Oxford Configuration Workshop
January 2012

# Honey Bee Foraging for Heuristic Optimisation


Duc Truong Pham
School of Mechanical Engineering
University of Birmingham

*with contributions from the* ***Cardiff Bay Bees****,*

*Siti Azfanizam Ahmad, (*Mei Choo Ang*)**, Hasan Al-Jabbouli, Ahmed Darwish, Afshin Ghanbarzadeh, Azar Imanguliyev, (*Mete Kalyoncu*)**, Ebubekir Koc, Ji Young Lee, (*Hosein Marzi*)**, Zaidi Muhamad, Mohamed Negm, Kok Weng Ng, Sameh Otri, Michael Packianather, (*Quang Tuan Pham*)**, Janyarat Phrueksanant, (*Francisco Javier Ramirez*)****,** Michael Sholedolu,Shahnorbanun Sahran, (*Shuo Xu*)**, Baris Yuce

**(*Honorary Bees*)

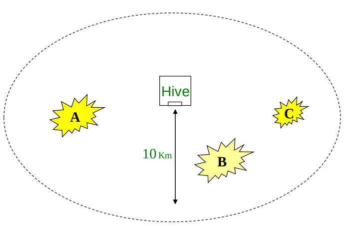

## The Bees Algorithm

- **Bees in Nature**
- **Basic Bees Algorithm**
- **Additional Features**
- **Applications**



## Bees in Nature

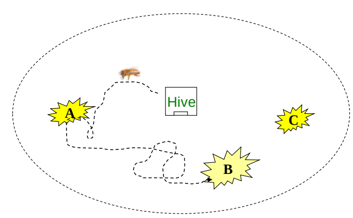
Honey bees can search for food far away from the hive. On average, a bee colony employs around 10% of its members as “scouts”. Scouts explore the fields surrounding the hive looking for sources of food.






## Bees in Nature

Scout bees move randomly from one flower patch to another looking for the best patch.

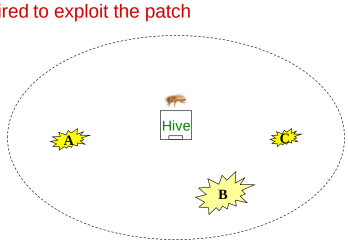





## Bees in Nature

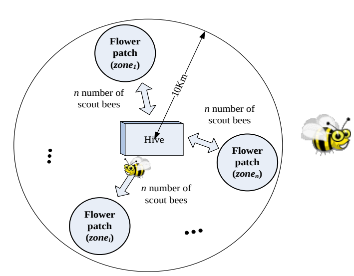
The bees evaluate a patch according to:

- The quantity and quality of food in the patch
- The distance from the hive and the amount of energy required to exploit the patch



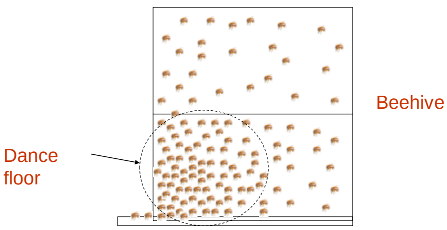



## Bees in Nature



## Bees in Nature

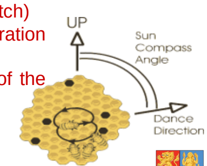
The scout bees ‘meet’ other bees on the ‘dance floor’ of the hive.






## Bees in Nature

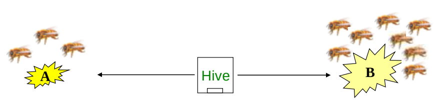
- Bees communicate through the waggle dance which contains information about:
    1. The direction of flower patches (Angle between the sun and patch)
    2. The distance from the hive (Duration of the dance)
    3. The quality rating (Frequency of the dance)






## Bees in Nature

- Bees use the information received from the waggle dance to decide which patch to visit.
- According to their fitness, patches can be visited by more or fewer bees, or abandoned.






## Bees in Nature – Foraging Mechanism

- **Random exploration of environment by scout bees.**
- **Exploitation of the most profitable food sources by recruited bees.**
- **Dynamic allocation of resources**
    - Depleted food sources are abandoned.
    - Newly discovered profitable food sources quickly attract foragers.



## Exploitation strategies

**Neighbourhood shrinking**

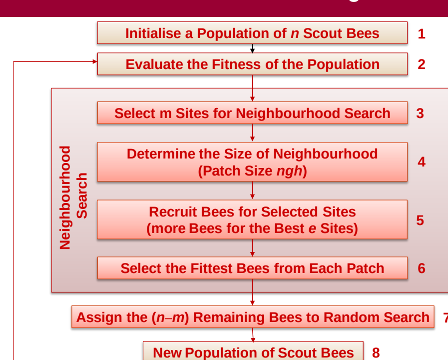
- If local search does not yield improvement, make it more focused by reducing the size of the neighbourhood of the point visited by a scout bee.
- Repeat the above step.
- This is a form of annealing.

**Site abandonment**

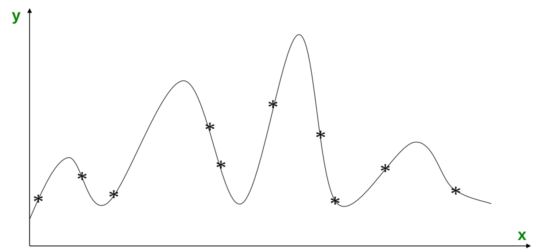
- If no improvement is achieved after a given number of neighbourhood shrinking cycles, the patch is abandoned.



## Flowchart of the Basic Bees Algorithm

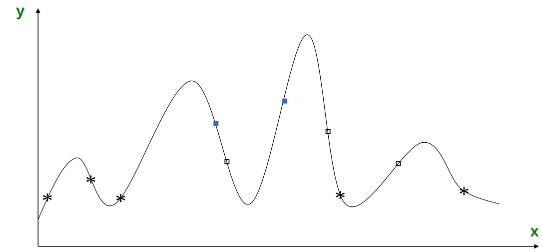


## A Simple Example



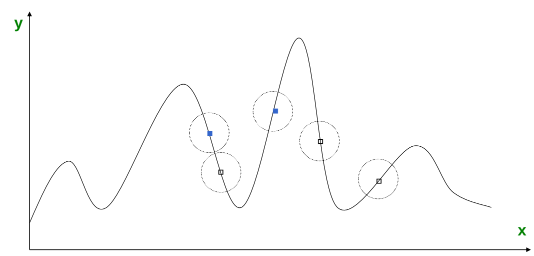

Graph 1. Initialise a population of (n=10*)* scout bees
with random search and evaluate the fitness

13/40



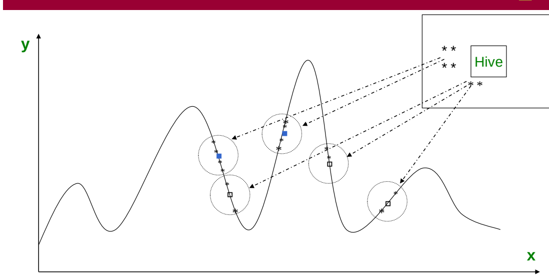

Graph 2. Select best (m=5) sites for neighbourhood search:
the best e=2 sites "▪" and (m-e=3) other selected sites"▫"

14/40



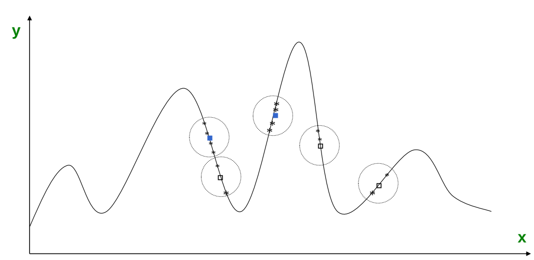

Graph 3. Determine the size of neighbourhood (patch size ngh)

15/40



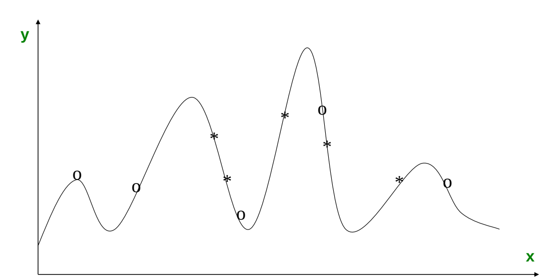

Graph 4. Recruit bees for selected sites
(more bees for the best *e=2* sites)

16/40



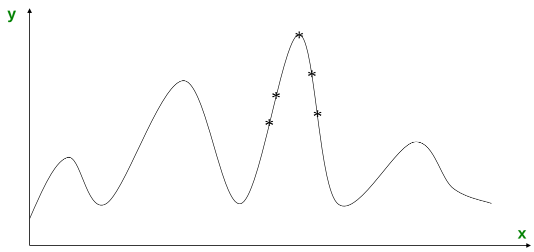

Graph 5. Select the fittest bee " * " from each patch

17/40




Graph 6. Assign the (n–m) remaining bees to random search

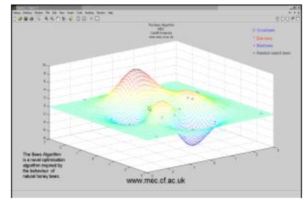
18/40




Graph 7. Find the global best point

19/40

## "Bee Movie"




20/40

## Problem Domain Encoding

- **Given space of feasible problem solutions** $U=\{\mathbf{x}\in R^n;\ min_i<x_i<max_i\ i=1,\dots,n\}$.
- **Given fitness evaluation function:** $f(\mathbf{x})$:U→R.
- **A solution is expressed as an *n*-dimensional vector of decision variables** $\mathbf{x}=\{x_1,\dots,x_n\}$.
- **No assumptions on solution space needed.**

21/40

## Learning Parameters

| | |
|---|---|
| *ns* | number of scout bees |
| *ne* | number of elite sites |
| *nb* | number of best sites |
| *nre* | recruited bees for elite sites |
| *nrb* | recruited bees for remaining best sites |
| *ngh* | initial size of neighbourhood |
| *stlim* | limit of stagnation cycles for site abandonment |

22/39

## Additional Features

### Local search operators

- Mutation
- Crossover
- Interpolation
- Extrapolation
- Creep (apply a small Gaussian perturbation to all the elements of the solution vector)
- Kalman filter (new_pos = old_pos + K∗estimation_error)

23/40

## Additional Features

### Young bees

Forcing new solutions to compete immediately with those already in the population is counterproductive.

A new solution may land at the foot of a tall mountain and still be worse than old solutions that have stalled at a lower peak.

A new category of bees is introduced, the "young members", which have evolved for fewer than a given number *M* of steps.

The new bees compete for survival only among themselves, until they reach "adulthood" after evolving *M* steps.

24/40

## Differences with PSO and ACO

**3 main differences:**

- **No** (complex) **interaction medium** for the exchange of information

- **Clear distinction** between exploratory search (scout bees) and exploitative search (recruited foragers)

- **In-built mechanisms** to overcome local fitness optimum traps (random scouting and site abandonment)



## Applications

***Continuous Problems*:**

- Mechanical design (welded beam; coil spring)
- Control system tuning (fuzzy membership functions and PID parameters)
- Digital filter design
- Classifier training (ANNs - RBF, LVQ, MLP; SVM)

***Discrete Problems*:**

- Manufacturing cell formation (simultaneous formation of part families and machine cells)
- Job scheduling for a production machine
- Printed-circuit board assembly sequencing



## Discussion

- **Problems**
  - Many learning parameters to optimise

- **Strengths**
  - Effective search strategy, quickly mobilising large resources to most promising areas
  - Suitability for accurately searching narrow valleys and holes, and for highly multi-modal functions
  - Simple strategy for tuning learning parameters



## Discussion

**Best search strategy for the Bees Algorithm**

- **Focus large resources on a few selected sites**
- **Move them quickly somewhere else**
  - Mutation is not efficient
  - Site abandonment prevents getting “stuck”



## Developments

**Hybridisation**

- **Ant Colony Optimisation** (using pheromone for increased sophistication of social interaction)
- **Particle Swarm Optimisation** (‘elite’ and other successful scouts acting together as a swarm or as leaders of their own swarms)
- **DCA** (for integrated or separate local search)



PER ARDUA AD ALTA

UNIVERSITY OF BIRMINGHAM

## Waggle dance

- Source: http://www.youtube.com/watch?v=-7ijl-g4jHg
- Duration: 54Sec
- Quality: low
- File (.WMV) size: 13,577KB)
- <RETURN>



## Foraging behaviour

<RETURN>



## Dynamic modelling

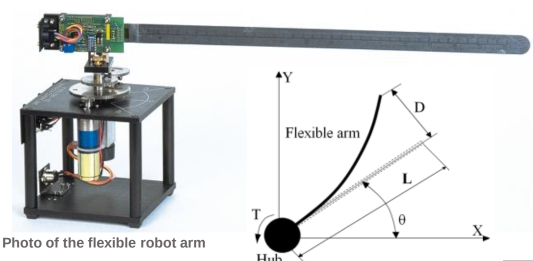
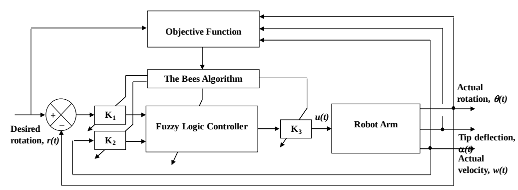



Photo of the flexible robot arm

Schematic diagram of the flexible arm



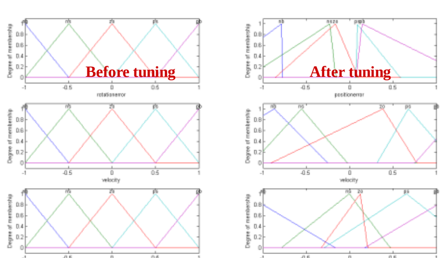
## Fuzzy Logic Controller for a flexible robot arm




Block diagram of the tuning of Fuzzy Logic Controller for a flexible single-link robot arm



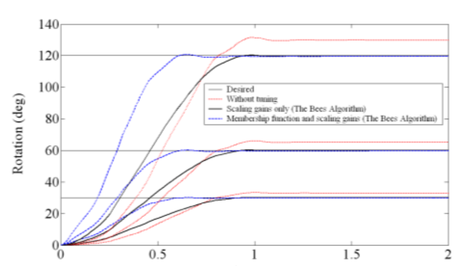
## Experimental Results




Membership functions of the fuzzy logic controllers



## Experimental Results




Rotation response of the system under different controllers for different step inputs <RETURN>

## Cell Formation Problem

- **Objective:** Group parts and machines into **clusters** to maximise the **bond energy measure**.
- **Bond Energy Measure** α:

$$\alpha = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N-1} a_{ij}a_{i,j+1} + \sum_{i=1}^{M-1}\sum_{j=1}^{N} a_{ij}a_{i+1,j}}{\sum_{i}^{M}\sum_{j}^{N} a_{ij}}$$

**M**: Machines

**N**: Parts

**$a_{ij}$**: A binary variable that takes the value of 1 if part j requires processing on machine i, and 0 otherwise



## Cell Formation Example

- Grouping efficiency measure, ε, for assessing the quality of the solutions:

$$\varepsilon = \underbrace{\frac{n_1}{\sum_{k=1}^{K} M_k N_k}}_{\text{Cell Density } \varepsilon 1} - \underbrace{\left(1-\frac{n_1}{n_1+n_2}\right)}_{\text{Intercellular Material Flows } \varepsilon 2}$$

***$n_1$*** number of non-zero entries within the manufacturing cells in the machine-part incidence matrix

***K*** number of manufacturing cells formed

***$M_k$*** and ***$N_k$*** (k = 1, 2,…, K) numbers of the machines and parts allocated to manufacturing cell k

***$n_2$*** number of exceptional elements in the machine-part incidence matrix



## Cell Formation

- **MxN machine-part incidence matrix**
- **String of length (M + N) needed to encode each candidate solution (CS)**

CS ($x_i$)={ 1,2,…,M / 1,2,…,N }

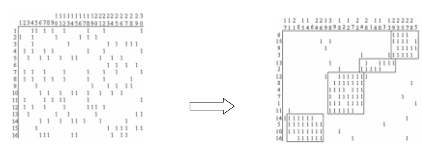
| m/n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 1 | 1 |
| 6 | 1 | 0 | 0 | 1 | 1 | 0 |

- **First M bits => sequence of machines that appear in the rows of the incidence matrix**
- **Last N bits => sequence of parts appearing in the columns of the matrix**



## Cell Formation Example

- 16 x 30 machines-parts incidence matrix
- Bees Algorithm Parameters: n = 100, m = 40, e = 20, nep = 200, nsp = 100 and maximum number of iterations = 1000



**Initial** **Final**

α = 1.301; $\varepsilon_1$ = 0.816; $\varepsilon_2$ = 0.155; ε = 0.661

<RETURN>