Assembling Of Personal Computer using Genetic Algorithm

Vrishank Gupta(2016UCO1677)
Sumit Nagpal(2016UCO1658)
COE 3

Problem Statement :-

Assembling a perfect personal computer, that meets various varying requirements of a family such as gaming, regular usage, programming etc., in such a huge market of features is quite a challenge now-a-days.

Despite the efforts put by consumers to customise their computers to meet the different requirements, the percentage of satisfied consumers is very less. The following genetic algorithm tries to find the optimum set of features, given that each feature adds to the cost of the computer but provides some benefit to the consumer, the selected features must be fulfilled within given budget. Here, this algorithm is being applied on a hypothetical financially middle class family who is trying to model their perfect computer within their budget.

Problem Reference:

This problem can be related to **0-1 Knapsack Problem** which is a pseudo polynomial combinatorial optimisation problem. Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The problem often arises in resource allocation where there are financial constraints and is studied in fields such as combinatorics, computer science, complexity theory, cryptography, applied mathematics, and daily fantasy sports.

It's time complexity is :-$O(n * w)$

where n:- number of items
w:- capacity of knapsack
It is polynomial in numerical value of w.

However, its actual complexity is:

$$O(n * 2^L)$$

where 2^L represents binary representation of w and for large w, this problem becomes less exponential.

Feature Set / Problem Features:-

S. No	Preferred Features	Value (Benefit)	Weight (Cost)	Qi (Generation Importance)	Si (Feature Importance)
1	Type (Laptop) (Portability)	500	300	5	2
2	Type (Desktop)	300	280	3	1
3	Processor Brand (Top brands: Intel, AMD etc)	700	600	5	3
4	Processor Brand (Intermediate brands: MediaTek)	650	400	5	3
5	Processor Brand (Lower Brands :RISE Technology)	400	300	5	3
6	Processor Technology (i-series)	900	500	7	8
7	Processor Technology (Core series)	700	400	5	8
8	Processor Technology (Pentium Series)	650	350	3	8
9	RAM Type (DDR3) (Clock Speed)	300	300	3	3
10	RAM Type (DDR4)	390	350	4	3
11	RAM Type (DDR5)	450	380	6	3
12	RAM Capacity (< 1 GB)	100	100	2	6
13	RAM Capacity (1-8 GB)	180	160	4	6
14	RAM Capacity (> 8 GB)	500	250	5	6
15	Cabinet Type (Fancy Gaming)	100	300	3	1
16	Cabinet Type (Regular)	120	200	3	1
17	GPU(Inbuilt)	50	20	2	4
18	GPU Dedicated(<1 GB)	200	100	4	4
19	GPU Dedicated(>1 GB)	400	250	5	4
20	Hard Drive (Solid State Drive)	800	400	7	7
21	Hard Drive (Magnetic Tape)	500	200	5	7
22	Hard Drive Capacity (< 500 GB)	300	200	4	4
23	Hard Drive Capacity (> 500 GB)	600	350	6	4

Chromosome:

Binary encoded chromosome has been used in this Genetic Algorithm

Example:-

1	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Constraints:

1. Budget Constraints: < ₹ 3030.
2. A feature may be selected or not selected (either 0 or 1).
3. Weight (cost) and value (benefit) are integral values.
4. Feature sets {1, 2}, {3,4,5}, {6,7,8}, {9,10,11}, {12,13,14}, {15,16}, {17,18,19}, {20,21} and {22,23}, can't be all zeroes at same time and at max, only one of the members of above listed sets can be '1'.

Fitness Function:

A feature gets more importance if it's of some later generation. A later generation feature tends to give better outcomes in terms of customer satisfaction. Moreover, some features tend to be more important than the others, like Processor generation is obviously more important than type of cabinet (i.e. fancy or regular). Therefore, it's better to assign priorities to each feature and same with generations of each feature to maximise customer satisfaction.

Therefore, based on above understandings, the following fitness function may be proposed

$$F_i = \frac{\sum (i = 0,n)(Pi * Ci * Vi)}{\sum (i = 0,n) Wi}$$

where,

$$Pi = 0.6 * Qi + 0.4 * Si$$

Q_i depends on Generation factor and S_i depends on net importance of i^{th} component in the final product.

n = length of chromosome (here, 23).

C_i = allele or value of gene of i^{th} chromosome.

V_i = corresponding value of task from feature set.

W_i = corresponding weight of task from feature set.

Objective Function:

$$G = \max\left(\frac{\sum(i=0,n)(Pi * Ci * Vi)}{\sum(i=0,n)Wi}\right)$$

where,

$$Pi = 0.6 * Qi + 0.4 * Si$$

Qi depends on latest Generation and Is depends on net importance of i^{th} component

n = length of chromosome (here,23)

C_i = allele or value of gene of i^{th} chromosome.

V_i = corresponding value of task from feature set.

W_i = corresponding weight of task from feature set.

Genetic Algorithm Overview:-

> Pseudocode:-

-> Make initial Population.
-> Calculate it's fitness.
-> Repeat till GA doesn't terminate,
    ->Select Parent.
    ->Perform Crossover.
    ->Perform Mutation.
    -> Select Survivors.
    -> Check if Terminating Condition is True or False.
-> Terminate GA if terminating condition is true.
-> Print Solution.

>Genetic Algorithm Operators:

Parent Selection:-

Two methods have been employed here:-

1. Roulette Wheel Method :- It is a stochastic method of selecting parent, which consists of a circular wheel that is divided on basis of frequency of a fitness value. A fixed point is chosen on wheel circumference and the wheel is rotated. The point of the wheel that comes in front of the fixed point is chosen as a parent. For second parent, process is repeated.

2. Tournament Method:- In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent. Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.

Crossover Operator:-

Three methods have been deployed for crossover:

1. **<u>One-Point Crossover:</u>**- In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.

For Example:-

Parent p1:-

1	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	1

Parent p2:-

0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	1	1	0

Offspring o1:-

1	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	0	1	0	0	1	1	0

Offspring o2:-

0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	1	1	0	0	1	0	0	1

2. **<u>Two-points Crossover:-</u>** In two-point crossover, two crossover points are picked randomly from the parent chromosomes. The bits in between the two points are swapped between the parent organisms.

<u>For Example:-</u>

Parent p1:-

1	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	1

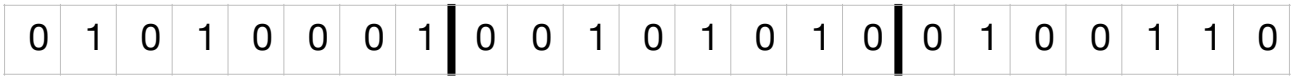
Parent p2:-

0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	1	1	0

Offspring o1:-

1	0	1	0	0	1	0	0	0	0	1	0	0	1	0	1	1	0	0	1	0	0	1

Offspring o2:-

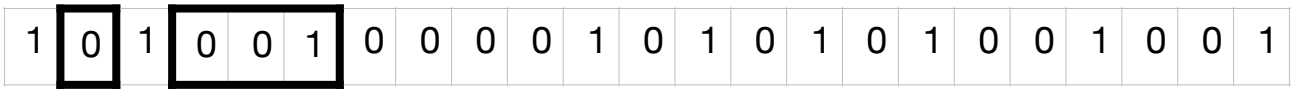


3. **Uniform Crossover** :- In uniform crossover, each bit from the offspring's genome is independently chosen from the two parents according to a given distribution. In contrast to k-point crossover, uniform crossover exchanges individual bits and not segments of the bit array. This means there is no bias for two bits that are close together in the array to be inherited together.

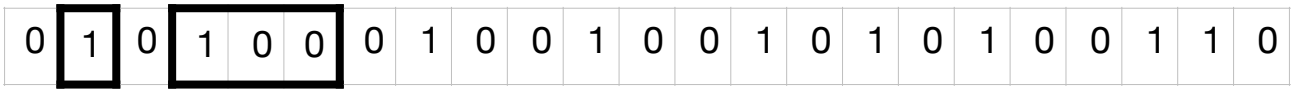
   Typically, each bit is chosen from either parent with equal probability. Other mixing ratios are sometimes used, resulting in offspring which inherit more genetic information from one parent than the other.

For Example:-

Parent p1:-



Parent p2:

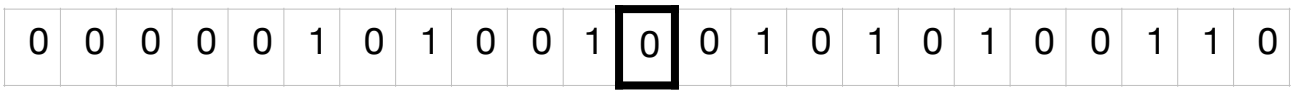


Offspring o1:-

Offspring o2:-

Mutation Operator:- Here we apply Single bit flip mutation operator. In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.

Offspring o1:-



After Bit Flip Mutation

0	0	0	0	0	1	0	1	0	0	1	**1**	0	1	0	1	0	1	0	0	1	1	0

Survivor Selection:-

Two basis of selecting survivors for next generation have been used:-

1. Age Based:- In Age-Based Selection, we don't have a notion of a fitness. It is based on the premise that each individual is allowed in the population for a finite generation where it is allowed to reproduce, after that, it is kicked out of the population no matter how good its fitness is.

2. Fitness based:- In this fitness based selection, the children tend to replace the least fit individuals in the population. The selection of the least fit individuals may be done using a variation of any of the selection policies described before – tournament selection, fitness proportionate selection, etc.

Termination Condition:-

The termination condition of a Genetic Algorithm is important in determining when a GA run will end. It has been observed that initially, the GA progresses very fast with better solutions coming in every few iterations, but this tends to saturate in the later stages where the improvements are very small. We usually want a termination condition such that our solution is close to the optimal, at the end of the run.

Usually, we keep one of the following termination conditions –

- When there has been no improvement in the population for X iterations.
- When we reach an absolute number of generations.
- When the objective function value has reached a certain pre-defined value.

Graphical Analysis :-

1. Varying Initial Population :-

   a). Population varied over 50-100 range, end graph observed for uniform and two point crossover for Roulette method. As one and two points crossover give same results, roulette method can be better observed on a graph.

   b). Generation vs Fitness graphs for roulette selection with uniform crossover, fitness based survivor selection and varying initial population, plotting best and average fitness.

Conclusion :-

-> On varying initial population, uniform crossover shows abrupt changes due to it's probabilistic nature, whereas two-points crossover presents a smooth curve.

-> Graph becomes flatter with increasing generations as we reach goal state.

-> Higher initial population reaches goal states faster.

2. Roulette wheel selection :-

-> Best crossover method, in terms of faster to find solution, retains characteristics with increase in termination condition and requires less numbers of generations, is found out to be 'Uniform Crossover' graphically.

-> Age based survivor selection doesn't work with roulette method as GA doesn't converge.

3. Tournament Selection :-

-> K is taken small as for large K, GA converges faster.

-> One point crossover with age based survivor selection and uniform crossover with fitness based survivor selection, remains consistent in their shape and characteristics and hence are stable.

->Fastest method for tournament selection is one point crossover with fitness based survivor selection.

4. Roulette vs. Tournament Selection :- Tournament selection is graphically more stable and converges faster to goal state than roulette selection and hence better than it.

References :-

-> **https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3**

-> **https://www.tutorialspoint.com/genetic_algorithms/**

->**http://www.cs.jhu.edu/~ayuille/courses/Stat202C-Spring10/ga_tutorial.pdf**

-> Introduction to Genetic Algorithms by Sivanandam, S.N., Deepa.