



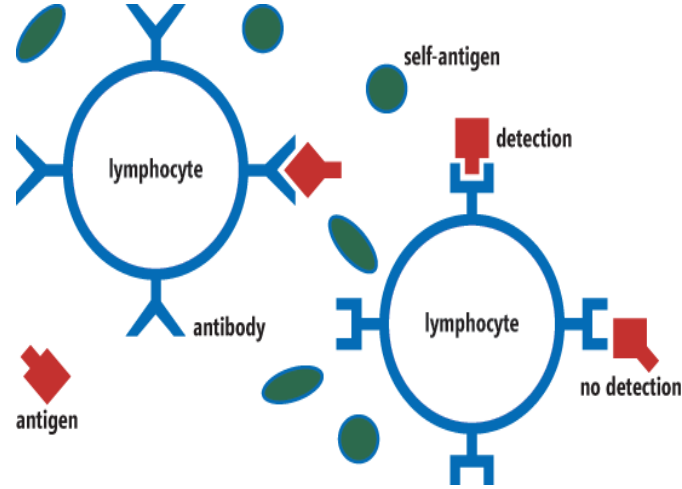
NTNU

# Lecture 9

## Bees Algorithm (BA)

## Artificial Immune Systems (AIS)

Kazi Shah Nawaz Ripon and Pauline Hadow



# Outline

- Bees Algorithm (BA)
- Artificial Immune Systems (AIS)

# Outline

- **Bees Algorithm (BA)**
- Artificial Immune Systems (AIS)

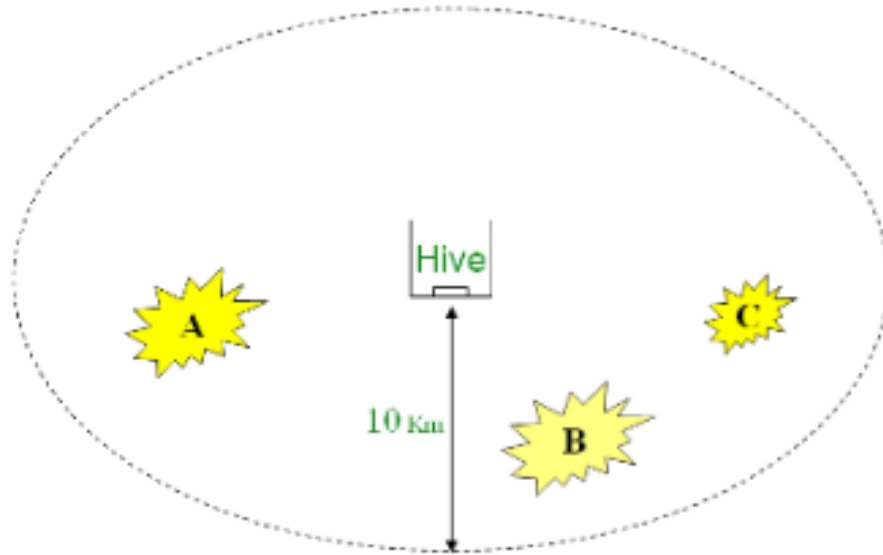


# Bees Algorithm (BA)

- The Bees Algorithm is a nature-inspired optimization algorithm that mimics the **food foraging behavior of honey bees** to find the optimal solution.
- Developed by Prof. D.T. Pham and his co-workers in 2005.
- BA performs both an **exploitative neighborhood search** combined with **random explorative search**.
- Mainly designed for **continuous optimization problem**.

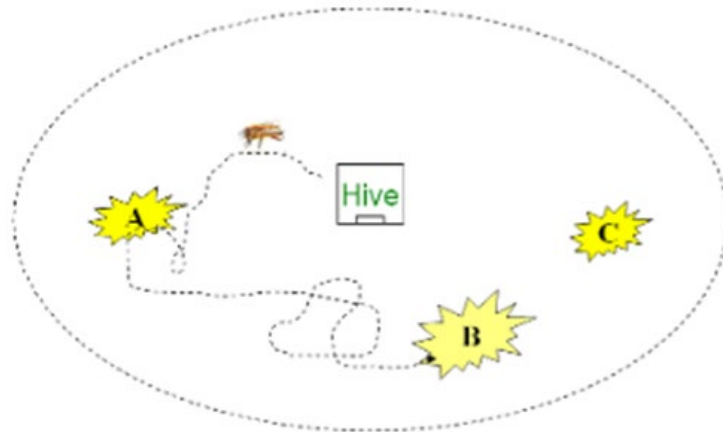
# Bees in Nature - 1

**A colony of honey bees** can exploit a large number of food sources in big fields and they can fly up to 11 km to exploit food sources.



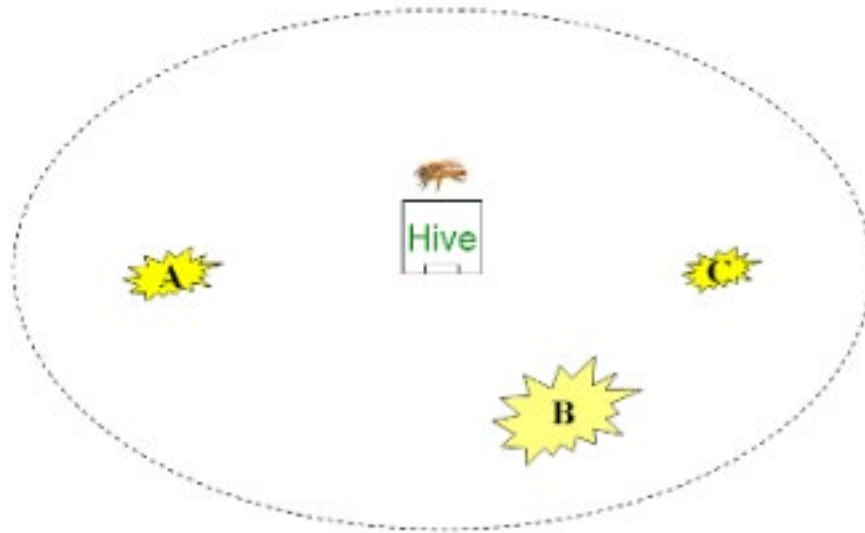
# Bees in Nature - 2

- The foraging process begins with **randomly** searching out promising flower patches by **scout bees**.
- Flower patches with plentiful amounts of nectar or pollen that can be collected with less effort should be visited by more scout bees,
  - whereas patches with less nectar or pollen should receive fewer scouts.



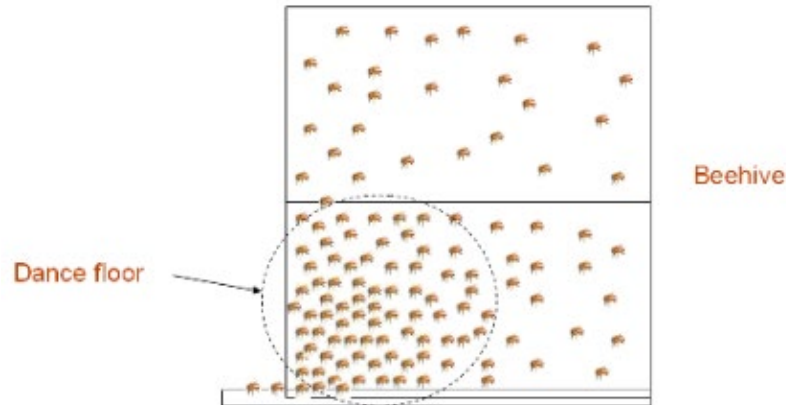
# Bees in Nature - 3

The scouts who return to the hive, **evaluate** the different patches depending on certain quality threshold (measured as a combination of some elements, such as sugar content).



# Bees in Nature - 4

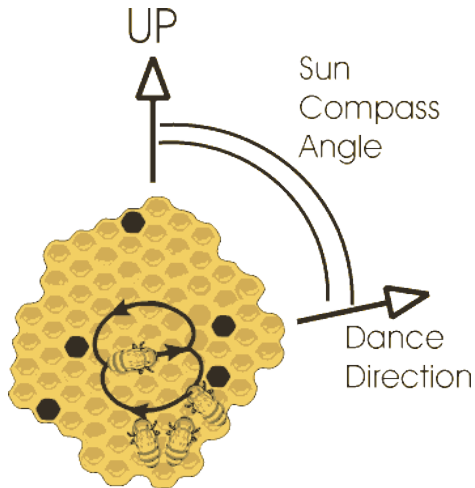
- Scout bees deposit their nectar and go to the dance floor in front of the hive to **communicate** to the other bees by performing “**waggle dance**”.
- A small number of scouts continue to search for new patches
  - while bees returning from flower patches continue to communicate the quality of the patch.





# Bees in Nature - 5

- Scout bees provide the following information by waggle dance:



- 1. Quality** of the food source (frequency of the dance).
- 2. Distance** of the source from the hive (duration of the dance).
- 3. Direction** of the source (angle between the sun and the patch).

- These information helps the colony to send its bees precisely.
- The scout backs to the flower patch with **follower bees** to gather food efficiently and quickly .

# Bees in Nature - 6

- The same patch will be **advertised** in the waggle dance again if it is still good enough as a food source and more bees will be recruited to that source.
- Thus, according to the fitness, patches can be visited by more bees or may be **abandoned**.



# Bees in Nature



# Strategy

- The information processing objective of the algorithm is to locate and explore good sites within a problem search space.
- Initially, scouts are sent out to **randomly** sample the problem space and locate good sites.
- The good sites are *exploited* via the application of a local search,
  - although many scouts are sent out each iteration always in search of additional good sites (*exploration*).

# BA: *Local Search vs Global Search*

- A fraction of the population (scout bees) searches randomly for regions of high fitness (**global search**).
- The most successful scouts recruit a variable number of idle agents (follower bees) to search in the proximity of the fittest solutions (**local search**).
- Cycles of global and local search are repeated until an acceptable solution is discovered, or a given number of iterations have elapsed.
- Implementing Algorithm: Main procedure includes local and global search routines which are executed **concurrently** at each cycle.

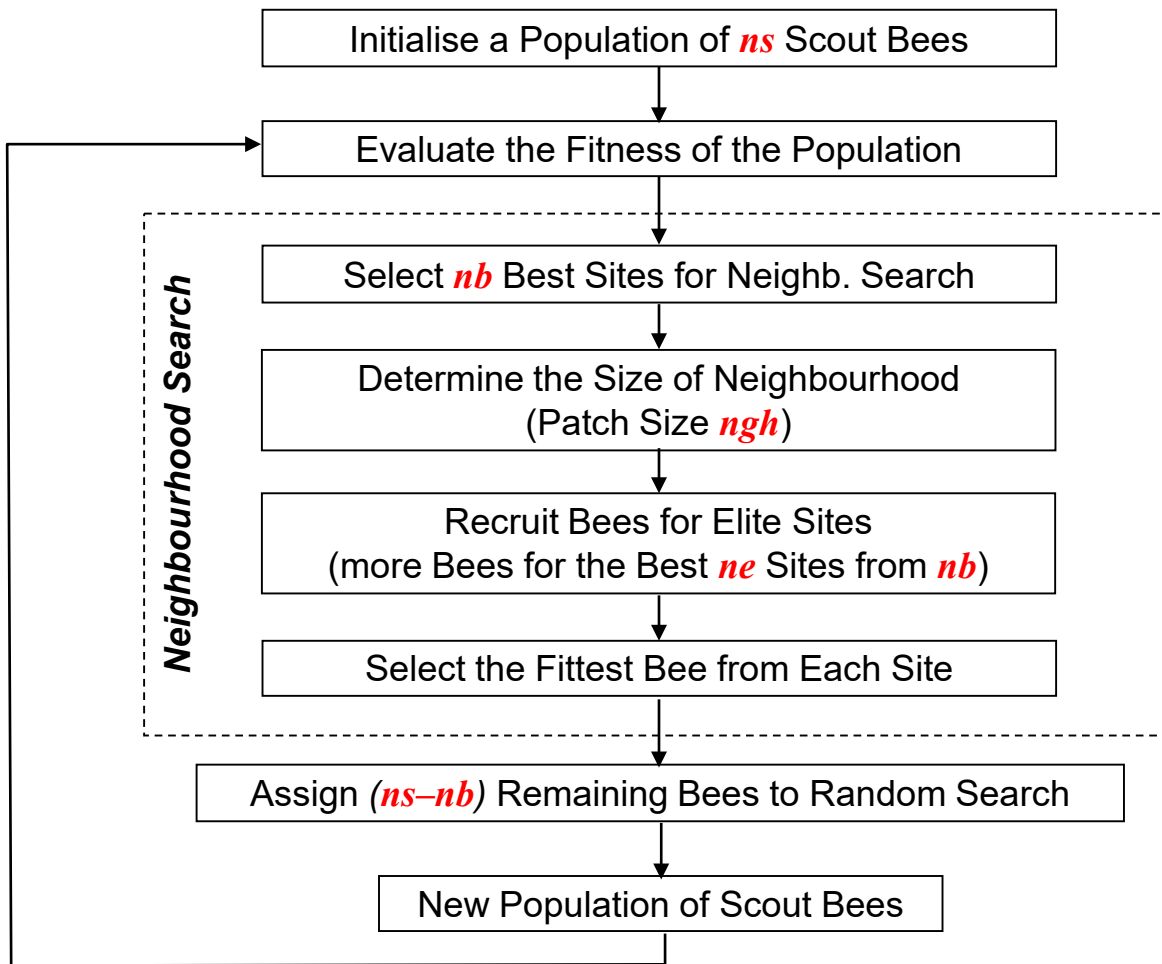
# Natural Bees

- Similar to other social insect colonies, honey bees also have the same four basic ingredients for self-organization :
  - (1) Positive feedback
  - (2) Negative feedback
  - (3) Randomness
  - (4) Multiple interactions

# Main Parameters

<i>ns</i>	Number of scout bees
<i>ne</i>	Number of elite sites
<i>nb</i>	Number of best sites
<i>nre</i>	Recruited bees for elite sites
<i>nrb</i>	Recruited bees for remaining best sites
<i>ngh</i>	Initial size of neighbourhood
<i>stlim</i>	Limit of stagnation cycles for site abandonment

The algorithm uses a population of  $n$  artificial bees, divided into  **$ns$  scouts** and  $n - ns$  followers.



**Flowchart of the Basic BA**





# Pseudocode: Standard BA

1 for  $i=1, \dots, ns$

    i scout[i]=Initialise\_scout()

    ii flower\_patch[i]=Initialise\_flower\_patch(scout[i])

- In the **initialisation** routine  $ns$  scout bees are randomly placed in the search space, and
  - Evaluate the fitness of the solutions where they land.
- For each solution, a **neighbourhood** (called **flower patch**) is delimited.

# Pseudocode: Standard BA

1 for  $i=1, \dots, ns$

  i scout[i]=Initialise\_scout()

  ii flower\_patch[i]=Initialise\_flower\_patch(scout[i])

2 do until stopping\_condition=TRUE

  i Recruitment()  $\longrightarrow$  **Waggle Dance**

- In the **recruitment** procedure, the scouts that visited the  $nb \leq ns$  fittest solutions (best sites) perform the waggle dance.
  - That is, they recruit followers to search further the neighborhoods of the most promising solutions.
- The scouts that located **the very best**  $ne \leq nb$  solutions (**elite sites**) recruit  $nre$  followers each,
  - whilst the remaining  $(nb - ne)$  scouts recruit  $nrb$  ( $\leq nre$ ) followers each.
- Thus, the number of followers recruited depends on the profitability of the food source.

# Pseudocode: Standard BA

```

1 for i=1,...,ns
    i scout[i]=Initialise_scout()
    ii flower_patch[i]=Initialise_flower_patch(scout[i])
2 do until stopping_condition=TRUE
    i Recruitment()
    ii for i =1,...,nb
        1 flower_patch[i]=Local_search(flower_patch[i])
        2 flower_patch[i]=Site_abandonment(flower_patch[i])
        3 flower_patch[i]=Neighbourhood_shrinking(flower_patch[i])
  
```

- In the **local search** procedure, the recruited followers are randomly scattered within the flower patches enclosing the solutions visited by the scouts (**local exploitation**).
  - If any of the followers in a flower patch lands on a solution of higher fitness than the solution visited by the scout, that followers becomes the new scout.

# Pseudocode: Standard BA

```

1 for i=1,...,ns
    i scout[i]=Initialise_scout()
    ii flower_patch[i]=Initialise_flower_patch(scout[i])
2 do until stopping_condition=TRUE
    i Recruitment()
    ii for i =1,...,nb
        1 flower_patch[i]=Local_search(flower_patch[i])
        2 flower_patch[i]=Site_abandonment(flower_patch[i])
        3 flower_patch[i]=Neighbourhood_shrinking(flower_patch[i])
  
```

} Not in Original BA Version

- If no follower finds a solution of higher fitness, the size of the flower patch is shrunk (**neighbourhood shrinking** procedure).
  - Usually, flower patches are initially defined over a large area, and their size is gradually shrunk by the neighbourhood shrinking procedure.
- If no improvement in fitness is recorded in a given flower patch for a pre-set number of search cycles, the local maximum of fitness is considered found, the patch is abandoned (**site abandonment**), and
  - a new scout is randomly generated.



# Pseudocode: Standard BA

```
1 for i=1,...,ns
    i scout[i]=Initialise_scout()
    ii flower_patch[i]=Initialise_flower_patch(scout[i])
2 do until stopping_condition=TRUE
    i Recruitment()
    ii for i =1,...,nb
        1 flower_patch[i]=Local_search(flower_patch[i])
        2 flower_patch[i]=Site_abandonment(flower_patch[i])
        3 flower_patch[i]=Neighbourhood_shrinking(flower_patch[i])
    iii for i = nb,...,ns
        1 flower_patch[i]=Global_search(flower_patch[i])
```

- As in biological bee colonies, a small number of scouts keeps exploring the solution space looking for new regions of high fitness (**global search**).
- The global search procedure re-initialises the last  $ns-nb$  flower patches with randomly generated solutions.



# Pseudocode: Standard BA

```
1 for i=1,...,ns
    i scout[i]=Initialise_scout()
    ii flower_patch[i]=Initialise_flower_patch(scout[i])
2 do until stopping_condition=TRUE
    i Recruitment()
    ii for i =1,...,nb
        1 flower_patch[i]=Local_search(flower_patch[i])
        2 flower_patch[i]=Site_abandonment(flower_patch[i])
        3 flower_patch[i]=Neighbourhood_shrinking(flower_patch[i])
    iii for i = nb,...,ns
        1 flower_patch[i]=Global_search(flower_patch[i])
```

- At the end of one search cycle, the scout population is again composed of  $ns$  scouts:
  - $nb$  scouts produced by the local search procedure (some of which may have been re-initialised by the site abandonment procedure), and
  - $ns-nb$  scouts generated by the global search procedure.

# Example: BA

The algorithm starts with the  $ns$  scout bees being placed randomly in the search space.

- (for example  $ns=100$ )

# Example: BA

Fitnesses of the sites visited by the scout bees after return are evaluated (*fitness function evaluation*).

- The evaluation of the 100 scout bees is stored in array as follow:

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	...	...	...	<b>99</b>	<b>100</b>
<b>20</b>	<b>50</b>	<b>60</b>	<b>30</b>	<b>80</b>	<b>10</b>	...	...	...	<b>35</b>	<b>72</b>

- Then the array will be reordered based on the evaluation from the higher to the lower value

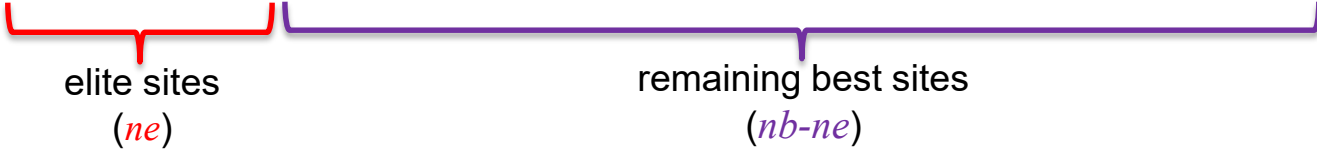


# Example: BA

The  $nb$  best sites will be selected from  $ns$ .

- For example  $nb=10$

1	2	3	4	5	6	7	8	9	10
80	78	75	72	69	66	65	60	59	58



- Then we choose the best  $ne$  site (elite bee) out off  $nb$ .
  - For example  $ne=2$



# Example: BA

A neighborhood search sites of size  $ngh$  is determined which will be used to update the  $nb$  bees declared in the previous step.

- Recruit Bees for the selected sites and evaluate the fitness of the sites.
  - Number of bees ( $nre$ ) will be selected to be sent to  $ne$  sites ( $nre=40$ ).
  - Choosing  $nrb$  bees which their number is less than  $nre$ , ( $nrb=20$ ) to be sent to  $nb-ne$  sites.

# Example: BA

Choosing the best bee (the highest fitness) from each site (***from both elite sites and the remaining best sites***) to form the next bee population.

## Example (*Implementation: creating neighborhood for best sites*)

- The best bee from each of  $ne$  sites is selected as follow:
  - For the first site from  $ne$ :
    - An array contains  $nre=40$  bees will be constructed,
      - where the value of each bee is equal to the value of the original scout bee with a little modification depending on the neighborhood  $ngh$ .

# Example (Implementation: creating neighborhood for best sites)

- Based on the fitness evaluation of  $nre = 40$  bees:
  - The results will be stored in temporary array.
  - The array will be ordered and the best value will be taken

<b>1</b>	<b>2</b>	<b>3</b>	...	<b>40</b>
<b>82</b>	<b>81.2</b>	<b>79.9</b>	...	<b>79.2</b>

# Example (Implementation: creating neighborhood for best sites)

It is repeated for all *nb* sites.

- At the end we will get the best *nb=10* bees which will be stored at the beginning of the array (*ns=100*)

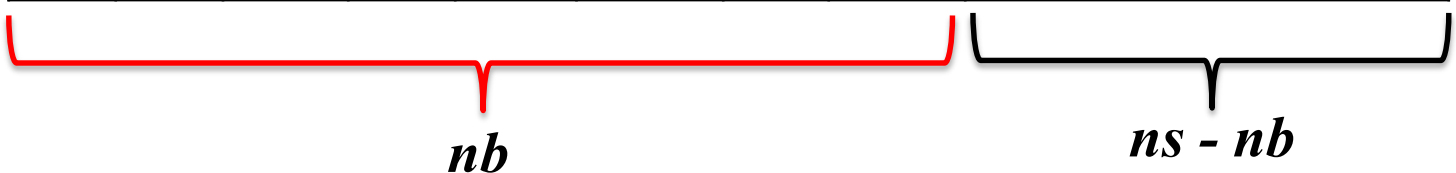
1	2	3	4	5	6	...	10	11	...	99	100
82	79	77	73	70	67	...	58.2				

# Example: BA

Initials new population:

- The remaining bees ( $ns - nb$ ) in the population will be assigned randomly around the search space (values from 11 to 100 in the previous array)
- The new population becomes as follow:

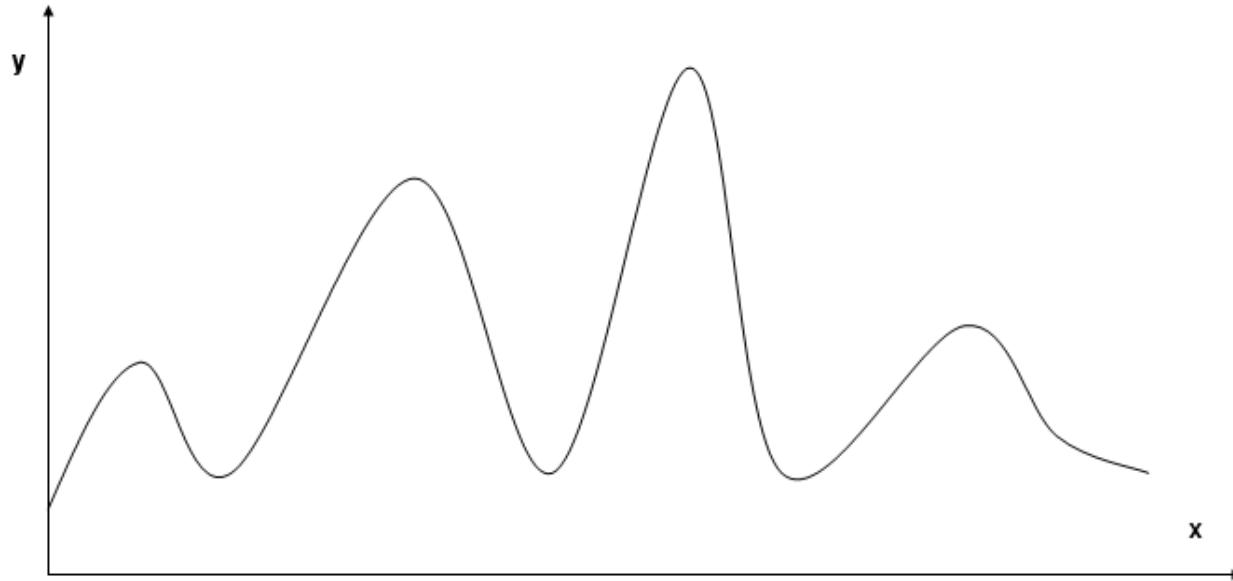
1	2	3	4	5	6	...	10	11	...	99	100
82	79	77	73	70	67	...	58.2	Random values			



$nb$ 
 $ns - nb$

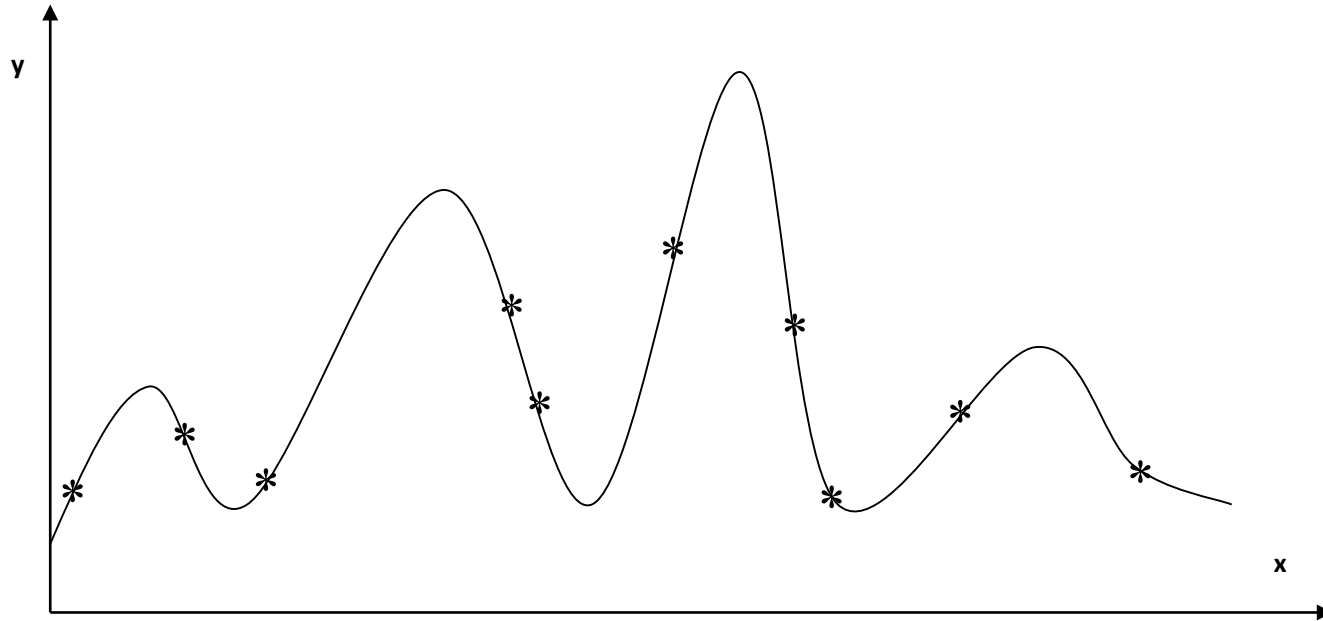
# Example: Function Optimisation

- The following figure shows the mathematical function.



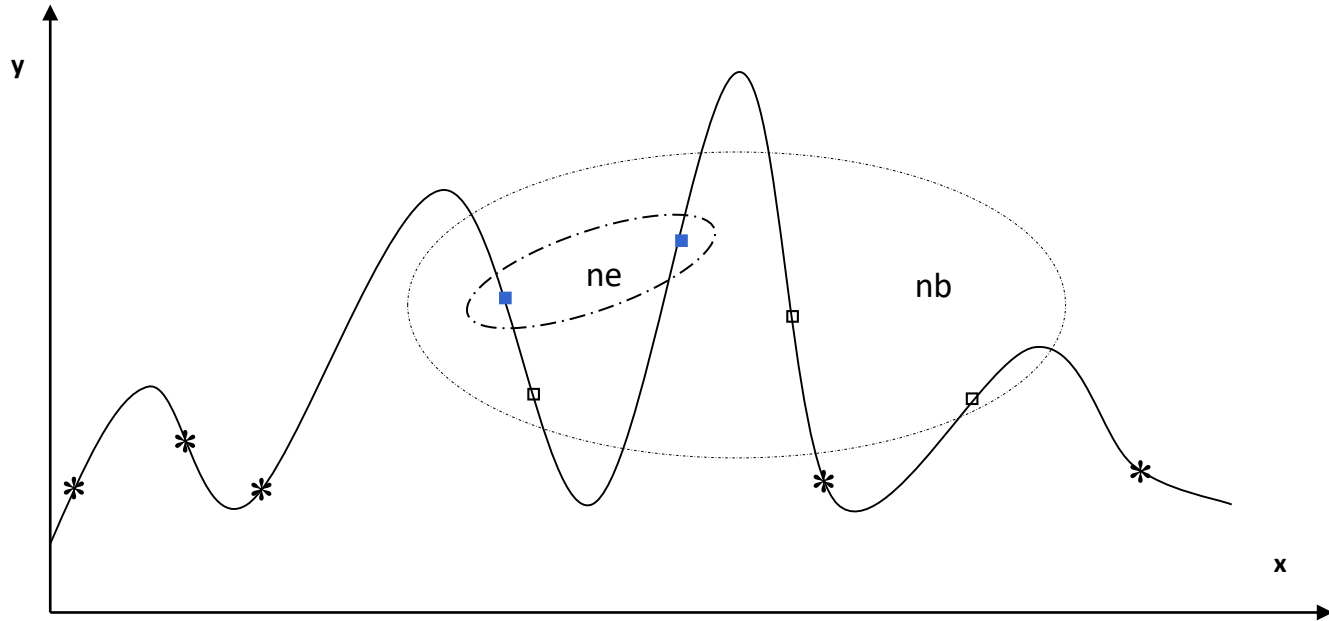


# Example: Function Optimisation



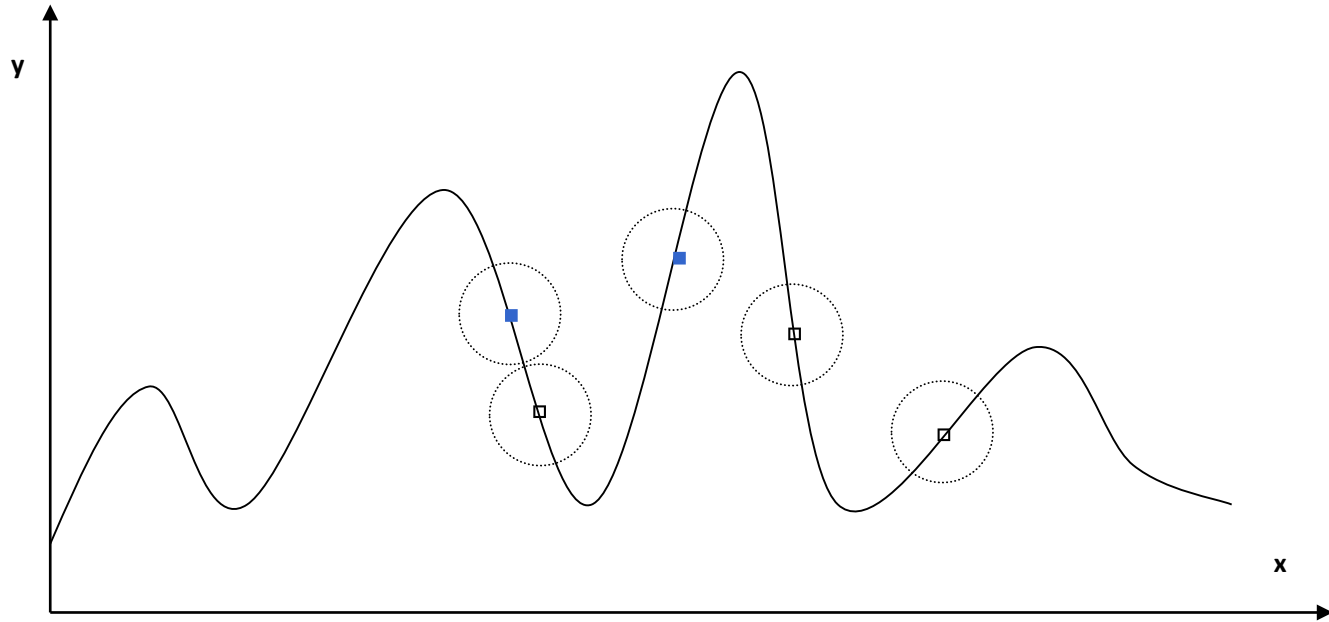
Initialise a Population of ( $ns=10$ ) Scout Bees with random Search and evaluate the fitness.

# Example: Function Optimisation



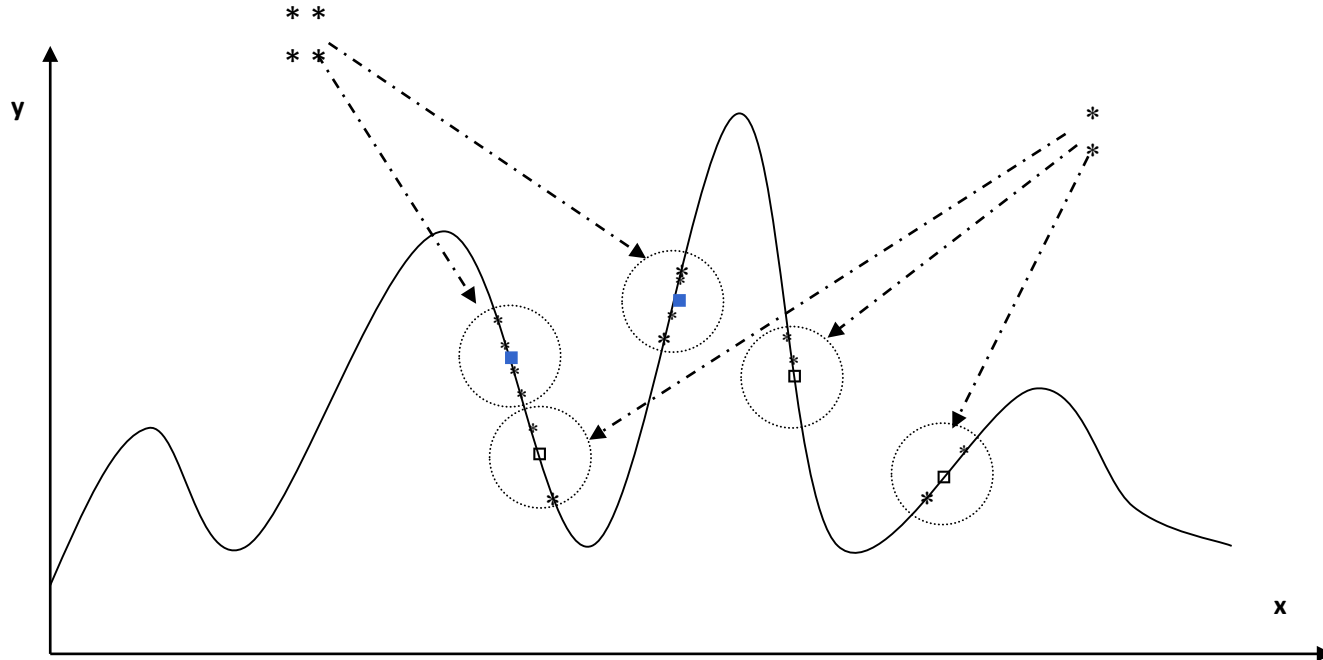
Select best ( $nb=5$ ) Sites for Neighbourhood Search:  
 ( $ne=2$ ) elite bees “■” and ( $nb-ne=3$ ) other selected bees “□”

# Example: Function Optimisation



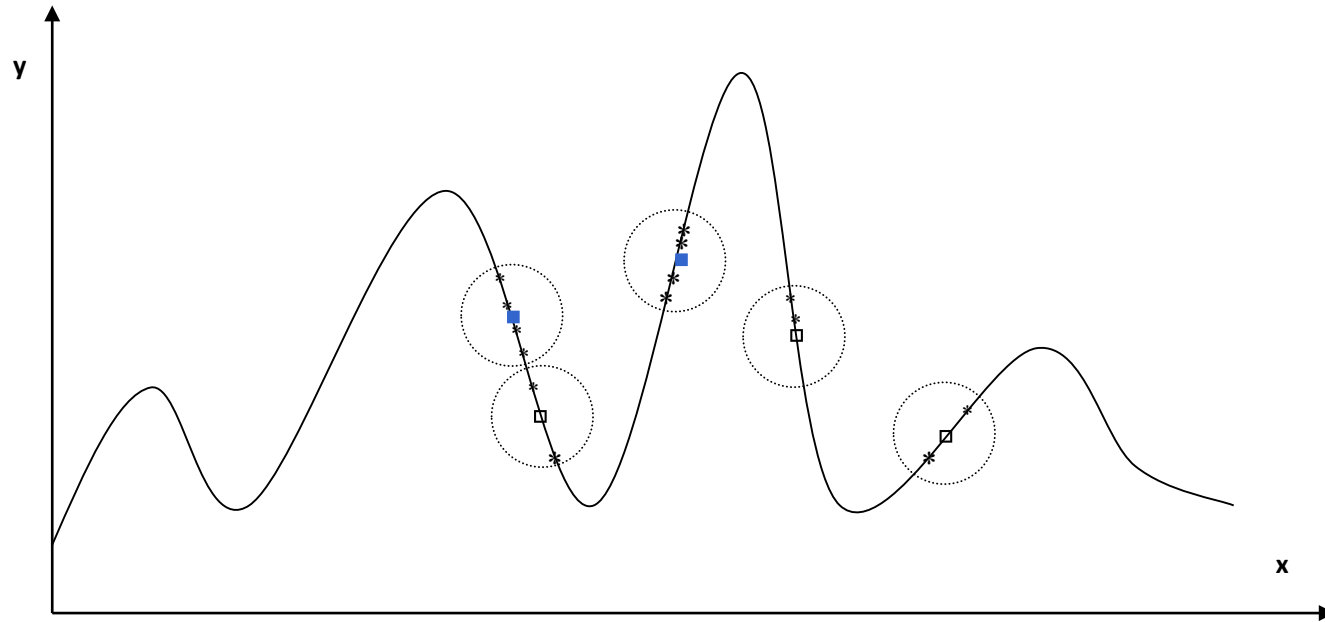
Determine the Size of Neighbourhood (Patch Size *ngh*)

# Example: Function Optimisation



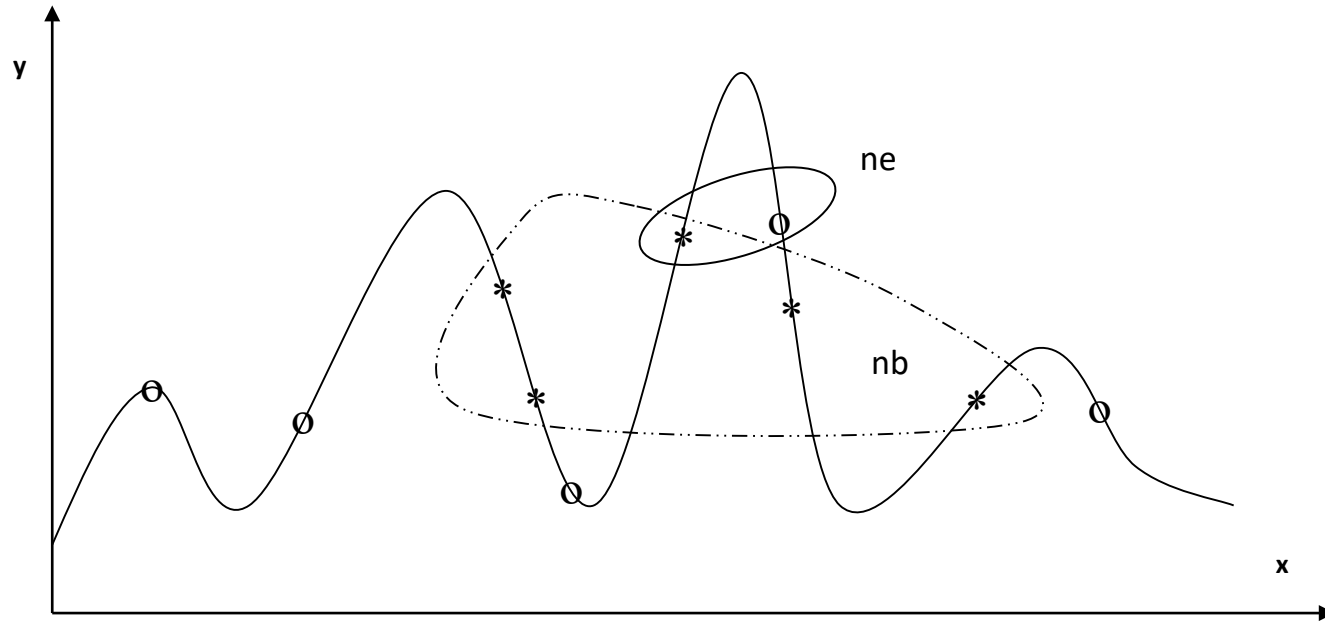
Recruit Bees for Selected Sites  
(more Bees for the  $ne=2$  Elite Sites)

# Example: Function Optimisation



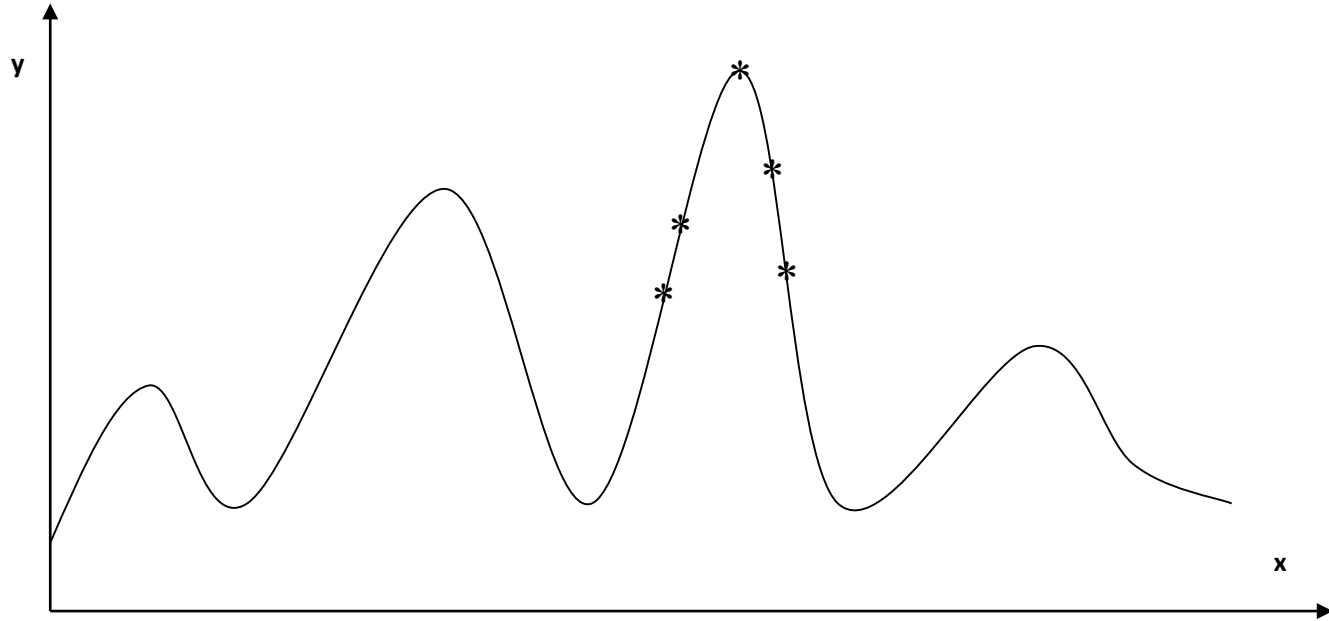
Select the Fittest Bee \* from Each Site

# Example: Function Optimisation



Assign the  $(ns - nb)$  Remaining Bees to Random Search

# Example: Function Optimisation



Graph 7. Find The Global Best point

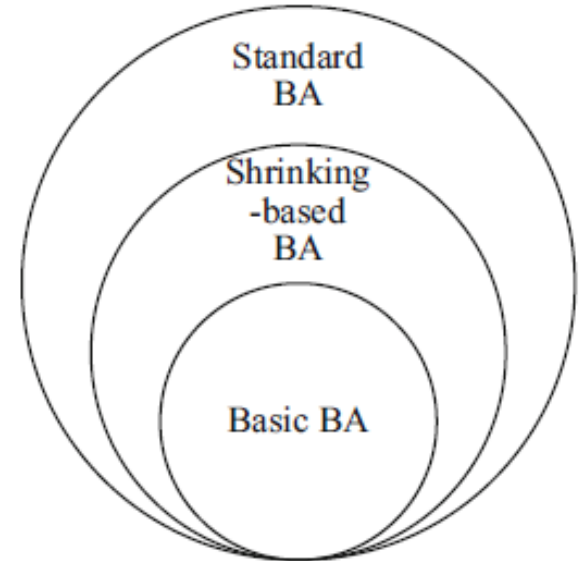
# Variants of BAs

- BA can be divided into four parts:
  1. parameter tuning
  2. Initialization
  3. local search
  4. global search.
- Several works to enhance the performance of BA by improving some of its parts.
- More than one version of the algorithm has been proposed.



# Variants of BAs

- Three main variation (*using several different formal names*):
  1. Basic BA
  2. Shrinking-based BA
  3. Standard BA
- Various implementations of the shrinking and site-abandonment procedures are explored and incorporated into BA to constitute different BA implementations.





## Neighbourhood Shrinking *(in improved version)*

- If no forager improved the fitness of the solution found by the scout, the size of the neighbourhood (flower patch) is shrunk.
- The neighborhood shrinking mechanism aims to focus progressively the search in a narrow area around the fitness peak, and **is akin to the SA**.
- At each cycle of stagnation, the size of the flower patch is customarily decreased using the following heuristic formula:

$$a(t+1)=0.8 \cdot a(t)$$

# Site Abandonment *(in improved version)*

- If the local search procedure fails to bring any fitness improvement in a flower patch for *stlim* consecutive BA cycles,
  - The search is assumed to have found the local fitness optimum.
- In this case, the flower patch is abandoned, and
  - A new scout bee is re-initialized at a random location in the search space.

# BA: Pros and Cons

## Pros:

- Very efficient in finding optimal solutions.
- Overcoming the problem of local optima.

## Cons:

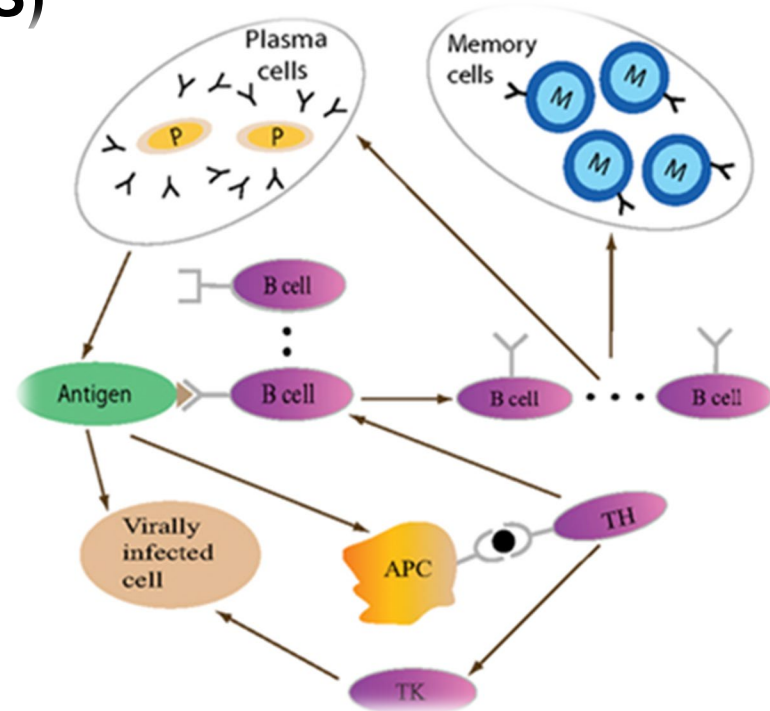
- A number of tunable parameters.

# BA Web Site (Cardiff University, UK)

<http://www.bees-algorithm.com/>

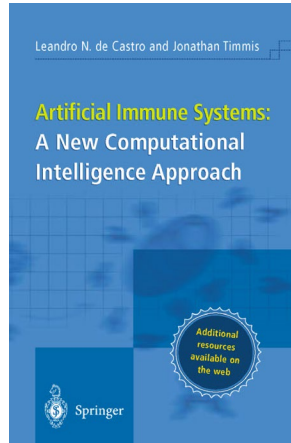
# Outline

- Bees Algorithm (BA)
- **Artificial Immune Systems (AIS)**



# Artificial Immune Systems (AIS)

AIS are intelligent and adaptive systems inspired by the immune system toward real-world problem solving. (**Dasgupta**)



AIS are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving. (**de Castro and Timmis**)

- Relatively new branch of computational intelligence.
- **Not modelling the immune system.**

# History

- Developed from the field of theoretical immunology in the mid 1980's.
- Bersini first use of immune algos to solve problems in 1990.
- Forrest et al – Computer Security mid 1990's.
- Hunt et al, mid 1990's – Machine learning.



# Biological Immune System (BIS)

**Immune system:** a system having the primary function of distinguishing **self from not self** and protects our body from **foreign substances and pathogenic organisms** by producing the immune response.



**Immunity:** state or quality of being resistant (immune), either by virtue of previous exposure (**adaptive immunity**) or as an inherited trait (**innate immunity**).

# Motivation

BIS is a

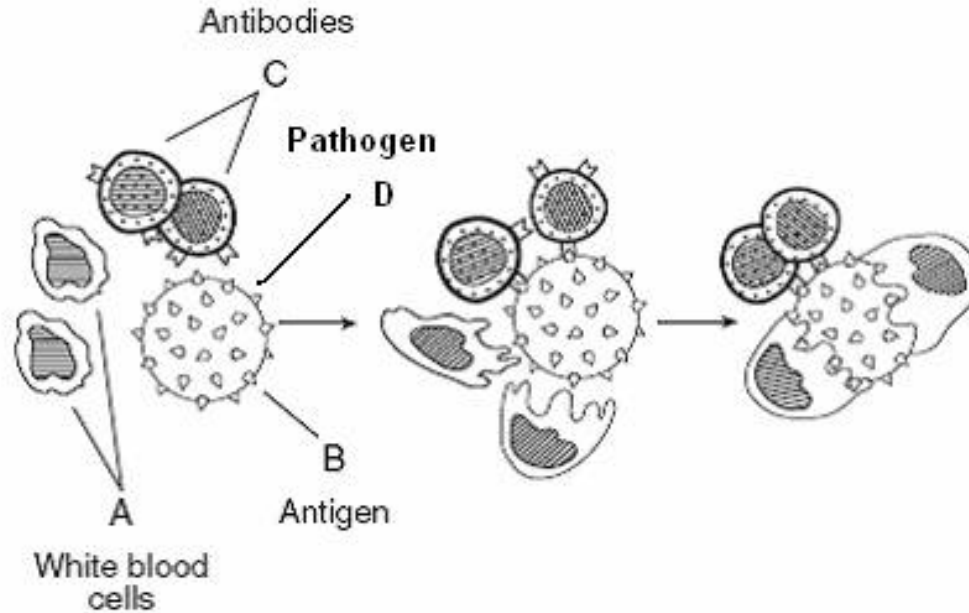
- robust,
- complex,
- highly parallel,
- distributed,
- adaptive

system that defends the body from **foreign pathogens**.

# Motivation

- BIS is a robust, complex, highly parallel, distributed, adaptive system that defends the body from **foreign pathogens**.
- It uses **learning**, **memory**, and **associative retrieval** to solve recognition and classification tasks.
- In particular, it learns
  - to recognize relevant patterns,
  - remember patterns that have been seen previously, and
  - use combinatorics to construct pattern detectors efficiently.

# Pathogen, Antigen, Antibody



# Appealing Features of BIS

- Recognition
  - Anomaly detection
  - Noise tolerance
- Robustness
- Uniqueness
- Autonomy
- Distributed
- Flexible
- Reinforcement learning
- Memory.

# Different Immune Systems

- Primary immune response (**non-specific / innate**)
  - Do not distinguish between one threat and another.
  - Are present at birth.
- Secondary immune response (**specific / adaptive**)
  - Responds to previously unknown foreign cells.
  - Protect against specifically identified threats.
  - Most develop after birth upon exposure to an Antigen.
  - Learning, adaptability, and memory are important characteristics of adaptive immunity (**main focus of interest**).

# White Blood Cells (WBCs)

- WBCs, also called leukocytes or leucocytes, are the cells of the immune system that are involved in protecting the body against pathogens.
- Carry ***antigen*** receptors that are specific.
- There are two primarily types:
  - B-lymphocytes (B cells)
  - T-lymphocytes (T cells)
- Both of these originate in the bone marrow.



# B-cells

- B cells are the detecting antibodies.
- Responsible for the production and secretion of **antibodies**.
- Each B-cell can only produce one particular antibody.
- Two types:
  - Memory Cells
  - Plasma Cells.



# T-cells

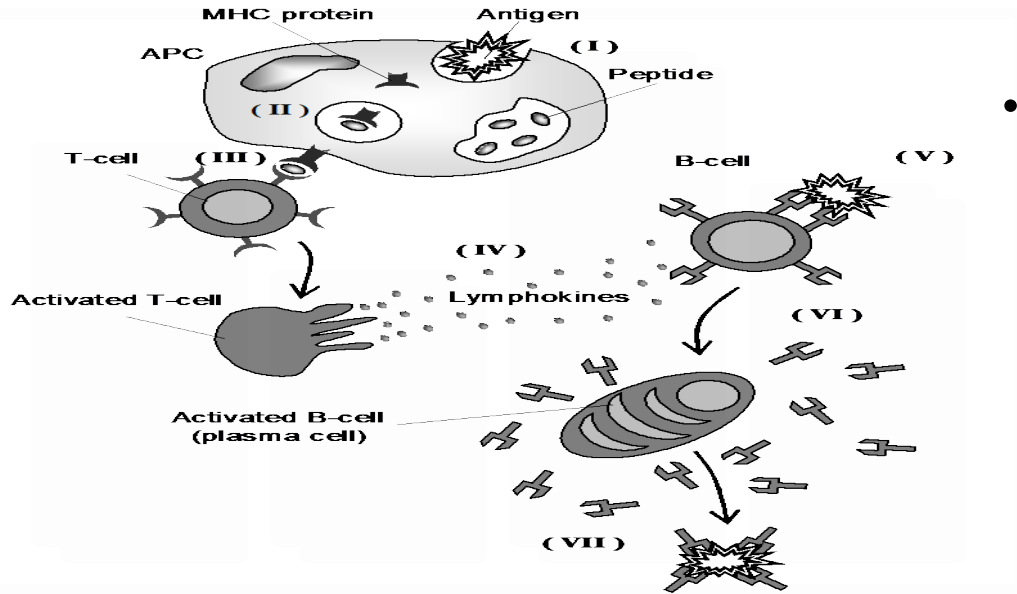
The T-cells are of three types:

- **Helper-T cells:** Essential to the activation cells of immune system (such as B-cells, or Suppressor T-cells). It is some kind of permission giver.
- **Killer T-cells:** Binds to foreign invaders and inject poisonous chemicals into them causing their destruction.
- **Suppressor T-cells:** Prevent action of cells that should never act. Inhibits the action of other immune cells thus preventing allergic reactions and autoimmune diseases.

# B-Cell, Plasma and Memory Cell Animation

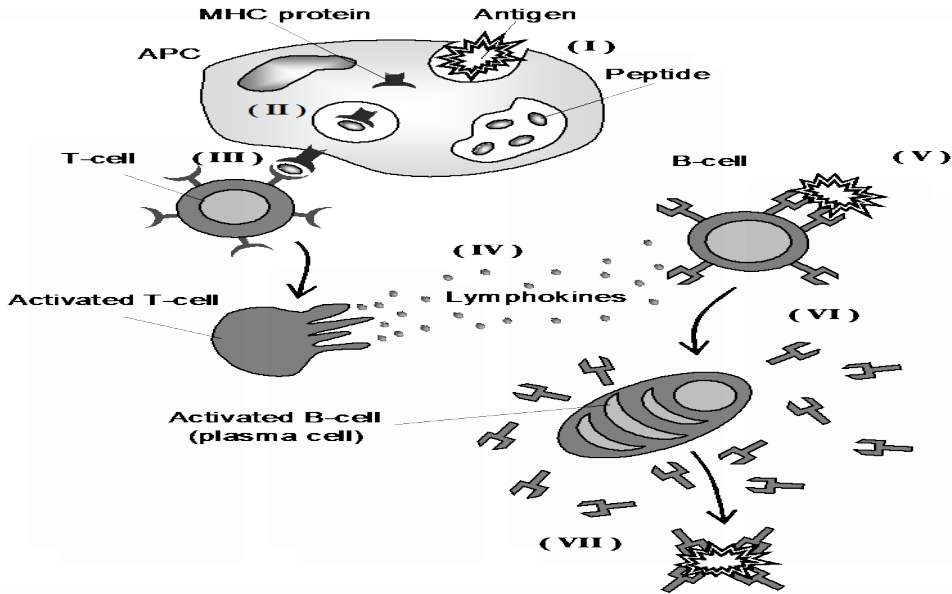


# How Does IS Work



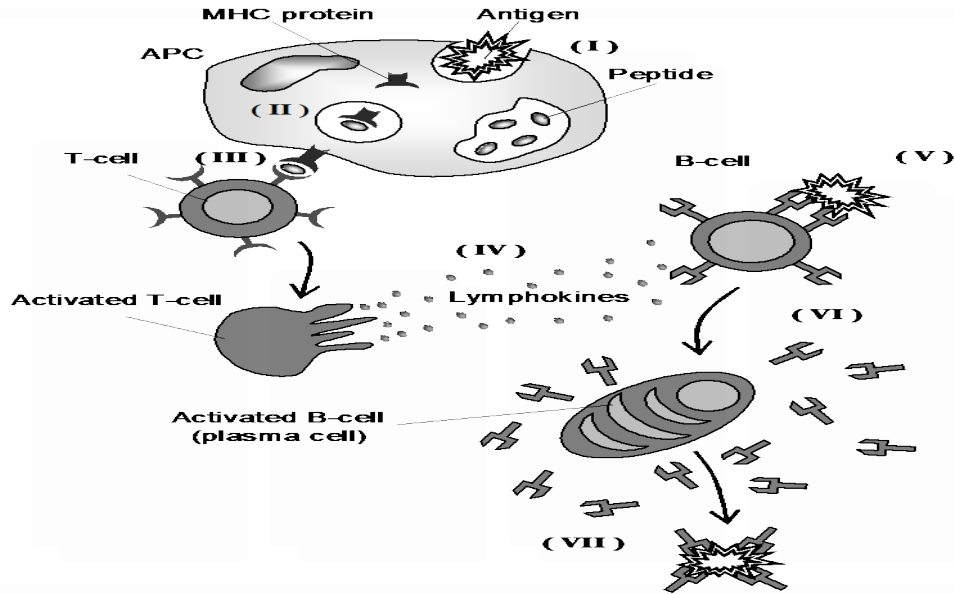
- Antigenes bind to B cells.

# How Does IS Work



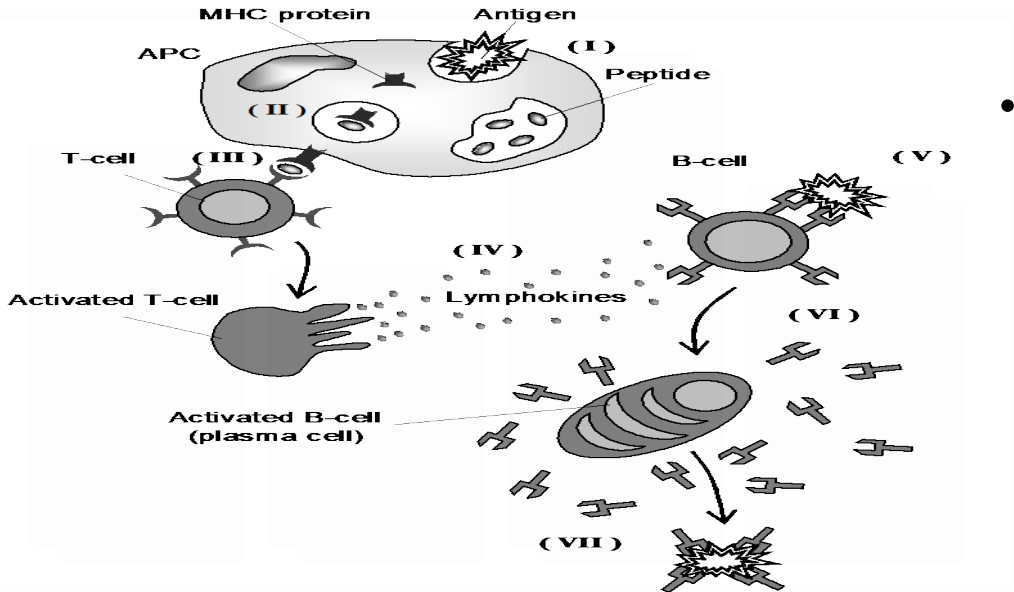
- Helper T cells activates B cells.
  - **Initiate B cell proliferation.**

# How Does IS Work



- B cells proliferate and produce plasma cells.
  - The plasma cells bear antibodies with the identical antigen specificity.
  - The antibodies are released and circulate through the body, binding to antigens.

# How Does IS Work



- B cells produce memory cells.
  - Memory cells provide future immunity.



# Self / Non-Self Recognition

- Immune system needs to be able to differentiate between self and non-self cells.
  - Cells belonging to the body itself are called the “self” cells.
  - External cells are called the “non-self” cells.
- Antigenic encounters may result in cell death, therefore
  - Some kind of ***positive selection***.
  - Some element of ***negative selection***.
- It is important to keep the “self” cells unharmed in the course of exterminating the “non-self” cells.

# Affinity: Define Interaction

- **Affinity: Closeness between Antibody and Antigen.**
- Affinity is related to distance

- Euclidian 
$$D = \sqrt{\sum_{i=1}^L (Ab_i - Ag_i)^2}$$

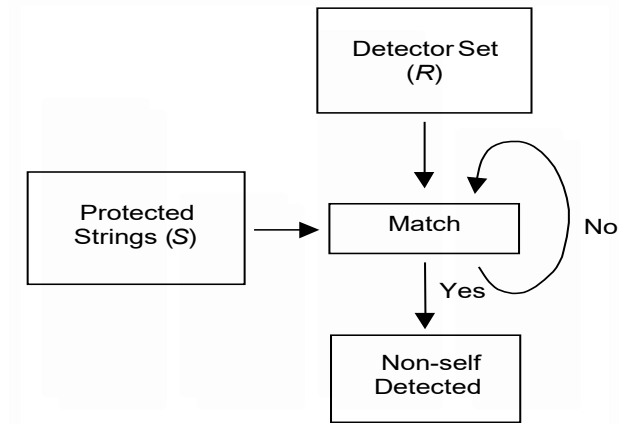
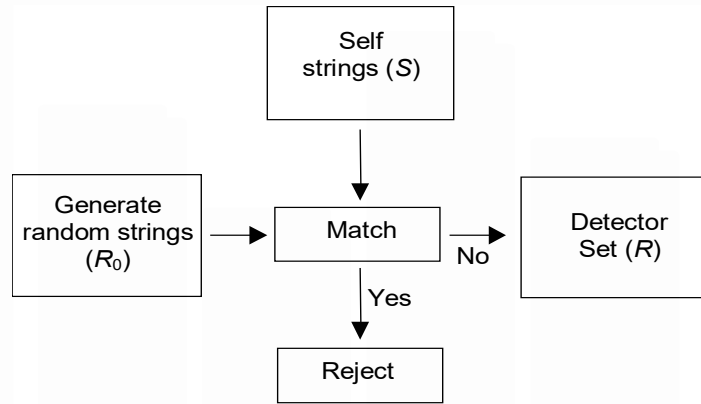
- Other distance measures such as Hamming, Manhattan etc.
- Affinity Threshold



# Basic Immune Models and Algorithms

- Immune Network Models.
- Negative Selection Algorithms.
- Clonal Selection Algorithm.
- Somatic Hypermutation.
- Bone Marrow Models.

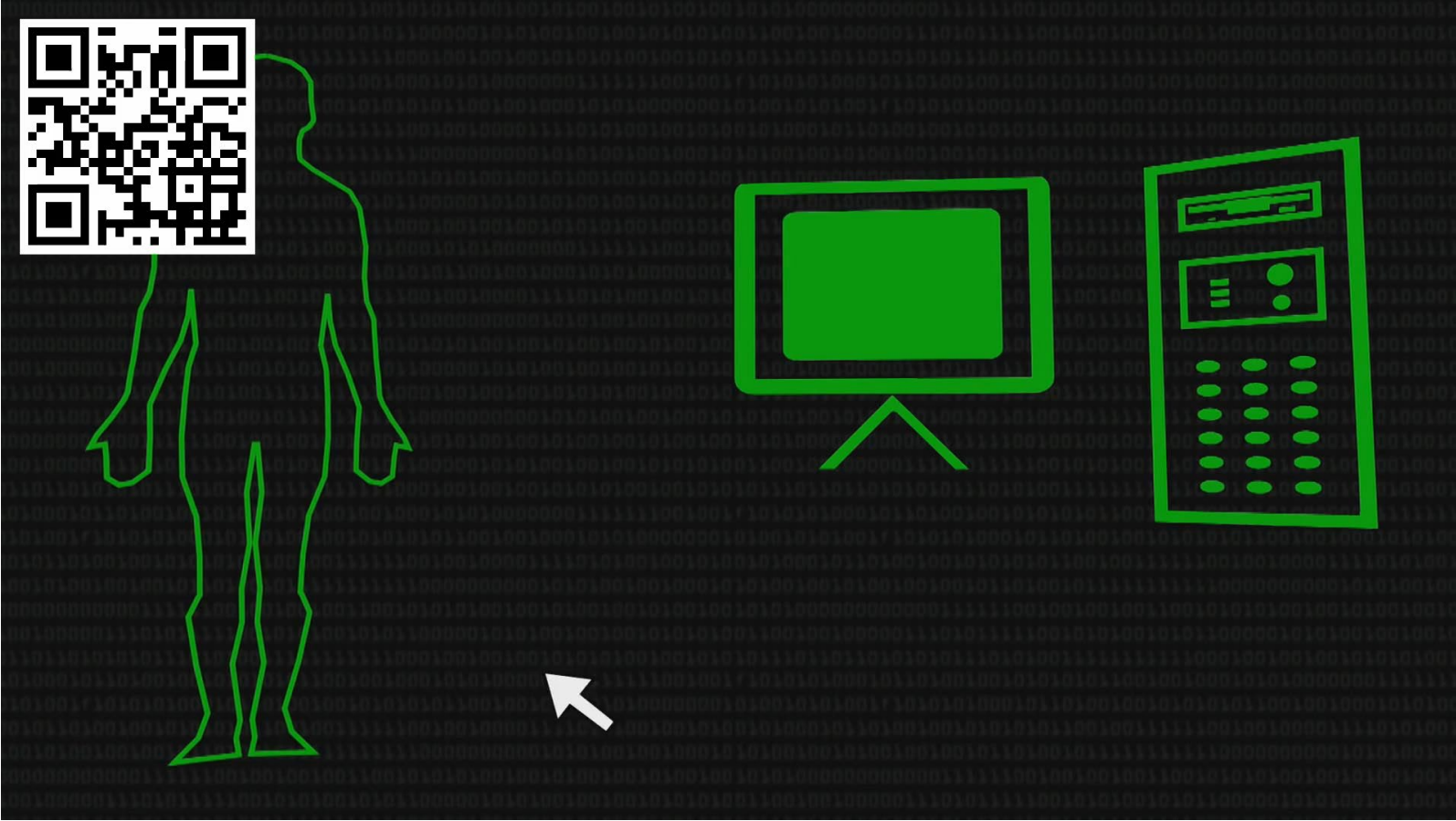
# Negative Selection Algorithms



- Concept of **Self** and **Non-Self**.
- Provides tolerance for self cells.
- It deals with the immune system's ability to detect unknown antigens.
  - while not reacting to the self cells.



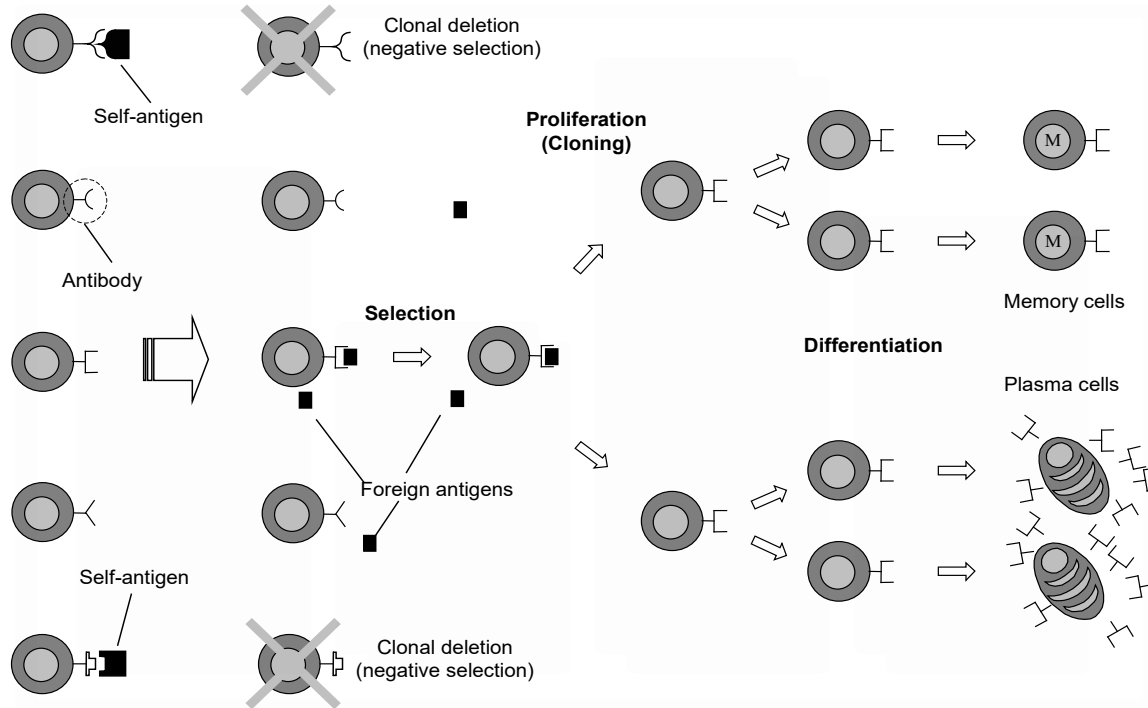
# Negative Selection Algorithms



# Clonal Selection

- It establishes the idea that only those cells that recognize the antigen proliferate.
  - Those who do not recognize are slowly removed.
- Cloning is directly proportional to the fitness.
- Mutation is inversely proportional to fitness.
- Procedure:
  - calculate the fitness.
  - select  $K$  best fit.
  - clone them proportional to their fitness.
  - mutate them inversely proportional to fitness.

# Clonal Selection (*Adaptive IS*)



# Clonal Selection

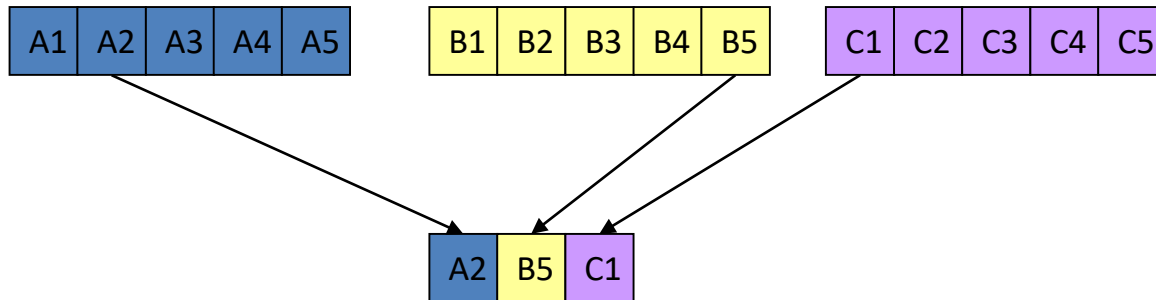
- End result of applying clonal selection is a set of antibodies which are representative of the antigenic data set:
  - Performs data reduction.
  - Used for clustering.
  - Used for classification (e.g new antigens can be presented, highest affinity antibody is indicative of class).

# Somatic Hypermutation

- Mutation rate in proportion to affinity.
- Very controlled mutation in the natural immune system.
- The greater the antibody affinity the smaller its mutation rate.
- **Classic trade-off between exploration and exploitation.**

# Bone Marrow Models

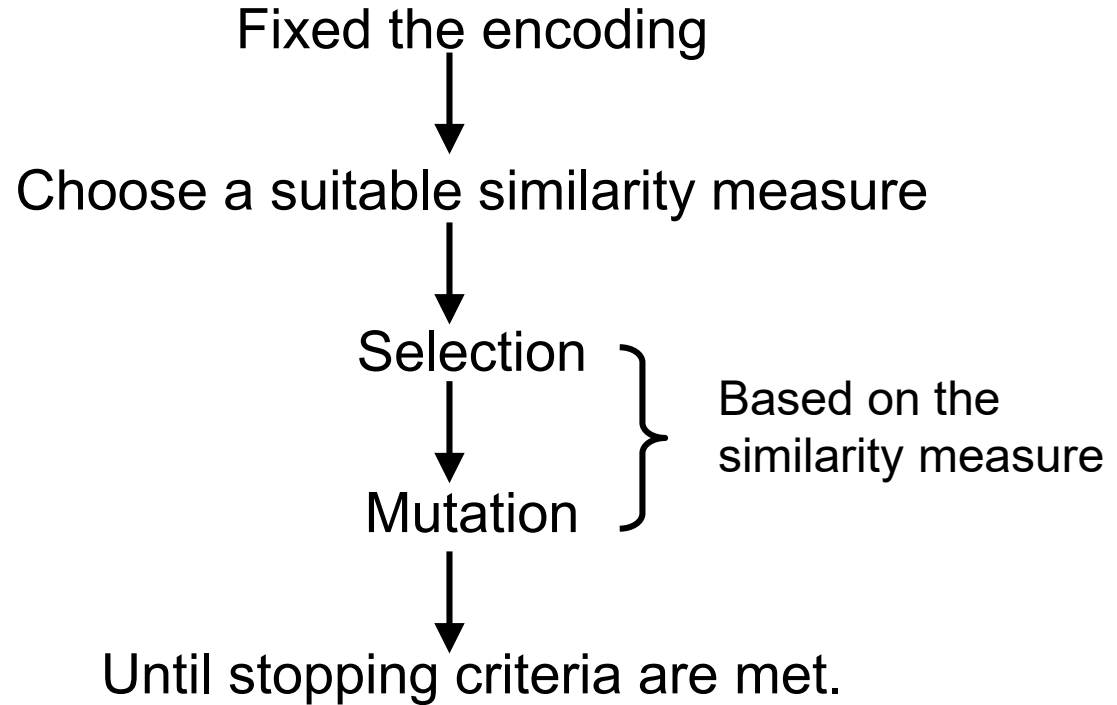
- Use gene libraries to store elements that can be combined to form antibodies.
- Antibody production through a random concatenation from gene libraries.
- Elements in library can be evolved using a genetic algorithm.
- $c$  libraries of length  $l \Rightarrow c^l$  possible antibodies.





# Implementation of A Basic AIS

- Four decisions have to be made:
  - Encoding
  - Similarity Measure
  - Selection
  - Mutation.



# Initialisation / Encoding

- Define ‘antigen’ and ‘antibody’ in the context of an application domain.
- Typically, an antigen is the target or solution.
  - e.g. data item we need to check to see if it is an intrusion (in IDS).
- The antibodies are the remainder of the data.
  - e.g. a set of network traffic that has already been identified (IDS).
- There can be more than one antigen at a time.
- There are usually a large number of antibodies present simultaneously.

# Choice of Representation

- Antigens and antibodies are represented or encoded in the same way.
- For most problems the most obvious representation is a string of numbers or features, where
  1. the length is the number of variables,
  2. the position is the variable identifier and
  3. the value is the value (could be binary or real) of the variable.
- For instance, in a five-variable binary problem, an encoding could look like this: (10010).



# Choice of Representation

- Assume the general case:

$$\mathbf{Ab} = \langle Ab_1, Ab_2, \dots, Ab_L \rangle$$

$$\mathbf{Ag} = \langle Ag_1, Ag_2, \dots, Ag_L \rangle$$

- Binary representation
  - Matching by bits
- Continuous (numeric)
  - Real or Integer, typically Euclidian
- Categorical (nominal)
  - E.g female or male of the attribute Gender. Typically no notion of order

# Representation (Example)

- For the movie recommendation, a possible encoding is:

$$User = \{\{id_1, score_1\}, \{id_2, score_2\}, \dots, \{id_n, score_n\}\}$$

- For intrusion detection, the encoding may be to encapsulate the essence of each data packet transferred,

$$[<protocol> <source ip> <source port> <destination ip> <destination port>]$$

$$[<tcp> <113.112.255.254> <108.200.111.12> <25>]$$

# Choice of Affinity Functions

- Referred to as the “**matching**” or “**fitness function in GA**”.
- The similarity measure or matching rule is one of the most important design choices in developing an AIS algorithm.
- Closely coupled to the encoding scheme.



# Binary Coding

- Consider the pairs of strings:

0 0 0 0 0

0 0 0 1 1

**3**

0 0 0 0 0

0 1 0 1 0

**3**

Similarity matching /  
bit-by-bit comparison

- Quite different as the three matching bits are not connected
- Might be better or worse!!!!**
- Count the number of continuous bits that match and return the length of the longest matching as the similarity measure.
  - First example still be 3; second example it would be 1.

# Not Binary

- If the encoding is non-binary, e.g. real variables, there are even more possibilities to compute the “distance” between the two strings:

- Euclidean  $D = \sqrt{\sum_{i=1}^L (Ab_i - Ag_i)^2}$

- Manhattan  $D = \sum_{i=1}^L |Ab_i - Ag_i|$

- Hamming  $D = \sum_{i=1}^L \delta_i$ , where  $\delta_i = \begin{cases} 1 & \text{if } Ab_i \neq Ag_i \\ 0 & \text{otherwise} \end{cases}$

- For some problems (e.g. data mining), similarity often means “**correlation**”

$$r = \frac{\sum_{i=1}^n (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^n (u_i - \bar{u})^2 \sum_{i=1}^n (v_i - \bar{v})^2}}$$

Pearson correlation coefficient



# Coding Again.....

- For some applications, “**matching**” might not actually be beneficial and hence those items that match might be eliminated.
- This approach is known as “**negative selection**” (discussed already).
- It mirrors what is believed to happen during the maturation of B-cells who have to learn not to “match” our own tissues as otherwise we would be subject to auto-immune diseases.

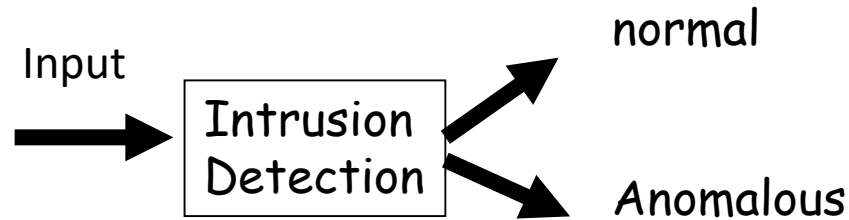
# Negative Selection: Example

- Under what circumstance would a negative selection algorithm be suitable for an artificial immune system implementation?

## Anomaly Detection / Intrusion Detection System

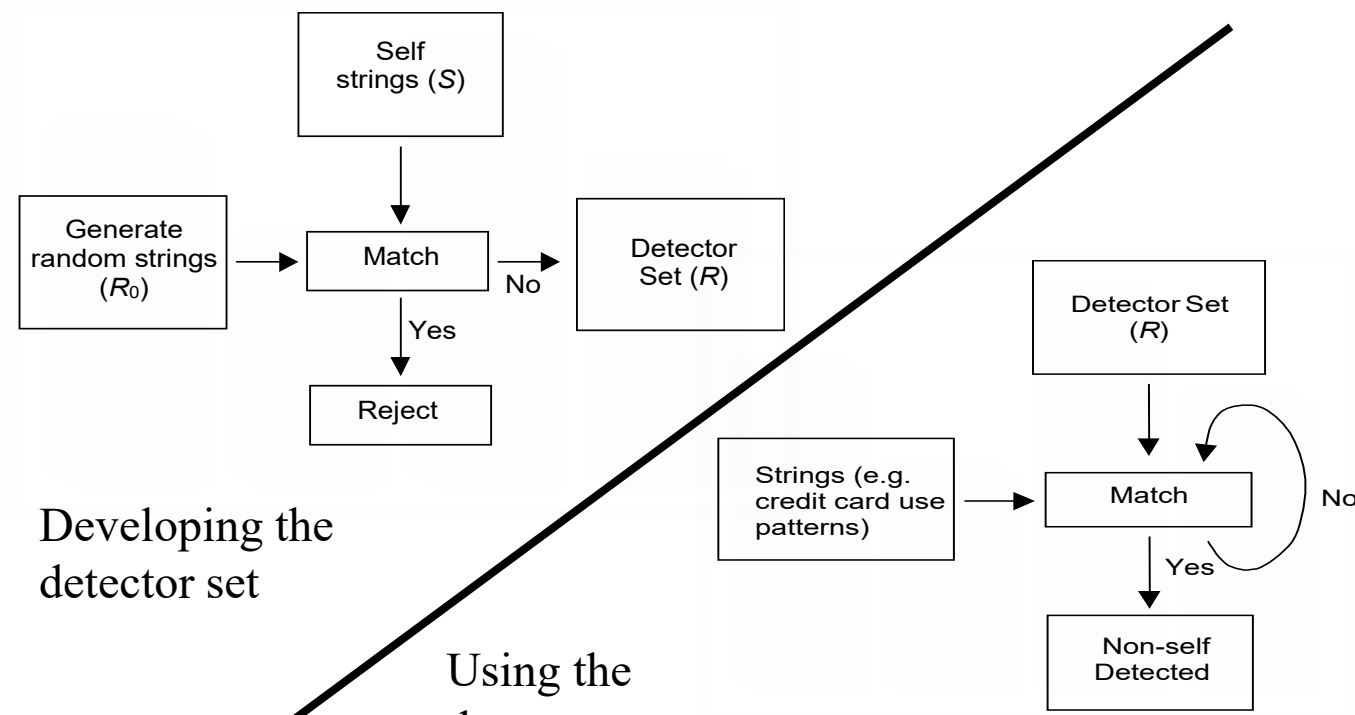
- The normal behavior of a system is often characterized by a series of observations over time.
- The problem of detecting **novelties**, or **anomalies**, can be viewed as finding deviations of a characteristic property in the system. (i.e. **non-self**).

# Intrusion Detection System



**Input can be OS log, network data packet, application system log, firewall log, etc.**

# IDS using Negative Selection Algorithms



# Mutation

- Very similar to that found in Genetic Algorithms.
  - for binary strings bits are flipped.
  - for real value strings one value is changed at random.
  - for others the order of elements is swapped.
- The mechanism is often enhanced by the '**somatic**' idea,
  - The closer the match (or the less close the match, depending on what we are trying to achieve),
    - the more (or less) disruptive the mutation.



# Typical Applications of AIS

- Computer Security (*Forrest'94'98, Kephart'94, Lamont'98,02, Dasgupta'99'01, Bentley'01,02*)
- Anomaly Detection (*Dasgupta'96'01'02*)
- Fault Diagnosis (*Ishida'92'93, Ishiguro'94*)
- Data Mining & Retrieval (*Hunt'95'96, Timmis'99'01, '02*)
- Pattern Recognition (*Forrest'93, Gibert'94, de Castro '02*)
- Adaptive Control (*Bersini'91*)
- Job shop Scheduling (*Hart'98, '01, '02*)
- Chemical Pattern Recognition (*Dasgupta'99*)
- Robotics (*Ishiguro'96'97, Singh'01*)
- Optimization (*DeCastro'99, Endo'98, de Castro '02*)
- Web Mining (*Nasaroui'02, Secker'05*)
- Fault Tolerance (*Tyrrell, '01, '02, Timmis '02*)
- Autonomous Systems (*Varela'92, Ishiguro'96*)
- Engineering Design Optimization (*Hajela'96 '98, Nunes'00*)

# Suggested Reading

- Yuce, B., Packianather, M. S., Mastrocinque, E., Pham, D. T., & Lambiase, A. (2013). Honey bees inspired optimization method: the bees algorithm. *Insects*, 4(4), 646-662.
- Aickelin, U., Dasgupta, D., & Gu, F. (2014). Artificial immune systems. In *Search methodologies* (pp. 187-211). Springer, Boston, MA.