Application Of Genetic Algorithm for Assembling a Personal Computer

Vrishank Gupta
(2016UCO1677)
Sumit Nagpal
(2016UCO1658)
Computer Engineering
Netaji Subhas University of Technology, Delhi-110078

Abstract- 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. This paper proposes a genetic algorithm 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. Experimental result yields the average fitness convergence at value 5524 which is marked improvement over of 23% over recently published paper Maya Hristakeva of Simpson University along with Dipti Shreshtha who used Group Selection Technique along with single point crossover for hardware selection.

Keywords: Genetic Algorithm, Computer Assembling, Hardware Selection

I. Introduction

Assembling a personal computer with so many options available for each component such as RAM, Processor generation and technology, Supply power unit, hard drive, cabinets, Graphical Processing Unit whether dedicated or inbuilt leaves us with so many varieties to choose from. However, having so much variety creates a room for confusion, specially when there is limited budget to be spared, we can't just pick the best possible variant as it might exceed allowed budget.

Therefore we need a strategy to pick items that satisfy costumer demands and provides them best possible version available within their defined budget. If we pick elements randomly, there is a possibility that some feature might get missed or we pick the different versions of same feature twice, this must also be avoided hence it won't be feasible to pick randomly. Moreover, any strategy that is being followed must give results in real time within defined time based constraints. So, here is a Genetic Algorithm based approach that tries to achieve the above said aim. Genetic algorithm is an effective optimisation technique, it is a heuristic search process, and

completely overcome the traditional optimisation algorithm is easy to fall into local optimal solution of the defect, which jumped by chromosome mutation local optimal solution, which eventually converge to the global optimal solution. In this paper, genetic algorithm based on real optimal nesting algorithm, and describes the method used to achieve goal. 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.

II. Related Work

Application of Genetic Algorithm on decision-based optimisation problems has been worked upon a lot by various researchers. The dataset has been massive, complex, inconsistent, and has no fixed rules as continuous changes and updates. This challenge always grasps attention of many researchers in the field of computer science as it is closely related to 0-1 Knapsack Problem 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.

Numerous claims have been provided to assemble an ideal budget computer that meets all needs.

Joel Hruska on his webpage, gave various comparison for cases for the pc i.e. high tower, low tower and suggested various combinations amongst them. For cooling, he compared benefits of heatsink by AMD and NVIDIA. He also contrasted between motherboards by Gigabyte and Noctua, in terms of price and utilities i.e.Mini-ITX, Micro-ATX (mATX), and ATX. For processors, he quoted "Intel's latest Core X series processors offer ludicrous amounts of speed and cores for those who can spend well above $500 on processors alone, while AMD's Ryzen series competes on frugality, with savings of several hundred dollars at the same general performance level."

Maya Hristakeva of Simpson University along with Dipti Shreshtha [1] terminated their algorithm when either 90% of the chromosomes in the population have the same fitness value or the number of generations is greater than a fixed number. This resulted in **assured convergence** of the algorithm in finite time to produce the desired fitness in the individuals. This also denied premature convergence of the dataset to produce nearly optimal solutions. They also applied an unconventional technique for selection and called that as "Group Selection". The main motive was to increase the probability of choosing fitter chromosomes to reproduce more often than chromosomes with lower fitness values.

Author compared Group selection with conventional Roulette wheel selection method and found that the results from the two selection functions, roulette-wheel and group selection, differ a lot depending on whether we used elitism to enhance their performance or not. When we do not use elitism the group selection method is better than the roulette-wheel selection method, because with group selection the probability of choosing the best chromosome over the worst one is higher than it is with roulette-wheel selection method. Thus, the new generation will be formed by fitter chromosomes and have a bigger chance to approximate the optimal solution. When we use elitism than the results from roulette-wheel and group selection method are similar because with elitism the two best solutions found throughout the run of the program will not be lost.

Sarac [2] developed a genetic algorithm for solving this problems in which GA's performance was optimised using [3]. Bhatia and Basu [4] proposed a gene induction approach for genetic algorithms. The approach was applied to 0/1 knapsack problem, and found near optimal results in all the representative problem instances in the literature. Pawlak [5] introduced RS that can process data with uncertainty, reduce the size of data sets, and generate decision rules from the data sets. In these data sets, some attributes may be redundant and can be eliminated without reducing the original classification quality. In RS, the process of finding a smaller set of attributes that ensures the same classification quality is called attribution reduction and the underlying set is referred to as a deduct. This paper presents an efficient genetic algorithm with fast convergence towards results to solve knapsack problem.

A genetic algorithm in which the number of individuals changes to show increase in accuracy of solution is shown in [6]. The number of population is doubled initially against the accuracy of solution. First stage increases the searching ability. Then, accuracy is improved at second stage by reducing the number of population. Wei, beibel and jiang derived an improved solution for 0-1 knapsack problem based on the dual population genetic algorithm, which can overcome the find of precocious and local convergence in iterative processes. The performance valuation shows that the solution is better than the traditional genetic algorithm [7]. Yanqin Ma and Jianchen Wan solves the 0-1 knapsack problem with the hybrid adaptive genetic algorithm which combined with greedy algorithm. It presents a method for optimal design of an improved adaptive algorithm and repairs the infeasible

solution with greedy algorithm [8] while Basima and Moutaz applied several mutation methods to different non-deterministic polynomial (NP) hard problems [9].

III. Chromosome

Binary encoded chromosome has been used in this Genetic Algorithm. It is a
23 bit chromosome where each bit represents one feature. 0 represents that feature is excluded and 1 represents inclusion of that feature. A sample chromosome:-

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

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'.

IV. 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 for the context:-

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

where,

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

where,

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.

V. Objective Function

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

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.

VI. 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.

VII. Crossover Operator

Three methods have been deployed for crossover:

1. One Point Crossover:- A random crossover point is selected and the tails of its two parents are swapped to get new off- springs.

2. 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.

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.

VIII. Mutation Operator

We have applied 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.

IX. 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.

X. 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's been no improvement in the population's fitness for X iterations.

-> When we reach an absolute number of generations.

-> When the objective function value has reached a certain pre-defined value.

XI. Graphical Analysis

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.

XII. 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 :-

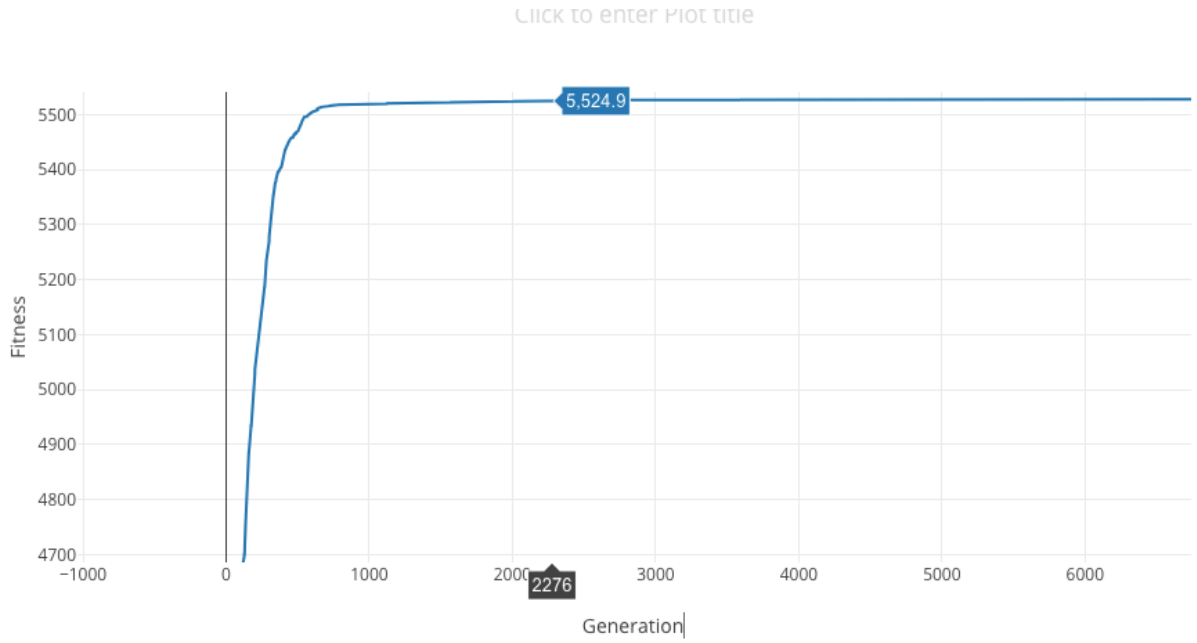
-> K is taken small as for large K,GA converges faster.

-> 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.

The Genetic Algorithm based approach produces nearly best solution. While converging towards best fitness under given constraints, it shows the fitness of each chromosome of each individual



XIII. References

[1] A. S. Anagun, and T. Sarac, ‘‘Optimization performance of genetic algorithm for item selection type problems using Taguchi method,’’ Lect. Notes Comput. Sc., vol. 3982, pp. 678-687, 2006.

-> https://www.extremetech.com/computing/263184-how-to-build-a-pc

[3] S. Chowdhury, and Y. Wu, “Taguchi’s Quality Engineering Handbook,” Wiley-Interscience, New York, NY, 2004.

[4] A. K. Bhatia, and S. K. Basu, “Tackling 0/1 knapsack similar problems with gene induction for item selection ,” Soft Comput., vol. 8, no. 1, pp. 1-9, 2003.

[5] Z. Pawlak, “Rough set,” Int. J. Inform. Comput. Sci., vol.11, pp. 341-356, 1982.

[6] Akihiko Tsukahara and Akinori Kanasugi, "Genetic algorithm that can dynamically change number of individuals and accuracy", IEEE Frontiers in the Convergence of Bioscience and Infromation Technologies, 2007, pp. 785-789.

[7] Wei Shen, Beibei Xu, Jiang-ping Huang, “An Improved Genetic Algorithm for 0-1 Knapsack Problems”, Second International Conference on Networking and Distributed Computing 2011.

[8] Yanqin Ma and Jianchen Wan, “Improved Hybrid Adaptive Genetic Algorithm for Solving Knapsack Problem”, The 2nd International conference on intelligent control and information processing 2011.

[9] Basima Hani Hasan and Moutaz Saleh Mustafa, “Comparative Study of Mutation Operators on the Behavior of Genetic Algorithms Applied to Non-deterministic Polynomial (NP) Problems”, Second International Conference on Intelligent Systems, Modelling and Simulation 2011.