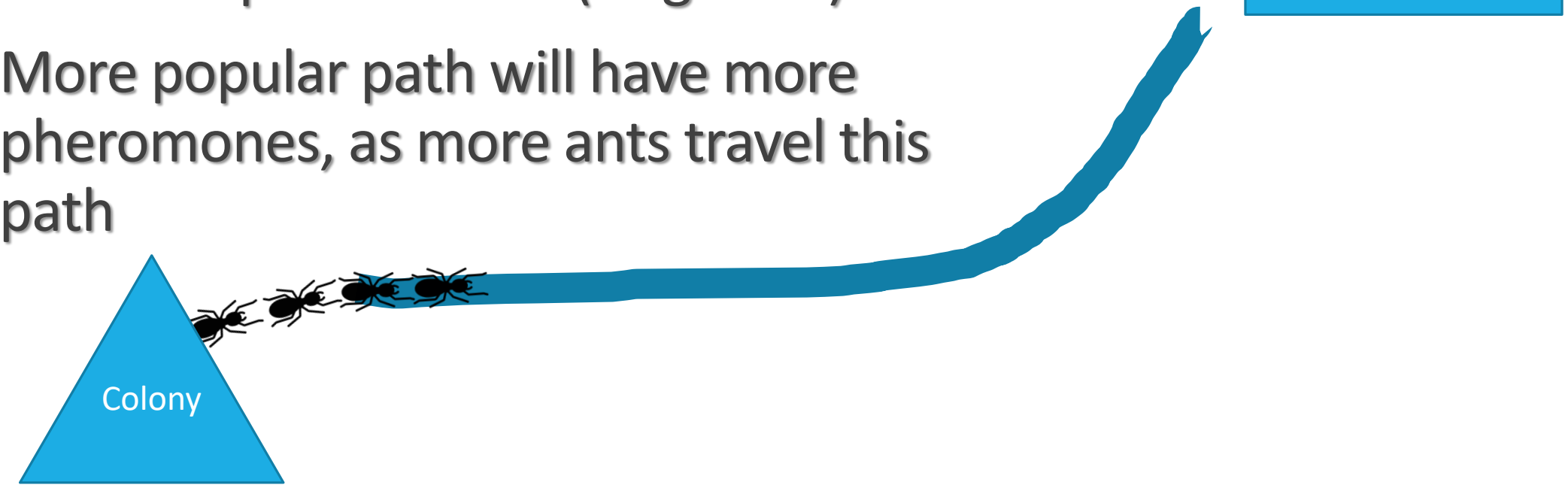
Analogy: 2. More ants travel on the more optimum paths

- Iteratively, shorter path will be traveled more often (it has more pheromones)
- In the future, other ants might follow the path with more pheromones



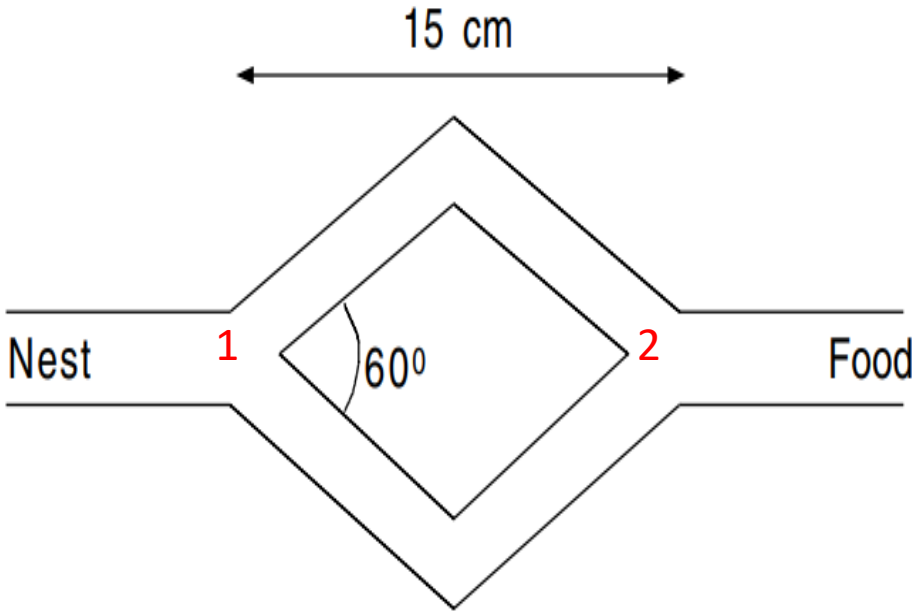
Analogy: 3. The optimum paths are maintained, the other paths are eliminated

- Overtime, the pheromones will be evaporated. Less travelled path will have less pheromones (forgotten)
- More popular path will have more pheromones, as more ants travel this path



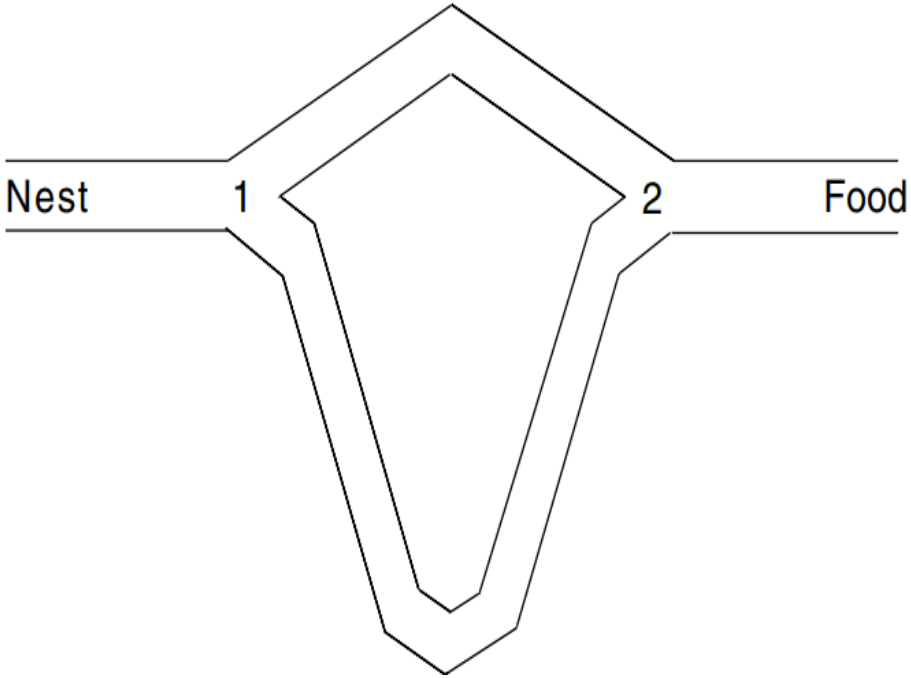
The Binary-bridge experiments: 1. Equal travel distance

- Ants were released from the nest
- At point 1, ants made random move (up or down)
- When ants did their journey back, at point 2, ants made another random decision
- Overtime, one of the path got more ants due to the biases of pheromones in the selection at point 1 and 2
- The preferred path got more and more pheromones, and the other path was forgotten in the long run



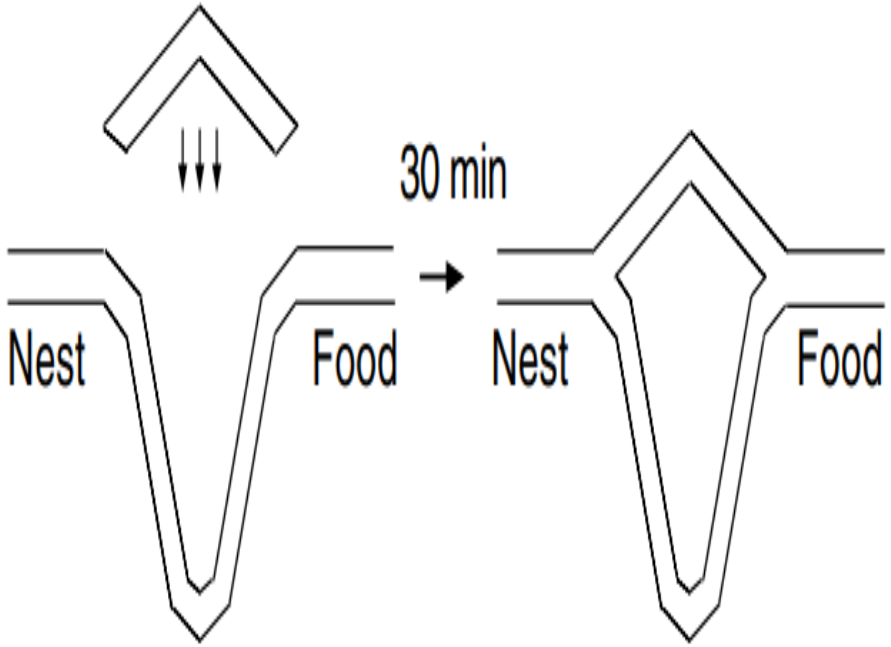
The Binary-bridge experiments: 2. Unequal travel distance

- Ants were released from the nest
- At point 1, ants made random move (shorter or longer paths)
- When ants did their journey back, at point 2, ants made another random decision
- Overtime, the shorter path got more and more pheromones, since it has shorter distance
  - More ants finished travelling the shorter route than the it in the longer route



The Binary-bridge experiments: 3. Local optimum

- In the beginning, the ants were only offered the longer path
- After a certain time, the shorter route was offered
- Some ants were observed travelling the shorter path
- But, most of the ants still travelled on the longer path (due to the biases of pheromones); thus, shorter path was neglected



The Ant System

- The first version of ACO is called the Ant System (AS)
- Five basic steps in AS
  1. Construct some trails (candidate solutions) by selecting components
  2. (Optional) Hill-Climbing the trails to improve the solutions
  3. Assess the fitness of the final trails
  4. Evaporate all the pheromone a bit
  5. Update the pheromone involved in trails based on the fitness of those solutions

The Ant System's Pseudocode

Algorithm 110 *The Ant System (AS)*

```
1: C ← {C_1, ..., C_n} components
2: e ← evaporation constant, 0 < e ≤ 1
3: popsize ← number of trails to construct at once
4: γ ← initial value for pheromones
5: t ← iterations to Hill-Climb

6: p⃗ ← ⟨p_1, ..., p_n⟩ pheromones of the components, all set to γ
7: Best ← □
8: repeat
9:     P ← {}                                   ▷ Our trails (candidate solutions)
10:    for popsize times do                     ▷ Build some trails
11:        S ← {}
12:        repeat
13:            C' ← components in C − S which could be added to S without being infeasible
14:            if C' is empty then
15:                S ← {}                       ▷ Try again
16:            else
17:                S ← S ∪ {component selected from C' based on pheromones and values or costs}
18:        until S is a complete trail
19:    S ← Hill-Climb(S) for t iterations       ▷ Optional. By default, not done.
20:    AssessFitness(S)
21:    if Best = □ or Fitness(S) > Fitness(Best) then
22:        Best ← S
23:    P ← P ∪ {S}
24:    for each p_i ∈ p⃗ do                      ▷ Decrease all pheromones a bit ("evaporation")
25:        p_i ← (1 − e)p_i
26:    for each P_j ∈ P do                      ▷ Update pheromones in components used in trails
27:        for each component C_i do
28:            if C_i was used in P_j then
29:                p_i ← p_i + Fitness(P_j)
30: until Best is the ideal solution or we have run out of time
31: return Best
```

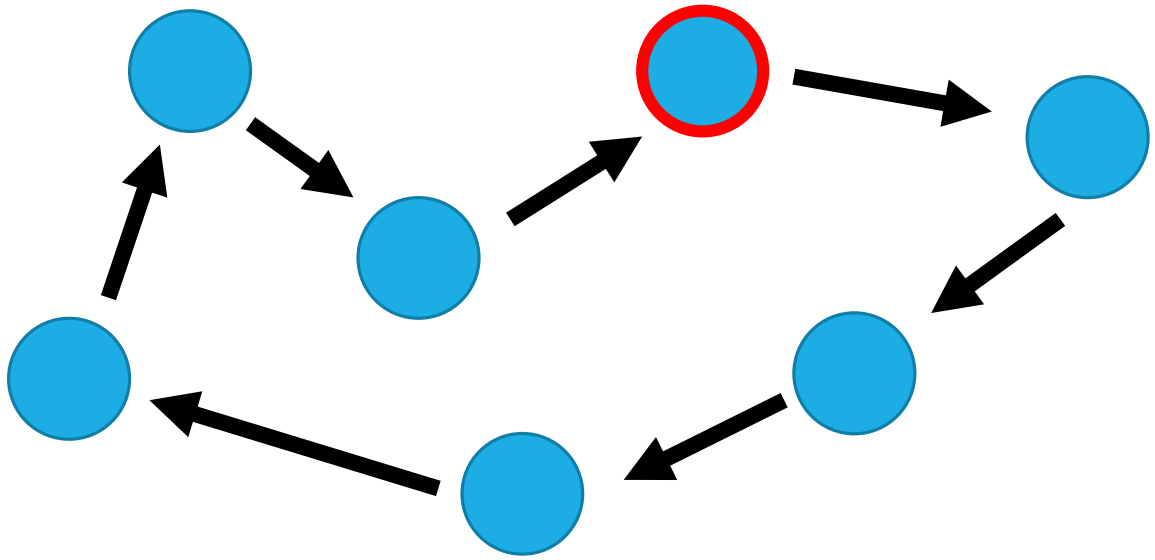
1. Build/Generate trails

2. Reduce pheromones

3. Update pheromones

Quick Introduction to Travelling Salesman Problem (TSP)

- Given a set of n cities, TSP requires a salesman to find the shortest route to return to the starting city while each city can be only visited once



The Ant System in solving TSP

- Let us have an example of 5 cities
  - From the problem, we will have a distance table between edges/cities (i, j). In most cases, we assume the distance table is symmetrical, $d_{ij} = d_{ji}$

d_{ij}	1	2	3	4	5
1	-				
2		-			
3			-		
4				-	
5					-

0. Initialization

- τ pheromones table
  - $\tau_{ij}(t)$ represents the pheromone for edge (i, j) at iteration (t)

$\tau_{ij}(t)$	1	2	3	4	5
1	-				
2		-			
3			-		
4				-	
5					-

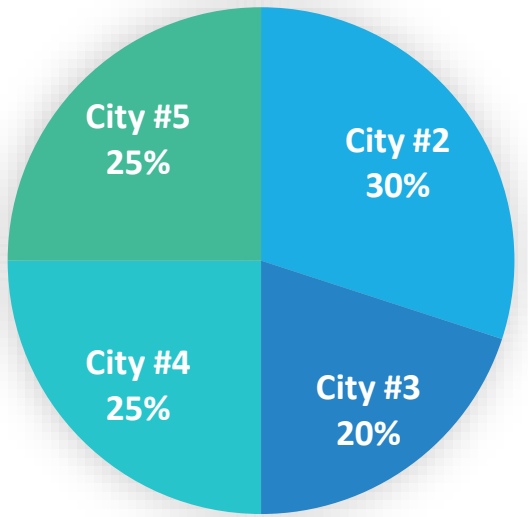
- Population size n_k (integer number)
- Evaporation constant e, where $0 < e < 1$

1a. Build/Generate the trails

- For example, let the n_k= 5, then we have 5 ants
- Initially, we select the starting point randomly
- The transition probability of AS is based on pheromone and probability selection
- For example, let ant #1 ($k = 1$) starts from city #1
  - The next city will be for ant #1 will

be based on probability selection based on pheromone (see GA).

Edge with larger pheromone will get higher probability



1b. Asses the fitness value

- For $k = 1$
  - $P(3|1) = \frac{\tau_{13}(t)}{\tau_{12}(t)+\tau_{13}(t)+\tau_{14}(t)+\tau_{15}(t)}$
  - $P(4|1-3) = \frac{\tau_{34}(t)}{\tau_{32}(t)+\tau_{34}(t)+\tau_{35}(t)}$
  - $P(5|1-3-4) = \frac{\tau_{45}(t)}{\tau_{42}(t)+\tau_{45}(t)}$
  - and so on
- By following the given distance table, we can asses the quality of the solutions

- Initial routes for the ants
  - For example, the population size is 5

$$x^1(0) = 1-3-4-5-2-1$$
$$x^2(0) = 2-1-3-4-5-2$$
$$x^3(0) = 3-5-4-1-2-3$$
$$x^4(0) = 4-3-2-1-5-4$$
$$x^5(0) = 5-4-3-2-1-5$$

- Evaluate the initial solutions

$$f\left(x^k(0)\right) = d_{13} + d_{34} + d_{45} + d_{52} + d_{21}$$

2-3a. Pheromones evaporation and update

- Pheromone Evaporation:
  - $\tau_{ij}(t) = (1 - e)\tau_{ij}(t)$
- Pheromone Update:
  - $\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t)$
  - $$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{f(x^k(t)} & \text{if edge (i,j) occurs in path } x^k(t) \\ 0 & \text{otherwise} \end{cases}$$
  - Q is some constant value, must be larger than 0, usually $Q = 1$

3b. Example on updating pheromones

- Consider the following population $(t = 0)$

$x^1(0) = 1 - 3 - 4 - 5 - 2 - 1$

$x^2(0) = 2 - 1 - 3 - 4 - 5 - 2$

$x^3(0) = 3 - 5 - 4 - 1 - 2 - 3$

$x^4(0) = 4 - 3 - 2 - 1 - 5 - 4$

$x^5(0) = 5 - 4 - 3 - 2 - 1 - 5$

- $\tau_{54}(1) = \tau_{54}(0) + \left(\frac{Q}{f(x^3(0))} + \frac{Q}{f(x^4(0))} + \frac{Q}{f(x^5(0))}\right)$

Some variants of Pheromone Update in AS

- There are three common ways in updating pheromones
  - $\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{f(x^k(t)} & \text{if edge (i,j) occurs in path } x^k(t) \\ 0 & \text{otherwise} \end{cases}$ Ant-cycle AS
  - $\Delta\tau_{ij}^{k}(t) = \begin{cases} Q & \text{if edge (i,j) occurs in path } x^k(t) \\ 0 & \text{otherwise} \end{cases}$ Ant-density AS
  - $\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{d_{ij}} & \text{if edge (i,j) occurs in path } x^k(t) \\ 0 & \text{otherwise} \end{cases}$ Ant-quantity AS

Elitist Ant System (EAS)

- Maintain the same procedure of AS
- Give more trust to n_e "elite ants"
- Before in AS
  - $\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t)$
- Now in EAS
  - $\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t) + n_e \Delta\tau_{ij}^{e}(t)$
  - n_e: number of elite ants

Smarter AS with learning rate (α)

- In AS or EAS, the step #2 (Evaporating the pheromones) follow

$$\tau_{ij}(t) = (1 - e)\tau_{ij}(t)$$

- It means that in every step, the reduction rate is constant
- We can improve the strategy by introducing learning rate

$$\tau_{ij}(t) = (1 - e)\tau_{ij}(t) + \alpha$$

- α can be a lot of things. For example, we can set α to be a proportion of the current and the previous average fitness values
- We want to value more the condition where we get big improvement than small improvement

The Ant Colony System

- ACS (Ant Colony System) was developed based on AS
- ACS is more exploitative than AS
- ACS works like the Ant System but with the following changes:
  1. The use of an elitist approach to updating pheromones: only increase pheromones for components used in the best trail discovered so far.
  2. The use of a learning rate in pheromone updates.
  3. A slightly different approach for evaporating pheromones.
  4. A strong tendency to select components that were used in the best trail discovered so far.

The Pseudocode of ACS

Algorithm 112 *The Ant Colony System (ACS)*

$C \leftarrow \{C_1, ..., C_n\}$ Components
popsize ← number of trails to construct at once
$\alpha \leftarrow$ elitist learning rate
$\beta \leftarrow$ evaporation rate
$\gamma \leftarrow$ initial value for pheromones
$\delta \leftarrow$ tuning parameter for heuristics in component selection ▷ Usually $\delta = 1$
$\epsilon \leftarrow$ tuning parameter for pheromones in component selection
$t \leftarrow$ iterations to Hill-Climb
$q \leftarrow$ probability of selecting components in an elitist way

$\vec{p} \leftarrow \langle p_1, ..., p_n \rangle$ pheromones of the components, all set to γ
Best ← □
repeat
    $P \leftarrow \{\}$ ▷ Our candidate solutions
    for *popsize* times **do** ▷ Build some trails
        $S \leftarrow \{\}$
        repeat
            $C' \leftarrow$ components in $C - S$ which could be added to S without being infeasible
            if C' is empty **then**
                $S \leftarrow \{\}$ ▷ Try again
            else
                $S \leftarrow S \cup \{$ component selected from C' using Elitist Component Selection $\}$
        until S is a complete trail
        $S \leftarrow$ Hill-Climb(S) for t iterations ▷ Optional. By default, not done.
        AssessFitness(S)
        if *Best* = □ or Fitness(S) > Fitness(*Best*) **then**
            Best ← S
    for each $p_i \in \vec{p}$ **do** ▷ Decrease all pheromones a bit ("evaporation")
        $p_i \leftarrow (1 - \beta)p_i + \beta\gamma$ — 2. Reduce pheromones
    for each component S_i **do** ▷ Update pheromones only of components in *Best*
        if S_i was used in *Best* **then**
            $p_i \leftarrow (1 - \alpha)p_i + \alpha$ Fitness(*Best*) — 3. Update pheromones
until *Best* is the ideal solution or we have run out of time
return *Best*

Elitist Component Selection in ACS

- Transition probability: The k^{th} ant travelling from node i to node j

$$j = \begin{cases} \underset{u \in \mathcal{N}_i^k(t)}{argmax} \ \tau_{iu}^{\epsilon}(t)\, \eta_{iu}^{\delta} & if\ rand(0,1) \le q \\ J & otherwise \end{cases}$$

$$p_{iJ}^k = \begin{cases} \dfrac{\tau_{iu}^{\delta}(t)\, \eta_{iu}^{\epsilon}}{\sum_{u \in \mathcal{N}_i^k(t)} \tau_{iu}^{\delta}(t)\, \eta_{iu}^{\epsilon}} \\ 0 \end{cases}$$

Notes:

- η_{ij}: the visibility which is inversely proportional to the distance between node i and j
- q: probability of selecting local optimum node
- $\mathcal{N}_i^k(t)$: a set of valid nodes to be visited by the k^{th} ant from node i at iteration t
- ϵ and δ are the parameters to control the influence of pheromone and distance influences
- $rand(0,1) \le q$ | Exploits by favouring the best edge
- $rand(0,1) > q$ | Explores

Elitist Component Selection in ACS: Numerical Examples

- Let
  - *Ant #1*, $k = 1$
  - *Iteration* $t = 2$
  - *Current node* $i = 3$
  - *Number of nodes = 4 (1,2,3,4)*
  - *Set of valid nodes,* $\mathcal{N}_3^1(2) = \{1,2,3\}$
- Assume
  - $\eta_{31}^{\epsilon} = \eta_{32}^{\epsilon} = \eta_{34}^{\epsilon}$ where $\eta_{31}^{\epsilon} = \frac{\epsilon}{d_{31}}$
  - $\tau_{31}^{\delta}(2)$=.35
  - $\tau_{32}^{\delta}(2)$=.5
  - $\tau_{34}^{\delta}(2)$=.65

- IF $rand(0,1) \leq q$:
  - $j = \begin{cases} \underset{u \in \mathcal{N}_3^1(2)}{argmax}\ \tau_{3u}^{\epsilon}(t)\,\eta_{3u}^{\delta} & if\ rand(0,1) \leq q \\ J & otherwise \end{cases}$
  - $j = 4$
- ELSE :
  - $p_{3J}^1 = \begin{cases} \frac{\tau_{3u}^{\delta}(2)\,\eta_{3u}^{\epsilon}}{\sum_{u \in \mathcal{N}_3^1(2)} \tau_{3u}^{\delta}(2)\,\eta_{3u}^{\epsilon}} \\ 0 \end{cases}$
  - Note: Similar to the selection in AS

Pheromone Evaporation in ACS

- Evaporation in AS: $\tau_{ij}(t) = (1 - e)\tau_{ij}(t)$
  - e: Evaporation constant
- Evaporation in ACS: $\tau_{ij}(t) = (1 - \beta)\tau_{ij}(t) + \beta\tau_{ij}(0)$
  - β: Evaporation rate

Pheromone Update in ACS

- Pheromone update in AS:
  - $\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^k(t)$
  - $\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{f(x^k(t)} & \text{if edge (i,j) occurs in path } x^k(t) \\ 0 & \text{otherwise} \end{cases}$
  - Q is some constant value, must be larger than 0, usually $Q = 1$
- Pheromone update in ACS
  - $\tau_{ij}(t+1) = (1-\alpha)\tau_{ij}(t) + \alpha\Delta\tau_{ij}(t)$
  - $\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{f(x^*(t))} & \text{if edge (i,j)} \in \text{ in local best } x^*(t) \\ 0 & \text{otherwise} \end{cases}$
  - α: elitist learning rate

Notes on Ant Colony Optimization

- In each update, we can improve the solution by using local search, such as Hill-Climbing
- ACS has a lot more parameters than AS
  - Each parameter is designed to control the mathematical calculation's effect (Exploration and Exploitation)
  - It also means that the parameters allow the user to tweak or set up strategy to either balancing or prioritizing one over the other between Exploration and Exploitation)

Next on Ant Colony Optimization

- All of the parameters above are fixed and defined in the beginning

Algorithm 112 *The Ant Colony System (ACS)*

1: $C \leftarrow \{C_1, ..., C_n\}$ Components
2: $popsize \leftarrow$ number of trails to construct at once
3: $\alpha \leftarrow$ elitist learning rate
4: $\beta \leftarrow$ evaporation rate
5: $\gamma \leftarrow$ initial value for pheromones
6: $\delta \leftarrow$ tuning parameter for heuristics in component selection ▷ Usually $\delta = 1$
7: $\epsilon \leftarrow$ tuning parameter for pheromones in component selection
8: $t \leftarrow$ iterations to Hill-Climb
9: $q \leftarrow$ probability of selecting components in an elitist way

- Improvement can be made to adjust the parameters automatically for setting up the transition between Exploration and Exploitation
- For example, we can emphasize more on the Exploration in the early iterations, and we prioritize more on the Exploitation in the latter stage of iterations

Hybrid Method based on Evolutionary and Swarm Intelligence

Some of the teaching materials were taken from: Cheng, M. Y., & Prayogo, D. (2014). Symbiotic organisms search: a new metaheuristic optimization algorithm. Computers & Structures, 139, 98-112.

Key Points

EVOLUTIONARY

- Simulation of evolution
- Natural selection
- Individual's survivability is dependent on the characteristics it inherits from its parents
- Mutation

SWARM INTELLIGENCE

- Collective behavior of animal (usually, but not always, is insect)
- Problem solving strategy
- No competition between individual and its parents
- Self organization

Hybrid methods

- A lot of recent works combine the Evolutionary and Swarm Intelligence concepts
- Motivation: More performance- than inspiration-oriented
- Goals:
  - Faster, more efficient and robust
  - More simple (less parameters)
  - Less curse of dimensionality effect

An example of hybrid method: Symbiotic Organisms Search (SOS)

- Invented by Min-Yuan Cheng and Doddy Prayogo in 2014
- Has been cited in more than 1200 research articles
- Inspired by the relation of organisms: mutualism, commensalism, and parasitism

ELSEVIER

Computers & Structures

Volume 139, 15 July 2014, Pages 98-112

Symbiotic Organisms Search: A new metaheuristic optimization algorithm

Min-Yuan Cheng [1] ✉, Doddy Prayogo ✉

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.compstruc.2014.03.007 Get rights and content

Highlights

- A new SOS algorithm is introduced to solve engineering optimization.
- Twenty-six mathematical problems and five engineering design problems are tested.
- The results obtained by SOS are compared with other optimization methods.
- Obtained results confirm the excellent performance of the SOS method.

Mutualism

- Mutualism: a type of symbiotic relationship where all species involved benefit from their interactions

	
Bee and flower • Bee gets nectar • Flower gets pollen distribution for reproduction	**The Oxpeckers and large mammals** • Oxpeckers pick at parasites on the mammal's body • The mammals are “freed” from potential parasites

Commensalism

- Commensalism: a type of relationship between two living organisms, where one organism benefits from the other without harming it.

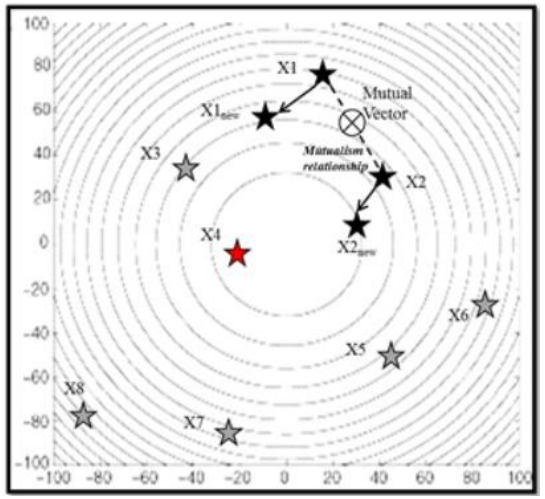
	
Remora and shark • Remora gets food leftovers • Remora gets “free-ride”	**Tree frog and plants** • The tree frog uses the plant for shelter and protection

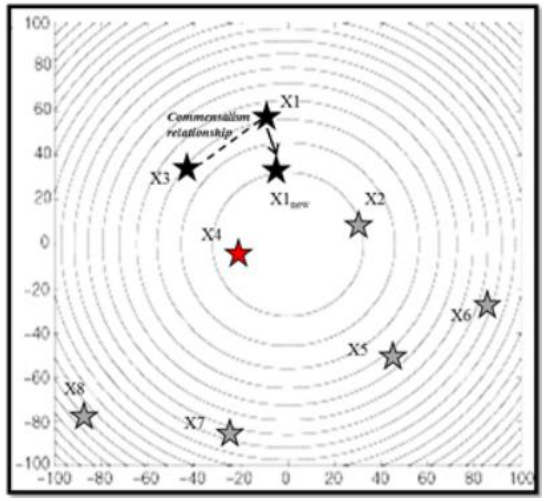
Parasitism

- Parasitism: relationship between two species in which one organism (parasite) lives on or within the other organism (host), causing the host some degree of harm.
  - Host-Symbiont relationships: fungi, leeches, lice, viruses, protozoa, tapeworm, etc.
  - Human and some insects (such as mosquito)

Swarm Intelligence in SOS

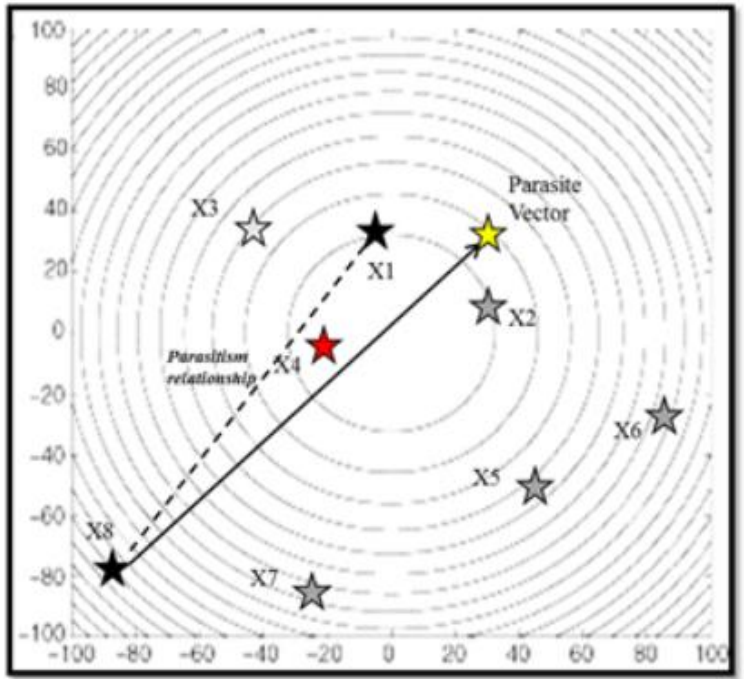
- Mutualism and Commensalism
  - In Mutualism, two individuals share information and move to new (and possible more optimum) location
  - In Commensalism, one individual more to new (and possible more optimum) location based on the information sharing from other individual





Evolutionary strategy in SOS

- Parasitism
  - SOS creates a new individual (parasite vector) by mutating an individual (A) to replace the position of other individual (B) in the ecosystem



SOS's pseudocode

```
1:    iter = 1
2:    Initialize ecosystem / population
3:    repeat
4:          Simulate interaction between organisms through the Mutualism Phase
5:          Simulate interaction between organisms through the Commensalism Phase
6:          Simulate interaction between organisms through the Parasitism Phase
7:          Update the best organism
8:    until iter = max_iter
```

Mutualism Phase

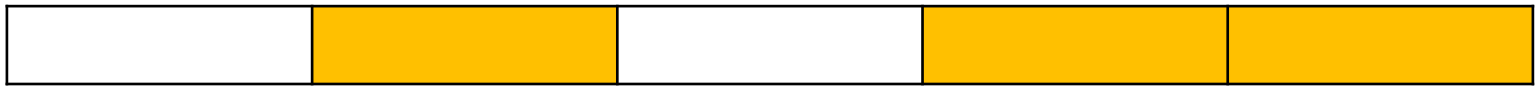
- Select two individuals from the ecosystem randomly (i and j)
- Calculate the mutual vector (MV), $MV = \frac{X_i + X_j}{2}$
  - MV represents the benefit from the mutualism relation between individual i and j
- Determine the benefit factor (BF), $BF = rand(1,2)$
  - BF represents the factor of the benefit gained from the relationship.
- New Candidates
  - $X_i' = X_i + rand(0,1) * (X_{best} - MV * BF_1)$
  - $X_j' = X_j + rand(0,1) * (X_{best} - MV * BF_2)$
- SOS moves X_i and X_j to X_i' and X_j', respectively.

Commensalism Phase

- Select two individuals from the ecosystem randomly (i and j), Move X_i based on X_j
- New candidate
  - $X_i' = X_i + rand(-1,1) * (X_{best} - X_j)$
- The $(X_{best} - X_j)$ part reflects the beneficial advantage provided by X_j to help X_i increasing its survival advantage.
- SOS moves X_i to X_i'

Parasitism Phase

- Select two individuals from the ecosystem randomly (i and j)
- Mutate X_j to X_j'



  - The yellow boxes is the mutation part.
  - The locations (which elements) are selected randomly.
  - The yellow boxes can be filled with random number.
  - The white boxes still follow X_j
- In this phase, SOS try to replace X_i with X_j'

The selection process in SOS

- SOS applies the greedy selection in all of the three phases
  - Greedy selection: Only accept more fit solutions
  - In mutualism and commensalism, only accept new location if it is fitter
  - In parasitism, only accept fitter mutated vector
- We see X_{best} in two out of three SOS phases
  - Greedy selection helps SOS to update its X_{best}
  - If X_{best} is rarely updated, the algorithm might stuck at local optimum quicker

SOS's advantage

- Much less parameters compared to other algorithms
  - Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, Bee Algorithm, Particle Bee Algorithm (PSO+BA)

GA	DE	PSO	BA	PBA	SOS
$n = 50$	$n = 50$	$n = 50$	$n = 50$	$n = 50$	$n = 50$
$m = 0.01$	$c = 0.9$	$w = 0.9–0.7$	$e = NP/2$	$e = NP/2$	
$c = 0.8$	$F = 0.5$	$v = X_{min}/10 \sim X_{max}/10$	$b = NP/4$	$b = NP/4$	
$g = 0.9$			$r = NP/4$	$r = NP/4$	
			$n_1 = 2$	$w = 0.9–0.7$	
			$n_2 = 1$	$v = X_{min}/10 \sim X_{max}/10$	
				$Pelite = 15$	
				$Pbest = 9$	

Reading Materials

- ACO
  - Luke, S. (2013). *Essentials of metaheuristics* (Vol. 2). Raleigh: Lulu.
  - Dorigo, M. and Stutzle, T. (2004). Ant Colony Optimization, MIT Press.
- SOS
  - Cheng, M. Y., & Prayogo, D. (2014). Symbiotic organisms search: a new metaheuristic optimization algorithm. *Computers & Structures*, *139*, 98-112.