ISSN 2355-0082

Design and Development of Computer Specification Recommendation System Based on User Budget With Genetic Algorithm

Michael[1], Winarno[2]
Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
[1]michael6@student.umn.ac.id
[2]pmwinarno@umn.ac.id

Received on September 12th, 2017
Accepted on June 8th, 2018

***Abstract*—There are a lot of things that must be considered when determining specifications of computer components to make sure those components chose are working compatible. According to a survey conducted to 78 respondents, about 72.5% of the respondents prefer to buy a built-up computer. The reason is because of a lack of knowledge of computer components and how to assemble computer properly. This research aimed to develop a recommendation system that able to give recommendation of buying computer based on compatible components to be assembled, with the available budget, so that people who do not know computer components can also buy a assembled computer. The Genetic Algorithm was chosen for making this recommendation system because this Algorithm gives more alternative solutions through the process of crossover and mutation compared to the Greedy Algorithm which doesn't produce a solution by trying all alternative solution nor Exhaustive Search on Brute Force Algorithm which takes a long time to find optimum solution. The recommendation system of computer components specifications based on the budget available has been successfully developed using Genetic Algorithm and achieved 75.75% user satisfaction.**

***Index Terms*— Computer Components, Genetic Algorithm, Greedy Algorithm, Recommendation System.**

I. INTRODUCTION

According to International Data Corporation (IDC), in 2016, overall sales of personal computer in Indonesia reached 2 million units. This indicates the need for computers is still quite a lot in Indonesia, especially for office and gaming needs.

In general there are two types of computer sold in computer stores, the built-up computers and the assembled computers. The built-up computers are computers directly made by the manufacturers of the computers, while the assembled computers are computers assembled by technicians base on buyer's request.

Based on a preliminary survey, about 72.5% of respondents who prefer to buy built-up computers reasoned they bought the computers because they don't know computer components. While 86.8% of the the respondents who prefer to buy assembled computers reasoned they bought the computer because they could adjust the price of the computer based on their budget.

The preliminary survey conclude that people bought built-up computers because they don't know computer components well. If there is a recommendation system able to give components specification of compatible computers based on their budget, they would buy an assembled computer so they can adjust the price of the computer base on their budget, and they no longer need to know computer components.

Researches related to the this problem has been done by some other researchers. Imbar in 2013 [1] select the specification of computer components based on the budget of the buyer. However, the research based on Greedy Algorithm produced sub-optimum solution because the algorithm did not operate thoroughly against all available alternative solutions.

Another research conducted by Haryanty in 2012 [2] using Genetic Algorithm with Roulette Wheel selection method and the result was producing combination of products optimal to buyer's budget. However, this research only used five computer components i.e processor, motherboard, memory (RAM), graphics card and hard disk; and did not check the compatibility of RAM according to the motherboard used.

This research used Genetic Algorithm to make improvement from the previous research by using seven computer component products and used Tournament Selection for the selection method which has an advantage in convergence speed compared to proportionate roulette wheel. It is expected to help people when buying assembled computer with the various of computer components and the incomprehension of computer components compatibility.

II. METHODOLOGY

A. Genetic Algorithm

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics.

GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. To evolve good solutions and to implement natural selection, measure to distinguish good solutions from bad solutions is needed. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over worse ones. The fitness measure must determine a candidate solution's relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions [1].

Another important concept of GAs is the notion of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. For example, small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time.

Once the problem is encoded in a chromosomal manner and a fitness measure for discriminating good solutions from bad ones has been chosen, the algorithm start evolving solutions using the following steps:

1. Initialization. The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated.
2. Evaluation. Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.
3. Selection. Selection allocates more copies of those solutions with higher fitness values and thus imposes the survival of the fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea, including roulette-wheel selection, stochastic universal selection, ranking selection and tournament selection.
4. Recombination. Recombination combines parts of two or more parental solutions to create new, possibly better solutions (i.e. offspring). There are many ways of accomplishing this (some of which are discussed in the next section), and competent performance depends on a properly designed recombination mechanism. The offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner (Goldberg, 2002).
5. Mutation. While recombination operates on two or more parental chromosomes, mutation locally but randomly modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes being made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution.
6. Replacement. The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation-wise replacement and steady-state replacement methods are used in GAs.

B. Tournament Selection

GAs uses a selection mechanism to select individuals from the population to insert into a mating pool. Individuals from the mating pool are used to generate new offspring, with the resulting offspring forming the basis of the next generation. A selection mechanism in GA is simply a process that favors the selection of better individuals in the population for the mating pool. The selection pressure is the degree to which the better individuals are favored: the higher the selection pressure, the more the better individuals are favored. This selection pressure drives the GA to improve the population fitness over succeeding generations. The convergence rate of a GA is largely determined by the selection pressure, with higher selection pressures resulting in higher convergence rates. However, if the selection pressure is too low, the convergence rate will be slow, and the GA will unnecessarily take longer to find the optimal solution. If the selection pressure is too high, there is an increased chance of the GA prematurely converging to an incorrect (suboptimal) solution.

Tournament selection provides selection pressure by holding a tournament among "s" competitors, with "s" being the tournament size. The winner of the tournament is the individual with the highest fitness of the "s" tournament competitors. The winner is then inserted into the mating pool. The mating pool, being comprised of tournament winners, has a higher average fitness than the average population fitness. This fitness difference provides the selection pressure,

which drives the GA to improve the fitness of each succeeding generation. Increased selection pressure can be provided by simply increasing the tournament size "s", as the winner from a larger tournament will, on average, have a higher fitness than the winner of a smaller tournament [2].

C. *Uniform Crossover*

Uniform crossover do not fragment the chromosomes for recombination. Each gene in offspring is created by copying it from the parent chosen according to the corresponding bit in the binary crossover mask of same length as the length of the parent chromosomes. If the bit in crossover mask is 1, then the resultant gene is copied from the first parent and if the bit in crossover mask is 0, then the resultant gene is copied from the second parent. A new crossover mask is generated arbitrarily for each pair of parent chromosomes. The quantity of crossover point is not fixed initially. So, the offspring have a mixture of genes from both the parents [3].

D. *Uniform Mutation*

The mutation operator randomly selects a position in the chromosome and changes the corresponding allele, thereby modifying information. The need for mutation comes from the fact that as the less fit members of successive generations are discarded; some aspects of genetic material could be lost forever. By performing occasional random changes in the chromosomes, GAs ensure that new parts of the search space are reached, which reproduction and crossover alone couldn't fully guarantee [4]. Uniform Mutation can be done using the following steps [5]. Choose one gene randomly. Replace the value of a chosen gene with a uniform random value selected between the user specified upper and lower bounds for that gene.

III. SYSTEM MODELING

A. *Use Case Diagram*

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. Fig 1 shows the Use Case Diagram of the application.

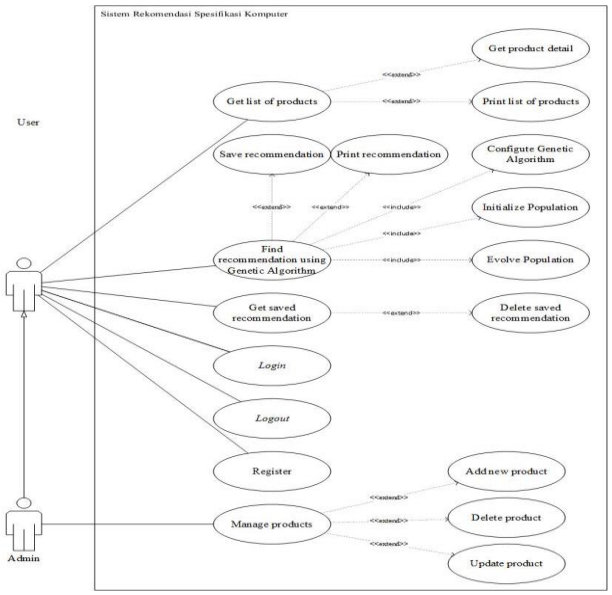

Fig 1. Use Case Diagram

B. *Class Diagram*

Class Diagram is UML structure diagram which shows structure of the designed system at the level of classes and interfaces, shows their features, constraints and relationships - associations, generalizations, dependencies, etc. Fig 2 shows the Class Diagram of the application.

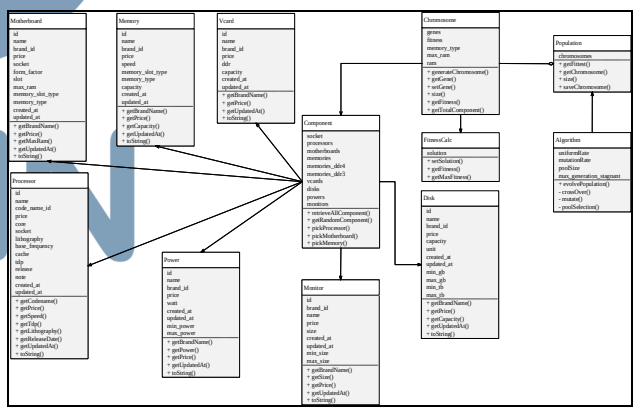

Fig 2. Class Diagram

IV. IMPLEMENTATION AND EXPERIMENT RESULTS

The system is built on a website platform using PHP, Bulma CSS framework, DataTables, and AngularJS. User will get recommendation of computer components base on their budget by using the following steps.

1. Defining the computer component that will be used. By the default, user will use all components.
2. Filter product by its minimum specification

3. Define budget.

A. *Computer Specification Recommendation System*

Users can define the computer components that will be used for getting a recommendation.

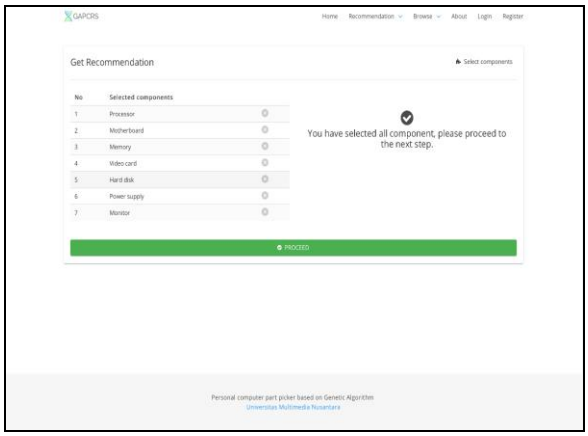

Fig 3. Defining Components

After user has defined the components, he/she will define the minimum specification of products for each component.

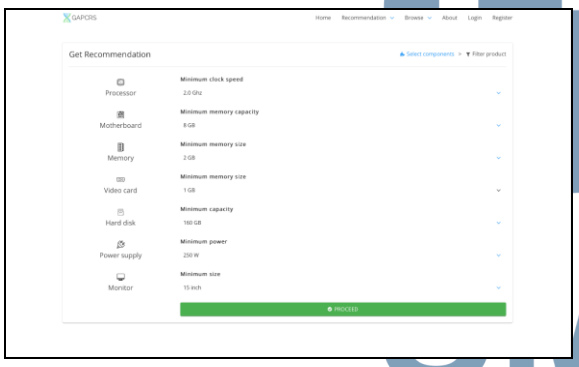

Fig 4. Defining Minimum Specification of Products

On the next page shown all of the components and their minimum specification of products. User will input budget and waiting until the Algorithm has found the solution. In Figure 3 the user has input IDR 3,500,000 to the budget.

Fig 5. Input Budget

The combination of computer components will be show after the algorithm has found solution. The compatibility for each component have been checked automatically by the application so user don't have to worry about compatibility issues.

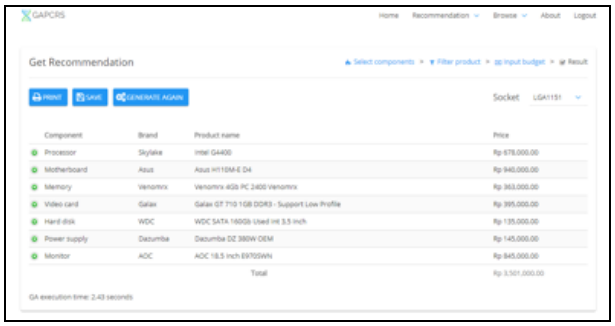

Fig 6. GA Result

B. *Testing*

Testing has been conducted by giving input IDR 3,500,000 to budget which is the worst case for the algorithm. The test used different population size and was performed for 30 times for each population size and processor's socket. The test set mutation rate and crossover rate 0.05 and 0.6 respectively. It is observed that the optimal values for mutation probability (0.001) and single point crossover with probability (0.6) with population size (50-100) as suggested by DeJong (1975) have been used in many GA implementations. Mutation probability above 0.05 is in general harmful for the optimal performance of Gas [6].

Table 1. Amount of Data Used in the Test

Component Name	Total Products
Processor (LGA 1151)	**17**
Processor (LGA 1150)	**17**
Processor(LGA 1155)	**25**
Motherboard	352
Memory	296
Video Card	458
Hard Disk	43
Power Supply	286
Monitor	252

Table 2. Test Result by Using 10 Population Size

Socket	Average Time	Average Generation	Average Difference
1151	0.626	502.7	Rp 10,500.00
1155	0.321666667	311.2333333	Rp 1,833.33
1150	0.414333333	323.9	Rp 2,400.00
Mean	0.454	379.277778	Rp 4,911.11

Table 3. Test Result by Using 30 Population Size

Socket	Average Time	Average Generation	Average Difference

Socket	Average Time	Average Generation	Average Difference
1151	1.049666667	326.3	Rp 2,600.00
1155	0.64	190.466667	Rp 866.67
1150	0.763	180.766667	Rp 800.00
Mean	0.81755556	232.511111	Rp 1,422.22

Table 4. Test Result by Using 50 Population Size

Socket	Average Time	Average Generation	Average Difference
1151	1.585	282.4	Rp 1,500.00
1155	0.824	136.5	Rp 666.67
1150	1.117666667	204.0666667	Rp 1,333.33
Mean	1.17555556	207.655556	Rp 1,166.67

Table 5. Test Result by Using 60 Population Size

Socket	Average Time	Average Generation	Average Difference
1151	1.80533333	270.233333	Rp 1,566.67
1155	1.097666667	155.4666667	Rp 966.67
1150	3.057	160.9	Rp 700.00
Mean	1.98666667	195.533333	Rp 1,077.78

Table 6. Test Result by Using 80 Population Size

Socket	Average Time	Average Generation	Average Difference
1151	1.99766667	224.233333	Rp 1,533.33
1155	1.68133333	183.033333	Rp 900.00
1150	1.36633333	153.8	Rp 600.00
Mean	1.68177778	187.022222	Rp 1,011.11

Table 7. Test Result by Using 100 Population Size

Socket	Average Time	Average Generation	Average Difference
1151	1.894	170.2	Rp 2,533.33
1155	1.594	137.9	Rp 466.67
1150	1.56266667	141.2	Rp 800.00
Mean	1.68355556	149.766667	Rp 1,266.67

C. Software Quality

Software quality questionnaire was filled by 33 respondents. The questionnaire questions were made based on EUCS model. By using Pearson Product-Moment Correlation, it was obtained that all the questions are valid, and by using Alpha Cronbach formula, it was obtained the questionnaire's reliability of 0.78 which can be considered as adequate [7]. The score for software quality is 75.75% which is considered good.

V. Conclusion and Future Works

Computer specification recommendation system using Genetic Algorithm has been successfully designed and developed. The score for software quality is 75.75% which is considered good.

Based on the experiments that have been carried out the most efficient population size in terms of time and the difference between budget and recommendation is 50. Larger population size will make the result is more accurate than smaller population size but it will take a longer time to find solution.

In order to improve the application, it is suggested that the application may provide a picture to each product and use different algorithm that has been mentioned before for crossover process, mutation process, and selection process to make the application works more efficient than before.

References

[1] Kumara Sastry, D. G., 2005. Genetic Algorithm. In: Search Methodologies:Introductory Tutorials in Optimization and Decision Support Techniques. Boston: Springer, Boston, MA, p. 97.

[2] Mehta, A. S. &. A., 2013. Review Paper of Various Selection Methods in Genetic Algorithm. International Journal of Advanced Research in Computer Science and Software Engineering, 3(7), pp. 1476-1479.

[3] Nitasha Soni, D. T. K., 2014. Study of Various Crossover Operators in Genetic Algorithms. International Journal of Computer Science and Information Technologies, 5(6), pp. 7235-7238.

[4] A.J. Umbarkar, P. S., 2015. CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. ICTACT JOURNAL ON SOFT COMPUTING, 6(1), pp. 1083-1092.

[5] Firas Alabsi, R. N., 2012. Comparison of Selection Methods and Crossover Operations using Steady State Genetic Based Intrusion Detection System. Journal of Emerging Trends in Computing and Information Sciences, 3(7), pp. 1053-1058

[6] D.D, P. V. &. P., 2015. The Optimal Crossover Or Mutation Rates In Genetic Algorithm. International Journal of Applied Engineering and Technology, 5(3), p. 40.

[7] Zahreza Fajar Setiara Putra, M. S. N. W., 2014. ANALISIS KUALITAS LAYANAN WEBSITE BTKP-DIY MENGGUNAKAN METODE WEBQUAL 4.0. Jurnal JARKOM, 1(2), pp. 174-184.