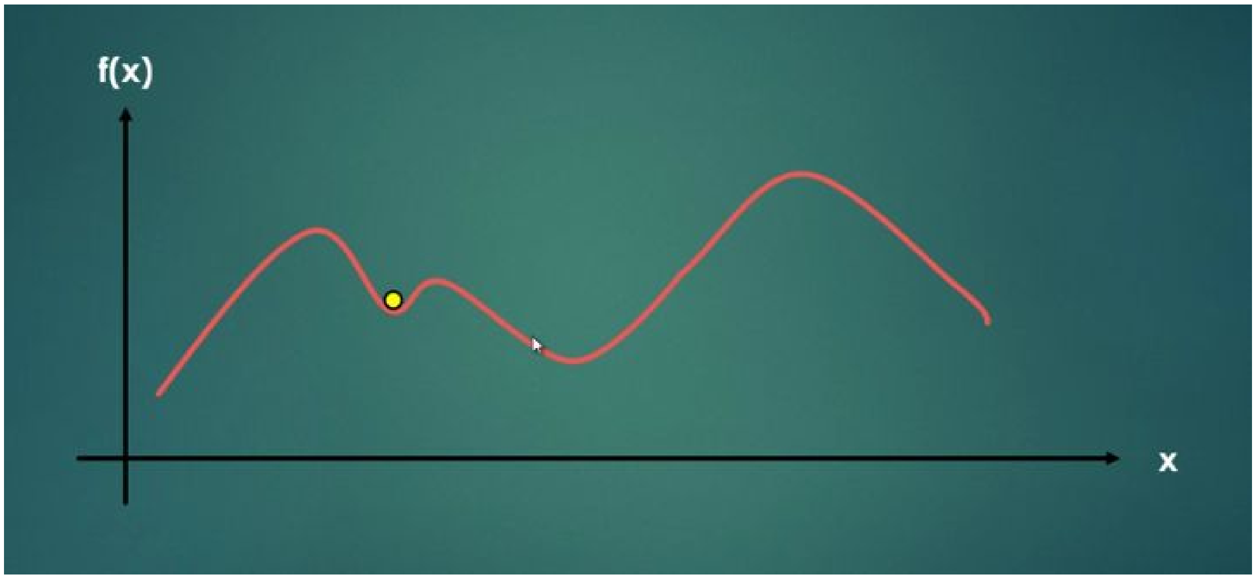
Artificial Intelligence - 2017
Meta - Heuristic Algorithms

Prepared By;
Vijani Supeshala Piyawardana | BSc (Hons) Computer Science |
Demonstrator - COMP3004L, COMP3008L - UCD |
NSBM Green University Town

- Simulated annealing - consequences of changing the cooling schedule
- Graph partitioning applications
- Crossover / mutation operators definition in genetic algorithm
- Graph partitioning problem in binary string

Simulated Annealing

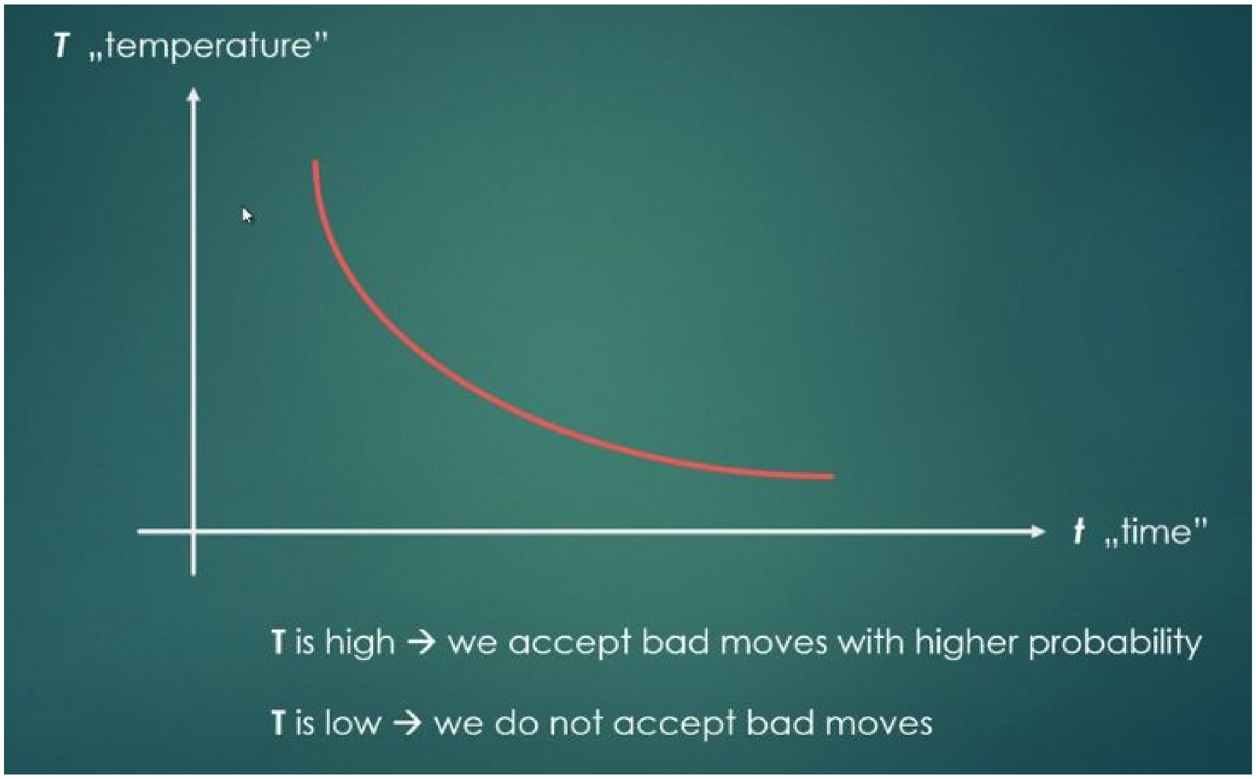
Helps to avoid LOCAL OPTIMUM



Genetic algorithm and simulated annealing has to make sure that we're going to end up with the global optimum.

Simulated Annealing mimic the annealing process to solve an optimization problem.
Uses a temperature parameter that controls the search.

- The temperature parameter typically starts off high and is slowly “cooled” or lowered in every iteration.
- At each iteration, a new point is generated (usually at random) and its distance from the current point is proportional to the temperature.
- If the new point has a better function value, it replaces the current point and iteration counter is incremented. It is possible to accept and move forward with a worse point.
- The probability of doing so (moving forward with a worse point) is directly dependent on the temperature. This unintuitive step sometimes help identify a new search region in hope of finding a better minimum.



T temperature is going to decrease with the time. We start from beginning T is very high , and at the end of the algorithm T is very low. T is important, it will help us to determine which solution must be accepted.

Acceptance function

We check whether the neighbour solution is better than our current solution or not.

If it is ---> we accept it
If the neighbour state is not better ---> we may accept it.
The temperature is going to determine it.
High T means we accept bad moves more often.

How to calculate the probability to accept or not

Exponential function (e)

(actualStateEnergy - neighbourStateEnergy) / temperature

According to this, if T is high , the probability is going to be lower.

Algorithm

1. We set the initial temperature + create a random initial solution.
2. We iterate until a stop condition is met :
    a. Ex: the system has sufficiently cooled or an approximated solution has been found.
3. We generate a neighbour :
   Ex: make some little change to the current state
4. We decide whether to move to that neighbour state with the help of the acceptance function.
5. Decreases the temperature + looping (repeat from step 2)

Genetic Algorithm

https://www.youtube.com/watch?v=ejxfTy4lI6I

- A search heuristic
- Mimics the process of natural selection.
- Used to generate useful solutions to optimization and search problems.
- Quite similar to Simulated Annealing.

- Generate solutions to optimization problems using techniques inspired by natural evolution, such as mutation or crossover.
    - A candidate solution is evolved toward better and better solutions.
    - Each candidate solution has a set of properties (genes) which can be mutated and altered.
    - Usually solutions are represented in binary.

Algorithm

- An evolution usually starts from a population of randomly generated individuals. (Similar to Simulated Annealing, because SA starts with a random initialization)

- It is an iterative process -> with the population, in each iteration called a generation.
- In each generation, the fitness of every individual in the population is evaluated, according to a fitness function.
- Fitness is usually the value of the objective function (fitness function) in the optimization problem being solved.

We are going to have several iterations, in every single iteration we are going to make a little changes to the actual solution, these are called mutations or crossovers. And on each iteration, we calculate the actual best solution, which is going to be the best solution for the next iteration. We keep mutating, doing crossover again, keep considering better solutions. If there are better solutions, we keep iterating until we bump into a situation where we found the solution we are looking for.

The more fit individuals are selected from the current population and each individual's gnome is modified (recombined and possibly randomly mutated) to form a new generation.
So we keep constructing generations from generations over and over again.

The new generations of candidate solutions is then used in the next iteration.
Commonly, the algorithm terminates when either a maximum no of generations has been produced, or satisfactory fitness level has been reached for the population.

Chromosome Representation

To use genetic algorithm, first we have to encode any potential solution.
Ex: string of real numbers or more often a binary bit stream / binary string.

1000110110101110110000001

At the beginning of the algorithm, a large population of random chromosomes is created.

Each chromosome is represented a different solution.
The fitness function tells us how good that chromosome is.

Cloning

Cloning is the copying of the same chromosome all the time by maintaining the same appearance of same characteristics. This is the most frequent genetic operation.

0011 → 0011

Crossover

0100110111
1101100101

Operation on two chromosomes, values in the genes can be swapped.
Binary representation of the solutions.
Crossover, sometimes we generate a random index, and after the given index, we just swap the two subarrays of bits.
This is how we generate new possible solutions.

010011 | 0111
110110 | 0101

010011 | 0111
110110 | 0101

010011 | 0111
110110 | 0101

If the crossover threshold is low ---> we will make few changes, the algorithm will not be more accurate, but it will be faster.
If the crossover threshold is high ---> we will make a lot of changes with high probability, algorithm gets slower, but more accurate.

Mutation

An operation on single chromosome.

Mutation threshold : 0.5
We generate a random number for each bit: 0.8
If the random number is greater than the threshold, we do not mutate the given chromosome bit.

Mutation threshold : 0.5
Given chromosome 0101110111

First bit , random number : 0.8 , 0.8>0.5 do not mutate	0101110111
2nd bit , random number : 0.55 , 0.55>0.5 do not mutate	0101110111
3rd bit , random number : 0.43 , 0.43<0.5 mutate	0101110111
4th bit , random number : 0.11 , 0.11<0.5 mutate	0100110111

And so on.

So mutation is : we iterate through the genes and we mutate if necessary. (flip the bits to opposite)
Values of the gene in a chromosome can be inverted by giving different characteristics of a person.

Difference between crossover and mutation

They perform 2 different roles.

Crossover is an operation which drive the population towards a local maximum/minimum. If we use only crossover it will yield approximately the same result as hill-climbing algorithm.

Mutation is so-called divergence operation. Force one or more member of the population to discover other regions of the search space. So it is essential in order to find the global optimum.

Genetic algorithms can be used for modelling dynamically changing situation (mutation) or generating solution with some features from already known solutions (crossover)

Ex : Toyota cars generations
RedLady papaya → some genes of pig skin + papaya, to create a thick skin on papaya, so transport is safe.

Past Exam Questions

2014 AI Exam (Batch 4)
Question 3

Simulated Annealing
(c)What is meant to be the *cooling schedule* of a simulated annealing algorithm?

Genetic Algorithm

(d)The 0-1 *knapsack problem*, is as follows; Given a set of n kinds of items, x_1,......x_n, and a bag. Each item x_j has a value p_j and a weight w_j. The goal is to fill the bag with the maximum weight C, such that *at most* one item of each kind is placed in the bag and the total value of items contained in the bag is maximized.
Propose a *genetic algorithm* to solve the 0-1 *knapsack problem* by specifying;

I. the representation of solutions
Ii. a crossover operator
Iii. a mutation operator

Explain how you would ensure that the output of the crossover and mutation operators represents a feasible solution.

2015 AI Exam (Batch 3)
Question 3

<u>Simulated Annealing</u>
(a) Describe how the simulated annealing (SA) algorithm can be used to solve optimisation problems. Your answer should refer to:

i. the Boltzmann probability distribution;
ii. the relevance of the temperature parameter;
iii. markov chains and quasi-equilibrium;
iv. the cooling schedule.

<u>Genetic Algorithm</u>
(b) Let G = (V, E) be an undirected graph with a vertex set V and an edge set E. A bipartitioning of G is a decomposition of the vertices of G into two disjoint subsets P0 ∈ V and P1 ∈ V such that P0 ∪ P1 = V and P0 and P1 contain equal numbers of vertices. An edge e = (v, w) ∈ E is said to be cut by the partitioning if v and w are assigned to different partitions. The Graph Partitioning Problem seeks to find a bi-partitioning of a graph that minimises the number of cut edges. Propose a genetic algorithm to solve the graph partitioning problem. In particular, propose

i. A solution representation (genome);
ii. A method to calculate the fitness of each genome;
iii. Cross-over and mutation operators.

State how your algorithm will ensure that genomes and the output of the cross-over and mutation operators represent feasible solutions.

2016 AI Exam (Batch 4)
Question 3

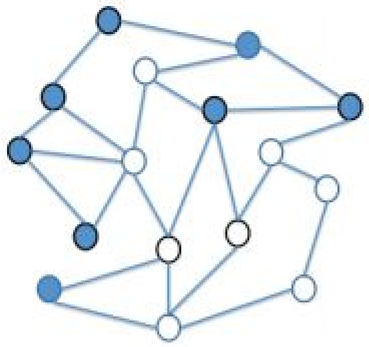
(a) ii. If Kirkpatrick's cooling schedule is used in simulated annealing and the temperature is initially set to 10, what is the temperature on the third iteration of the Metropolis algorithm, when α = 0.9?

iii. In a simulated annealing minimisation problem, if the objective function value at a state i is 2, a neighbour j has an objective function value of 3 and the temperature value is T = 1, what is the probability that a move to j will be accepted?

iv. When T = 1 in a simulated annealing algorithm, a uniform random number generator returns a value of 0.4 and the difference in objective function value between a state i and its neighbour j is f(j) − f(i) = 1.5. Is state j accepted or not?

v. Define the graph partitioning problem for splitting a graph into two disjoint clusters?

vi. What is the edge-cut of the graph partitioning shown below?



(a) i. A general purpose problem solving framework which is not specific to a particular application domain. (1 mark = general purpose framework) (1 mark = not application specific) **(2 marks)**

ii. Answer is $10 \times \alpha \times \alpha = 8.1$ when $\alpha = 0.9$

(2 marks)

iii. Probability of acceptance $= \exp(-(f(j) - f(i))/T) = \exp(-(3-2)/1) = \exp(-1) = 1/e = 0.367$

(2 marks)

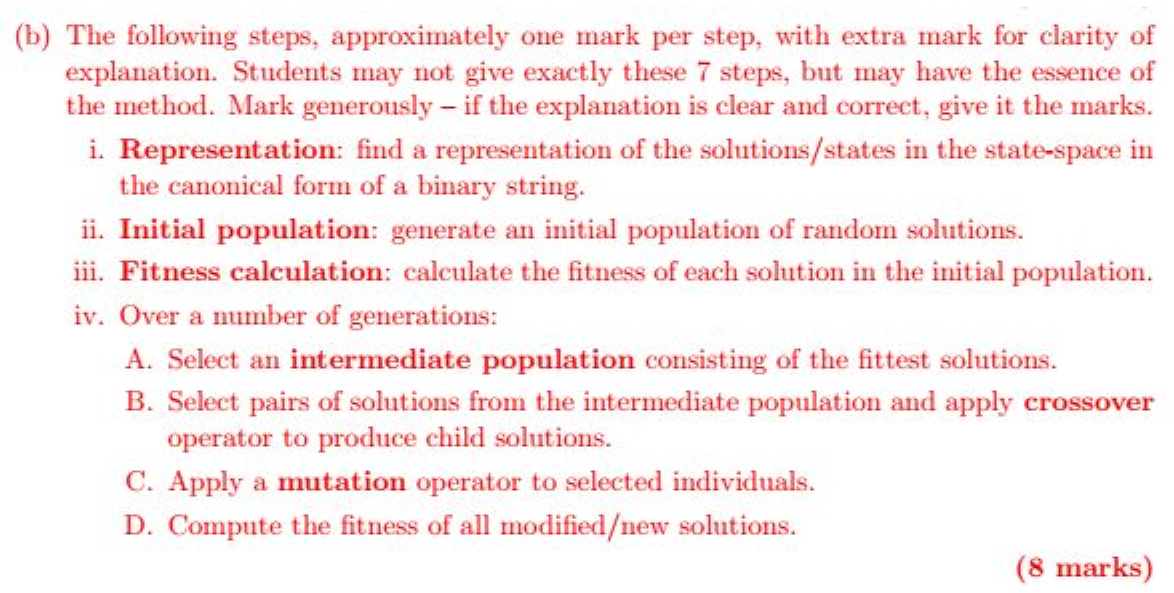
iv. $\exp(-(f(j) - f(i))/T) = \exp(-1.5) = 0.22$ Since $0.22 < 0.4$, the state j is accepted **(2 marks)**

v. The *graph partitioning problem* = to find a partition of the nodes in the graph into **disjoint** clusters, such that each cluster is the **same size** and there is the fewest number of edges joining nodes in different clusters i.e. **edge-cut is minimised** **(2 marks)**

vi. 12 (i.e. the number of edges joining nodes in different partitions)

(2 marks)

(b) Describe the steps that a genetic algorithm carries out in order to solve an optimisation problem.



(b) The following steps, approximately one mark per step, with extra mark for clarity of explanation. Students may not give exactly these 7 steps, but may have the essence of the method. Mark generously – if the explanation is clear and correct, give it the marks.

i. **Representation**: find a representation of the solutions/states in the state-space in the canonical form of a binary string.

ii. **Initial population**: generate an initial population of random solutions.

iii. **Fitness calculation**: calculate the fitness of each solution in the initial population.

iv. Over a number of generations:

   A. Select an **intermediate population** consisting of the fittest solutions.

   B. Select pairs of solutions from the intermediate population and apply **crossover** operator to produce child solutions.

   C. Apply a **mutation** operator to selected individuals.

   D. Compute the fitness of all modified/new solutions.

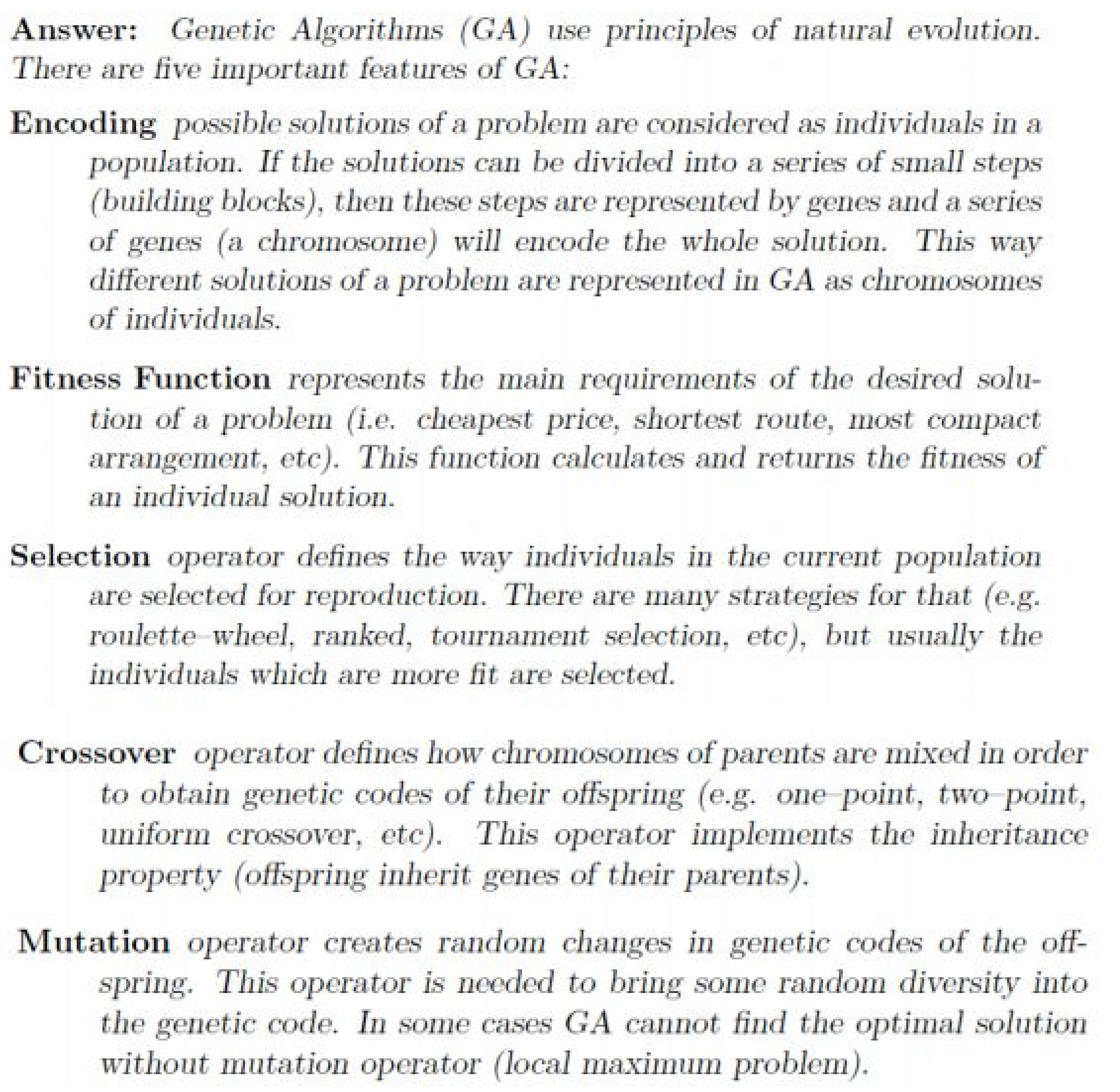
(8 marks)

Extra questions and answers

Question One

a.

Name and describe the main features/elements/steps of Genetic Algorithms (GA).



Answer: *Genetic Algorithms (GA) use principles of natural evolution. There are five important features of GA:*

Encoding *possible solutions of a problem are considered as individuals in a population. If the solutions can be divided into a series of small steps (building blocks), then these steps are represented by genes and a series of genes (a chromosome) will encode the whole solution. This way different solutions of a problem are represented in GA as chromosomes of individuals.*

Fitness Function *represents the main requirements of the desired solution of a problem (i.e. cheapest price, shortest route, most compact arrangement, etc). This function calculates and returns the fitness of an individual solution.*

Selection *operator defines the way individuals in the current population are selected for reproduction. There are many strategies for that (e.g. roulette–wheel, ranked, tournament selection, etc), but usually the individuals which are more fit are selected.*

Crossover *operator defines how chromosomes of parents are mixed in order to obtain genetic codes of their offspring (e.g. one–point, two–point, uniform crossover, etc). This operator implements the inheritance property (offspring inherit genes of their parents).*

Mutation *operator creates random changes in genetic codes of the offspring. This operator is needed to bring some random diversity into the genetic code. In some cases GA cannot find the optimal solution without mutation operator (local maximum problem).*

b.
Suppose a genetic algorithm uses chromosomes of the form x = abcdefgh with a fixed length of eight genes. Each gene can be any digit between 0 and 9. Let the fitness of individual x be calculated as:
$f(x) = (a + b) - (c + d) + (e + f) - (g + h)$,
and let the initial population consist of four individuals with the following chromosomes:

x1 = 6 5 4 1 3 5 3 2
x2 = 8 7 1 2 6 6 0 1
x3 = 2 3 9 2 1 2 8 5
x4 = 4 1 8 5 2 0 9 4

a) Evaluate the fitness of each individual, showing all your workings, and arrange them in order with the fittest first and the least fit last.

Answer:

$$\begin{aligned}
f(x_1) &= (6+5)-(4+1)+(3+5)-(3+2)=9\\
f(x_2) &= (8+7)-(1+2)+(6+6)-(0+1)=23\\
f(x_3) &= (2+3)-(9+2)+(1+2)-(8+5)=-16\\
f(x_4) &= (4+1)-(8+5)+(2+0)-(9+4)=-19
\end{aligned}$$

The order is x_2, x_1, x_3 and x_4.

b) Perform the following crossover operations:

i) Cross the fittest two individuals using one–point crossover at the middle point.

Answer: *One–point crossover on x_2 and x_1:*

$$\begin{array}{rr|l} x_2 = & \mathbf{8\,7\,1\,2} & \mathbf{6\,6\,0\,1} \\ x_1 = & 6\,5\,4\,1 & 3\,5\,3\,2 \end{array} \Rightarrow \begin{array}{rl} O_1 = & \mathbf{8\,7\,1\,2}\,3\,5\,3\,2 \\ O_2 = & 6\,5\,4\,1\,\mathbf{6\,6\,0\,1} \end{array}$$

ii) Cross the second and third fittest individuals using a two–point crossover (points b and f).

Answer: *Two–point crossover on x_1 and x_3*

$$\begin{array}{rr|c|l} x_1 = & \mathbf{6\,5} & \mathbf{4\,1\,3\,5} & \mathbf{3\,2} \\ x_3 = & 2\,3 & 9\,2\,1\,2 & 8\,5 \end{array} \Rightarrow \begin{array}{rl} O_3 = & \mathbf{6\,5}\,9\,2\,1\,2\,\mathbf{3\,2} \\ O_4 = & 2\,3\,\mathbf{4\,1\,3\,5}\,8\,5 \end{array}$$

iii) Cross the first and third fittest individuals (ranked 1st and 3rd) using a uniform crossover.

Answer: *In the simplest case uniform crossover means just a random exchange of genes between two parents. For example, we may swap genes at positions a, d and f of parents x_2 and x_3:*

$$\begin{array}{ll} x_2 = & \underline{8}\,7\,1\,\underline{2}\,6\,\underline{6}\,0\,1 \\ x_3 = & \underline{2}\,3\,9\,\underline{2}\,1\,\underline{2}\,8\,5 \end{array} \Rightarrow \begin{array}{ll} O_5 = & \mathbf{2\,7\,1\,2\,6\,2\,0\,1} \\ O_6 = & \mathbf{8\,3\,9\,2\,1\,6\,8\,5} \end{array}$$

c) Suppose the new population consists of the six offspring individuals received by the crossover operations in the above question. Evaluate the fitness of the new population, showing all your workings. Has the overall fitness improved?

Answer: *The new population is:*

$$\begin{array}{lcl} O_1 & = & 8\,7\,1\,2\,3\,5\,3\,2 \\ O_2 & = & 6\,5\,4\,1\,6\,6\,0\,1 \\ O_3 & = & 6\,5\,9\,2\,1\,2\,3\,2 \\ O_4 & = & 2\,3\,4\,1\,3\,5\,8\,5 \\ O_5 & = & 2\,7\,1\,2\,6\,2\,0\,1 \\ O_6 & = & 8\,3\,9\,2\,1\,6\,8\,5 \end{array}$$

Now apply the fitness function $f(x) = (a+b)-(c+d)+(e+f)-(g+h)$:

$$\begin{array}{lcl} f(O_1) & = & (8+7)-(1+2)+(3+5)-(3+2) = 15 \\ f(O_2) & = & (6+5)-(4+1)+(6+6)-(0+1) = 17 \\ f(O_3) & = & (6+5)-(9+2)+(1+2)-(3+2) = -2 \\ f(O_4) & = & (2+3)-(4+1)+(3+5)-(8+5) = -5 \\ f(O_5) & = & (2+7)-(1+2)+(6+2)-(0+1) = 13 \\ f(O_6) & = & (8+3)-(9+2)+(1+6)-(8+5) = -6 \end{array}$$

The overall fitness has improved.

d) By looking at the fitness function and considering that genes can only be digits between 0 and 9 find the chromosome representing the optimal solution (i.e. with the maximum fitness). Find the value of the maximum fitness.

Answer: *The optimal solution should have a chromosome that gives the maximum of the fitness function*

$$\max f(x) = \max\left[(a+b) - (c+d) + (e+f) - (g+h)\right] \ .$$

Because genes can only be digits from 0 to 9, the optimal solution should be:

$$x_{\text{optimal}} = 9\,9\,0\,0\,9\,9\,0\,0\ ,$$

and the maximum fitness is

$$f(x_{\text{optimal}}) = (9+9) - (0+0) + (9+9) - (0+0) = 36$$

e) By looking at the initial population of the algorithm can you say whether it will be able to reach the optimal solution without the mutation operator?

Answer: *No, the algorithm will never reach the optimal solution without mutation. The optimal solution is $x_{\text{optimal}} = 9\,9\,0\,0\,9\,9\,0\,0$. If mutation does not occur, then the only way to change genes is by applying the crossover operator. Regardless of the way crossover is performed, its only outcome is an exchange of genes of parents at certain positions in the chromosome. This means that the first gene in the chromosomes of children can only be either 6, 8, 2 or 4 (i.e. first genes of x_1, x_2, x_3 and x_4), and because none of the individuals in the initial population begins with gene 9, the crossover operator alone will never be able to produce an offspring with gene 9 in the beginning. One can easily check that a similar problem is present at several other positions. Thus, without mutation, this GA will not be able to reach the optimal solution.*

Question Two

Outline the simulated annealing cooling schedule, describing the various components.

The cooling schedule of a simulated annealing algorithm consists of four components.

- Starting Temperature
- Final Temperature
- Temperature Decrement
- Iterations at each temperature

Starting Temperature

The starting temperature must be hot enough to allow a move to almost any neighbourhood state. If this is not done then the ending solution will be the same (or very close) to the starting solution. Alternatively, we will simply implement a hill climbing algorithm.
However, if the temperature starts at too high a value then the search can move to any neighbour and thus transform the search (at least in the early stages) into a random search. Effectively, the search will be random until the temperature is cool enough to start acting as a simulated annealing algorithm.
The problem is finding the correct starting temperature. At present, there is no known method for finding a suitable starting temperature for a whole range of problems. Therefore, we need to consider other ways.

If we know the maximum distance (cost function difference) between one neighbour and another then we can use this information to calculate a starting temperature.
Another method, suggested in (Rayward-Smith, 1996), is to start with a very high temperature and cool it rapidly until about 60% of worst solutions are being accepted. This forms the real starting temperature and it can now be cooled more slowly.
A similar idea, suggested in (Dowsland, 1995), is to rapidly heat the system until a certain proportion of worse solutions are accepted and then slow cooling can start. This can be seen to be similar to how physical annealing works in that the material is heated until it is liquid and then cooling begins (i.e. once the material is a liquid it is pointless carrying on heating it).

Final Temperature

It is usual to let the temperature decrease until it reaches zero. However, this can make the algorithm run for a lot longer, especially when a geometric cooling schedule is being used (see below).
In practise, it is not necessary to let the temperature reach zero because as it approaches zero the chances of accepting a worse move are almost the same as the temperature being equal to zero.
Therefore, the stopping criteria can either be a suitably low temperature or when the system is "frozen" at the current temperature (i.e. no better or worse moves are being accepted).

Temperature Decrement

Once we have our starting and stopping temperature we need to get from one to the other. That is, we need to decrement our temperature so that we eventually arrive at the stopping criterion. The way in which we decrement our temperature is critical to the success of the algorithm. Theory states that we should allow enough iterations at each temperature so that the system stabilises at that temperature. Unfortunately, theory also states that the number of iterations at each temperature to achieve this might be exponential to the problem size. As this is impractical we need to compromise. We can either do this by doing a large number of iterations at a few temperatures, a small number of iterations at many temperatures or a balance between the two.

One way to decrement the temperature is a simple linear method.
An alternative is a geometric decrement where

$$t = t\alpha$$

where $\alpha < 1$.

Experience has shown that α should be between 0.8 and 0.99, with better results being found in the higher end of the range. Of course, the higher the value of α, the longer it will take to decrement the temperature to the stopping criterion.

Iterations at each Temperature

The final decision we have to make is how many iterations we make at each temperature.
A constant number of iterations at each temperature is an obvious scheme.

Another method, first suggested by (Lundy, 1986) is to only do one iteration at each temperature, but to decrease the temperature *very* slowly. The formula they use is

$$t = t/(1 + \beta t)$$

where β is a suitably small value. I have entered this formula on a spreadsheet (available from the web site) so that you can play around with the parameters, if you are interested.

An alternative is to dynamically change the number of iterations as the algorithm progresses. At lower temperatures it is important that a large number of iterations are done so that the local optimum can be fully explored. At higher temperatures, the number of iterations can be less.

Question Three

Show a simulated annealing algorithm

This is the algorithm shown in the course textbook. Other algorithms may also be acceptable (bearing in mind that this lecture was given by a guest lecturer who may have presented the algorithm differently).

Marks will also be given for showing a hill climbing algorithm (one that only accepts better neighbourhood moves) that has an acceptance criteria that accepts worse moves with some probability (that probability being exp(LE/T))

Vocabulary

Simulate : බොරුවට අඟවනවා
Annealing : anneal : මෘදු කරනවා
Heuristic : (it is like a hint or a guide) enabling a person to discover or learn something for themselves.
Mimic : අභිරූපක, ව්‍යාජ
Proportional : සමානුපාතික
Unintuitive - intuitive : සහජඥානය පිළිබඳ
Evolution -
Evaluation -
Mutation : විපර්යාසය, වෙනස් වීම
Crossover : මාරු වෙනවා
Arbitrary : අභිමතානුකූල