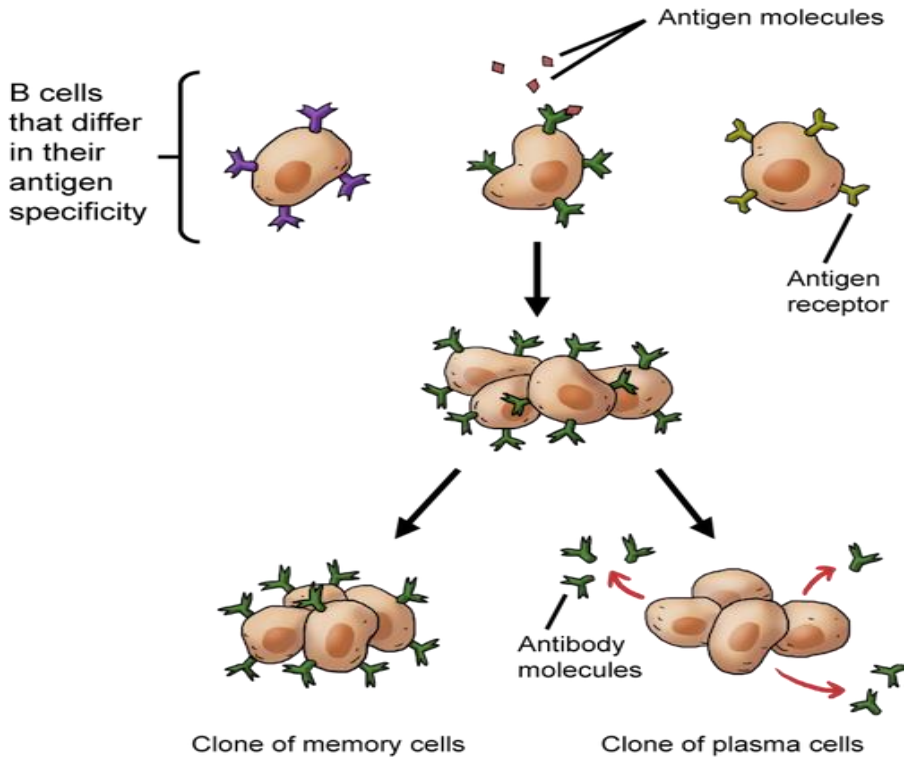
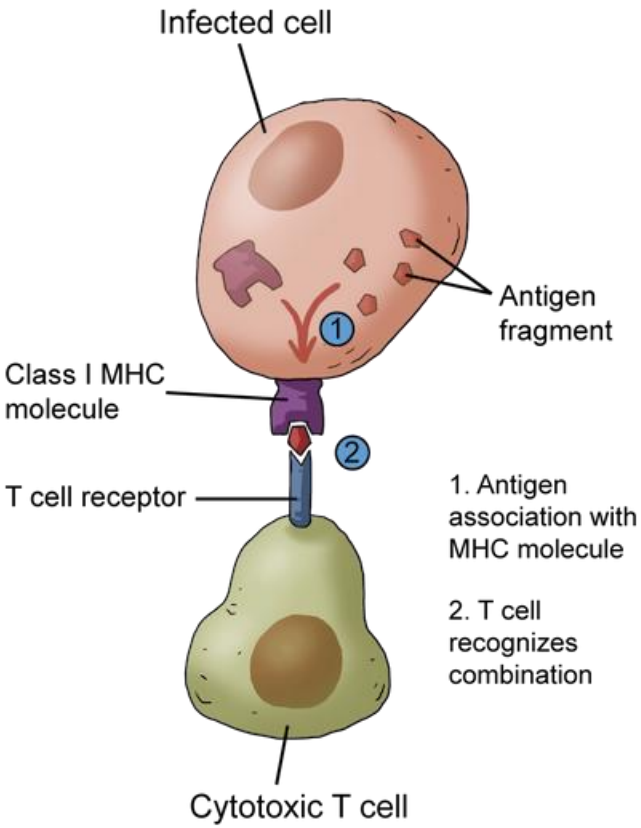
Artificial Immune system

Artificial Immune System (AIS)

- Identify the foreign materials using pattern recognition and produce relevant antibodies for neutralizing the effect.
- Self-strengthening of the immune system is a crucial property for the prevention of diseases.
- Each time through learning the immune system increases its durability.
- The leaning strength and memorizing property of immune system processes can be made use of in solving optimization problems.

Immune System (IS)





major histo-compatibility complex (**MHC**)

Helper T cells

Cytotoxic T cells

Suppressor T cells.

http://www.shmoop.com/animal-systems/immune-system.html

Artificial Immune System (AIS)

- The adaptive immune system uses somatically generated antigen receptors which are clonally distributed on the two types of lymphocytes:

- **B (**bursa of Fabricius**) cells and T(**thymus**) cells**.

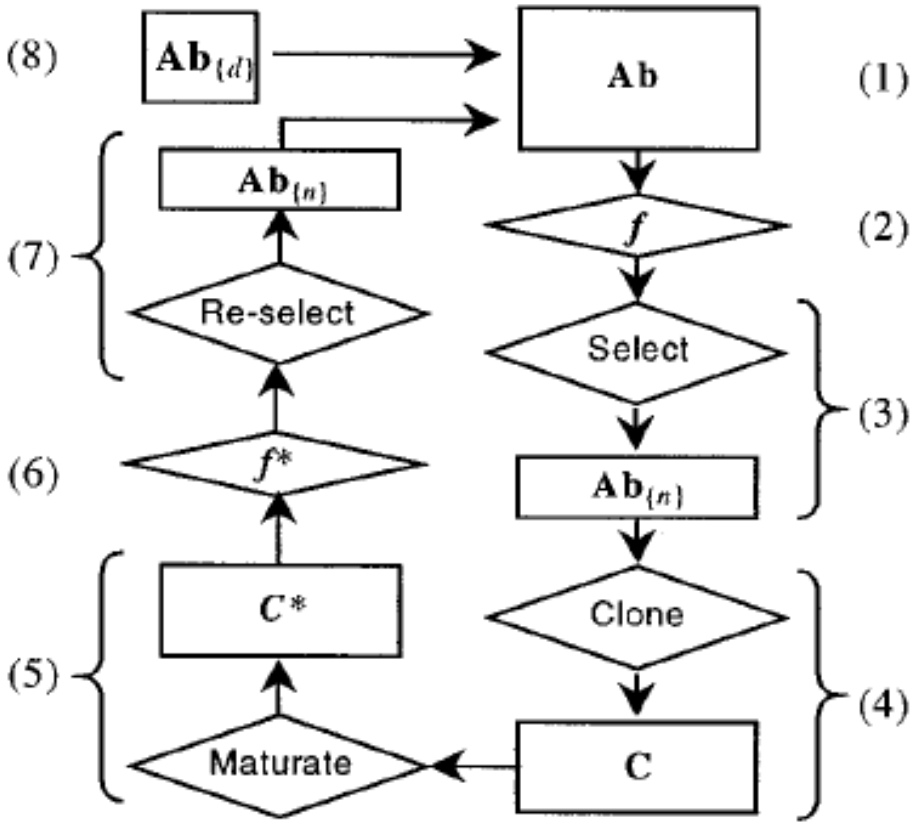
- These antigen receptors are generated by random processes and, as a consequence, the general design of the adaptive immune response is based upon the **clonal selection** of lymphocytes expressing receptors with particular specificities (Burnet, 1959-1978).

Clonal selection

- Clonal selection is a popular theory proposed by Burnet (1959) [35].
- Clonal selection theory is a scientific theory in immunology that explains the functions of cells (lymphocytes) of the immune system in response to specific antigens invading the body.

- Clonal selection is used to explain the processing of adaptive immune system to antigens.

Clonal selection



Computational procedure for CLONALG: optimization version.

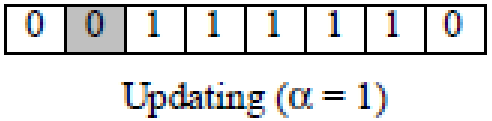
The number of clones generated for all these selected antibodies

$$N_c = \sum_{i=1}^{N} \text{round}(\beta \cdot N).$$

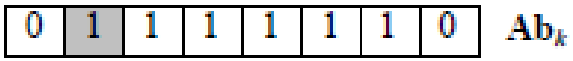
Where *Nc* is the total number of clones generated for each of the Ab's, β is a multiplying factor, *N* is the total number of Ab's, and is the operator that rounds its argument toward the closest integer.

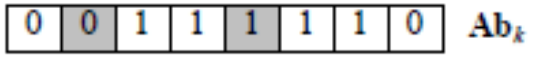
Weights Updating
affinity maturation

- affinity maturation
  - Hyper mutation mechanism
    - Mutation mechanism with high rates
- The hyper mutation rate α determines how many points of the bit string will be mutated.



Updating ($\alpha = 1$)



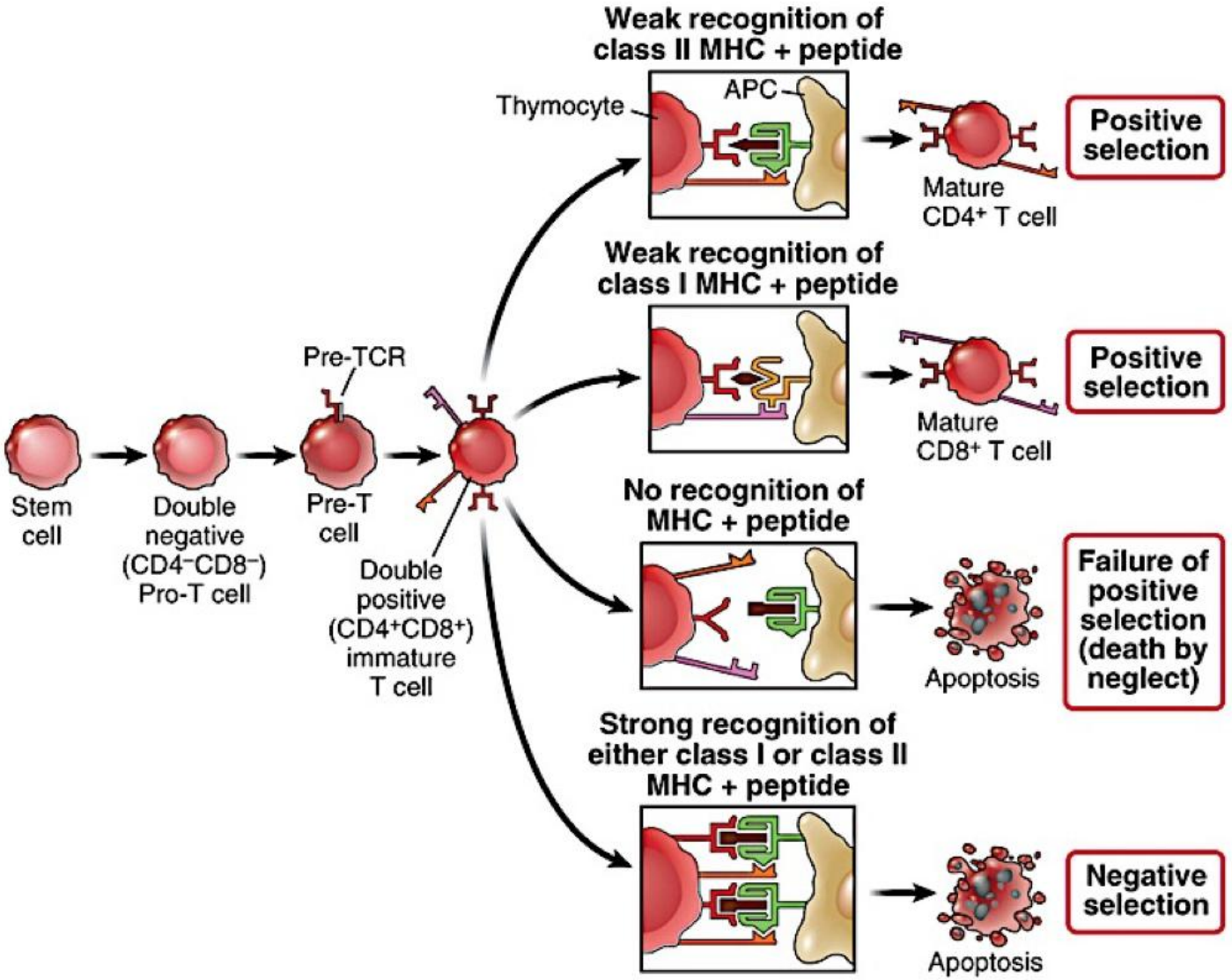


Updating ($\alpha = 2$)

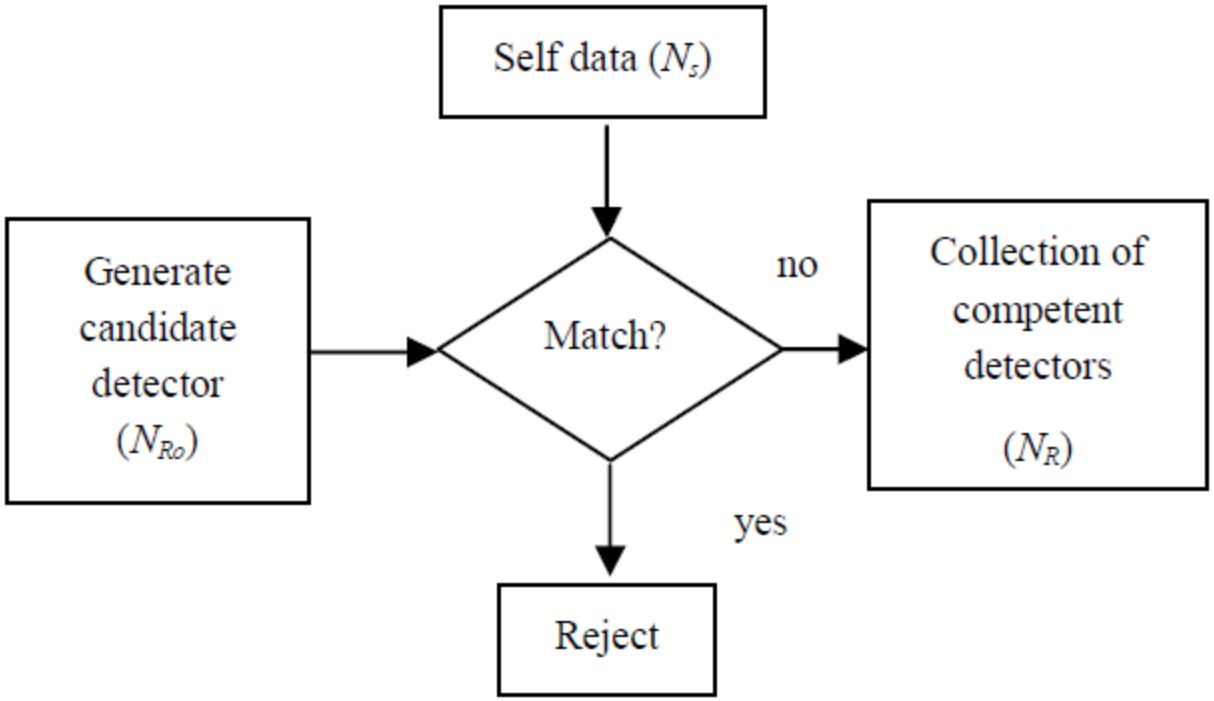
Clonal selection algorithm

```
1.      /* Initialize generation number as zero*/
        Generation := 0
2.      /* Initialize the first generation population*/
        Ab_Pop(Generation) := Init(Clonal Ab_pop(P))
3.      /* Fitness evaluation of the current population */
        Ag=Evaluate_Fitness (Ab_Pop(Generation))
4.      while termination criteria not met do
  4.1       /*Select n individuals from the population pool*/
                Selected_Ab_Pop(Generation):=Selection(Ab_Pop(Generation))
  4.2       /*generate Cloned population from the selected n individuals*/
                Cloned_Ab_Pop(Generation):=Clone(Selected_Ab_Pop(Generation)
  4.3       /*Maturate cloned population and merge with the population pool*/
                Mature(Generation):=Maturation(Cloned_Ab_pop(Generation))
  4.4       /*Randomly generate fresh individuals*/
                Rand(Generation):=Random()
  4.5       /*Merge and update current population*/
                Pop(Generation)=Merge(Ab_Pop(Generation),Mature(Generation), Rand(Generation))
  4.6       /* Fitness evaluation of the current population*/
                Evaluate_Fitness (Ab_Pop(Generation)
  4.7       /* Select individuals from the population pool for the next generation*/
                Pop(Generation+1):=Re_Selection(Ab_Pop(Generation))
                Generation := Generation + 1
5.      end while
```

Artificial immune system- negative selection



Artificial immune system- negative selection



S. Forrest, A. Perelson, et al. (1994).

Artificial immune system- negative selection

Immune system	Optimization problem
Pathogen	Problem (environment of antigens) (e.g., city graph wherein nodes represent antigens)
Immune response	Solution (e.g., shortest path)
B-cells	Agents
Clonal selection	Creating new agents in order to explore the environment (i.e., proliferation)
Positive/negative selection	Selection of useless/bad agents to kill themselves (i.e., apoptosis)

Application to TSP

- **Nodes** represents **antigen**.
- **B-cells are agents** that progress from a city to neighboring cities and can clone or destroy themselves based on **positive/negative selection** criteria.

Application to TSP

1. The algorithm starts with an initial agent at the source city.
2. At each algorithm cycle, an agent could clone itself and the newly spawned clone moves to neighboring cities.
3. When an agent reaches a city that belongs to its already visited cities set, the positive selection rule is triggered and the agent kills it (i.e., useless solution).
4. Otherwise, the agent clones it and the clone acquires a copy of the already visited cities set from its parent.
5. When all survival agents have accomplished their tour (i.e., reach the source city), the negative selection rule is triggered and among these B-cell agents that constitute the immune response, the one that held the best tour is selected (i.e., useless agents are destroyed).

Application to TSP

```
Initialization
Create a mobile agent A
A.citiesList= Cities // the set of cities
A.souceCity = Random(Cities) // agent is positioned on a starting city
A.visitedList = {} // the set of visited cities
A.mAffinity = dist // maximal affinity generated at random
A.currentCity =Null // the city in which the agent is positioned
A.LastCity = A.currentCity // the city lastly visited
A.cAffinity=0 // current affinity of the actually tour
//Agent terminates if all cities are visited
while (A.CitiesList ≠ Null) do
  A.cAffinity= A.cAffinity+ δ(LastCity, currentCity)
   if(A.currentCity ∉ A.visitedList and A.cAffinity < A. mAffinity)
      A.visitedList.Add(currrentCity)
      A. citiesList.Remove(currrentCity)
         // the agent clones itself and moves with its clone
            A.LastCity= currentCity
            B=A.clone() // if there at least two neighbors
```

Application to TSP

```
            // n1 is selected at random from neighbors such as n1 ∉ visitedList
A.currrentCity = n1; B.move()
// n2 selected at random from neighbors such as n2 ∉ visitedList-{n1}
B.currrentCity = n2 ; A.move()
        endFor
  else
   // positive selection, the agent not make a tour and kills itself
       A.die()  //useless solution
   endif
done
// negative selection, agent die itself if an other agent that have a better tour
   A.die() // bad solution
```

Thank you