# Robust Adaptive Genetic *K*-Means Algorithm Using Greedy Selection for Clustering

Abba Suganda Girsang[1]
Master in Computer Science, Bina Nusantara University
Jakarta, Indonesia
agirsang@binus.edu

Fidelson Tanzil[2], Yogi Udjaja[3]
School of Computer Science, Bina Nusantara University
Jakarta, Indonesia
fidelson.tanzil@binus.ac.id[2], yogi.udjaja@binus.ac.id[3]

***Abstract*—Clustering is a task to divide objects into group depends on their similarity. The optimal of solving clustering problem occurs when the data joins in one group which has a similar category. This study combines Adaptive Genetic Algorithm, K-Means and Greedy Selection to solve clustering problem, named RAGKA. In first step, the centroid is determined by K-Means. Crossover and mutation are performed based on the fitness value of each centroid. At last, the greedy search is operated to get the better solution. To show the performance of RAGKA, five data sets of clustering problem are used. Moreover, RAGKA is compared with other methods as well. The result shows that RAGKA is successfully to solve cluster problem and outperforms than the others.**

***Keywords*: *Clustering, K-Means; Adaptive Genetic Algorithm; Greedy Selection* Introduction**

## I. INTRODUCTION

Clustering is one of the popular methods in the analysis of the data or called data mining techniques [1][2][3][4][5]. Clustering is useful for data reduction (reducing a large amount of data to a number of characterizing sub-groups), developing classification schemes, and suggesting or supporting hypotheses about the structure of the data [6]. The goal of clustering is to reduce the size of data by grouping similar data items together.

Many various approaches have been reported to solve clustering problems such as Artificial Bee Colony (ABC) [2][5], Ant Colony Algorithm [8][9], *K*-Means [7][11], Genetic Algorithm (GA) [11][12]. The most popular algorithm for clustering is *K*-Means algorithm which is a center based, simple and fast algorithm [1][14][15][16][17][18].

The basic *K*-Means algorithm for clustering depends on initial cluster center. Although the good performance to solve clustering, *K*-Means has the drawback that is easy trapped into local optimal [18]. Maulik and Bandyopadhyay [18] integrated the simplicity of the K-Means algorithm with the capability of GA in avoiding local optimum. The result showed that combination *K*-Means and GA is able to improve but still cannot achieve the optimal results. In GA, there are two operators namely crossover and mutation; normally probability of crossover and mutation were fixed but and significantly affect the behavior and the performance of GA [19]. They proposed and by adapting based on the fitness value of the solutions to solve some combinatorial problems.

This strategy is also used by Wulandhari, Wibowo, and Desa [20] to solve multiple data bearings. Based on succesful of these algorithm, RAGKA is proposed to get better solution in solving cluster problem. RAGKA firstly finds the centroid using *K*-Means. To avoid local optimal, crossover and mutation are performed based on the fitness value of each centroid. To enhance the optimal solution, the greedy search is operated.

## II. RELATED WORK

### *A. K-Means Clustering*

In general, clustering deals with the computation of a partition matrix $U = [u_{kj}]K_{\times n}$ , where $u_{kj}$ denotes the degree of membership of the *j*th data point, $x_j$, $j = 1, \ldots, n$, to cluster $C_k$, $K = 1, \ldots, K$. For clustering, $u_{kj} = 1$ if point $x_j$ belongs to cluster $C_k$ , and 0 otherwise. An example is the grouping of social networks; social networks can be clustered from different regions (see Fig. 1)

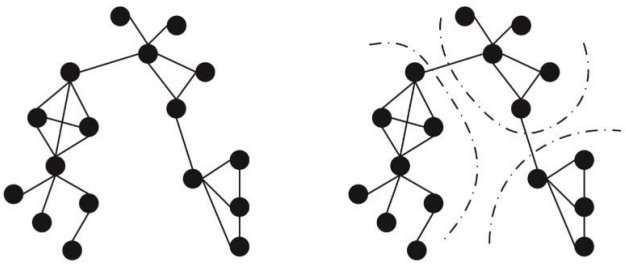

Fig. 1. Cluster social networks [21]

The most popular algorithm for clustering is *K*-Means algorithm. *K*-Means for clustering was developed in 1976 by McQueen [10]. *K*-Means is one technique that is widely used algorithms to solve problem of clustering. The basic steps of *K*-Means algorithm for clustering are as follows:

1) Choose $K$ initial cluster center $z_1, z_2, \ldots, z_K$ at randomly from the $n$ points $(x_1, x_2, \ldots, x_n)$
2) Assign point $x_i, i = 1,2, \ldots, n$ to cluster $C_j, j \in (1,2, \ldots, K)$ where $\| x_i - z_i \| \leq \| x_i - z_p \|, p = 1,2, \ldots, n$ and $j \neq p$.

978-1-5090-5130-4/16/$31.00 ©2016 IEEE

3) Compute new cluster center $z_1^*, z_2^*, \dots, z_K^*, z_i^* = \frac{1}{n_1} \sum_{x_j \epsilon C_i} x_j, i = 1, 2, \dots, K$, where $n_i$ is the number of elements belonging to cluster $C_i$.
4) If $z_i^* = z_i, i = 1,2, \dots, K$ then terminate. Otherwise continue from Step 2.

The above process shows the first thing to do is to determine the *K* cluster centers randomly, after that assign each data point to the nearest cluster center, and define a new center cluster, and calculate the value of center. In this case, the process does not terminate at Step 4, the process is executed depends on number of iterations [18].

B. Adaptive Genetic Algorithm (AGA)

Genetic Algorithm is a search technique based on the principles of biological evolution with a certain probability [22][8]. The main methods of GA consist:

1) Selection
Selection is the process of selecting the best chromosome as the survival of the natural genetic systems to emphasize the fitter individuals in the population in hopes that their next generation or offspring will in turn have even higher fitness [23].
2) Crossover
Crossover is a cross between chromosomes to produce a new chromosome better with a certain probability. In this study used a single point crossover. Crossover single point has the disadvantage that so called positional bias. Positional bias is the inability crossover to maintain the best fitness causing a best fitness could turn bad [23] (see Fig. 2).

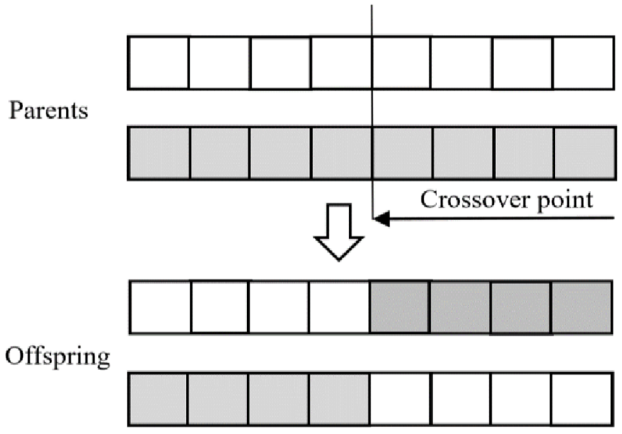


Fig. 2. Crossover single point operation [24]

3) Mutation
Mutations are changes in genes in the probability that the chromosome is stuck with a local optimum can produce a chromosome which has the optimal global value. Because of probability, mutations still have the possibility to get a local optimum [23] (see Fig. 3).

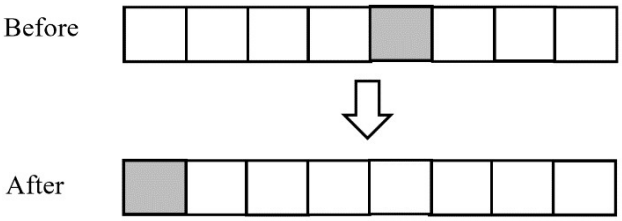


Fig. 3. Mutation operation [23]

The structure of GA in general is shown as follows:

**Step 1:** Initialization population.
**Step 2:** Evaluate the fitness for each chromosome.
**Step 3:** Perform genetic operators, including selection, crossover, and mutation, on the current population to introduce a new population.
**Step 4:** Exchange the current population with the new population.
**Step 5:** If criterion is met then stop the iterations, else go to Step 2.

AGA concept used in this study is consistent with the conditions of GA in general; the difference is in the crossover and mutation. The concept of crossover and mutation used in this study is after obtaining the best fitness value, best fitness value is maintained, so that the crossover and mutation need not be done again.

1) Adaptive Crossover

$$p_c(i, \emptyset_{1s}^i, \emptyset_{2s}^i) = \begin{cases} p_{c0} \dfrac{(F_{max}(i) - F'(i,s))}{(F_{max}(i) - \bar{F}(i))} & \text{if } F'(i,s) > \bar{F}(i) \\ p_{c0} & \text{otherwise} \end{cases} \quad (1)$$

Where

$$F'(i,s) = \begin{cases} F(\emptyset_{1s}^i) & \text{if } F(\emptyset_{1s}^i) > (F\emptyset_{2s}^i) \\ F(\emptyset_{2s}^i) & \text{otherwise} \end{cases} \quad (2)$$

$F(\emptyset_{1s}^i), F(\emptyset_{2s}^i)$: Fitness value from parents 1 and 2, respectively $F_{max}(i)$: Maximum fitness value of the population$Q_i \bar{F}(i)$: Average fitness value of the population $Q_i$ [20].

2) Adaptive Mutation

$$p_m(i,j) = \begin{cases} p_{m0} \dfrac{(F_{max}(i) - F(i,j))}{(F_{max}(i) - \bar{F}(i))} & \text{if } F'(i,j) > \bar{F}(i) \\ p_{m0} & \text{otherwise} \end{cases} \quad (3)$$

Where $F(i,j)$ is the fitness value of the *j*th chromosome in the population $Q_i$ [20].

## III. ALGORITHM RAGKA

There are some steps RAGKA as shown below.

Start
1. t=0
2. initialize centroid *P(t)*
3. *K*-Means
4. *t=t*+1
5. if termination criterion achieved go to step 11
6. select *P(t)* from *P(t-1)*
7. adaptive crossover *P(t)*
8. adaptive mutation *P(t)*
9. Greedy Selection for fitness step 3 and step 8
10. Go to step 3
11. Output best fitness and stop End

The string representation, initialization population, *K*-Means, selection, adaptive crossover, adaptive mutation, and greedy selection are described as follow. String Representation: Each string is a real number that represents sequential *K*-cluster centers. For an N-dimensional space, the long of a chromosome is *N*K* words, where the first *N* positions (genes) deputize the *N* dimensions of the first cluster center, the next *N* positions represent those of the second cluster center, and so on. As a depiction let us consider the following example. Example: Let *N* = 2 and *K* = 3, i.e., the space (*N*) is two dimensional and the number of clusters (*K*) being considered is three. Then the chromosome 51.6, 72.3, 18.3, 15.7, 29.1, 32.2 deputize the three cluster centers (51.6, 72.3), (18.3, 15.7) and (29.1, 32.2). Note that every real number in the chromosome is an indivisible gene [18] (see Fig. 4).

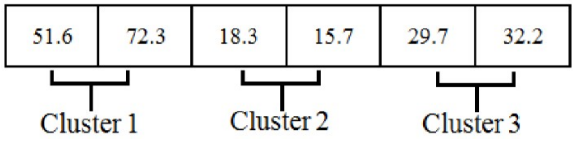


Fig. 4. String Representation

**Initialization Population:** The K cluster centers encoded in every chromosome are initialized to K randomly chosen point from the dataset. This process in repeated for each of the P chromosomes in the population, where P is the size of the population.

***K*-Means:** The clusters are established according to the centers encoded in the chromosome under consideration. This is done by assigning each point $\boldsymbol{x_i, i = 1, 2, \dots, n}$ to cluster $\boldsymbol{C_j, j \in (1, 2, \dots, K)}$ if $\boldsymbol{\|x_i - z_i\| \leq \|x_i - z_p\|, p = 1, 2, \dots, n}$ and $\boldsymbol{j \neq p}$. Afterward, calculate new cluster center $\boldsymbol{z_1^*, z_2^*, \dots, z_K^*, z_i^* = \frac{1}{n_1} \sum_{x_j \in C_i} x_j, i = 1, 2, \dots, K}$ *where* $n_i$ is the number of elements belonging to cluster $\boldsymbol{C_i}$. These $\boldsymbol{z_i^* s}$ now replace the previous $\boldsymbol{z_i^* s}$ in the chromosome.

**Selection:** The selection process selects according to their fitness. This study use Roulette Wheel Selection, which the better chromosomes have more chances to be selected and that will go forward to from the mating pool for the next generation. Weaker individuals are not without a chance.

**Adaptive Crossover:** In crossover, each chromosome exchanges their information with other chromosomes for generating two child chromosomes. This study used single point crossover with adaptive crossover probability, which the crossover probability get from fitness value. For chromosomes of length l, then split into two parts. The portions of the chromosomes lying to the right of the crossover point are replace to produce two offspring.

**Adaptive Mutation:** Each chromosome undergoes mutation with adaptive mutation probability. In mutation, two genes are exchanging information in each chromosome. The genes that mutated are selected by randomize two number which represent genes position in chromosomes.

**Greedy Selection:** The performance of each cluster is compared with the old cluster with the new cluster. In other words, if the new cluster has an equal or better solution than the old cluster, then the new cluster will replace the old cluster. This step will compare between fitness value of K-Means and fitness value faster using Adaptive Genetic Algorithm, the best solution is retained (see Fig. 5).

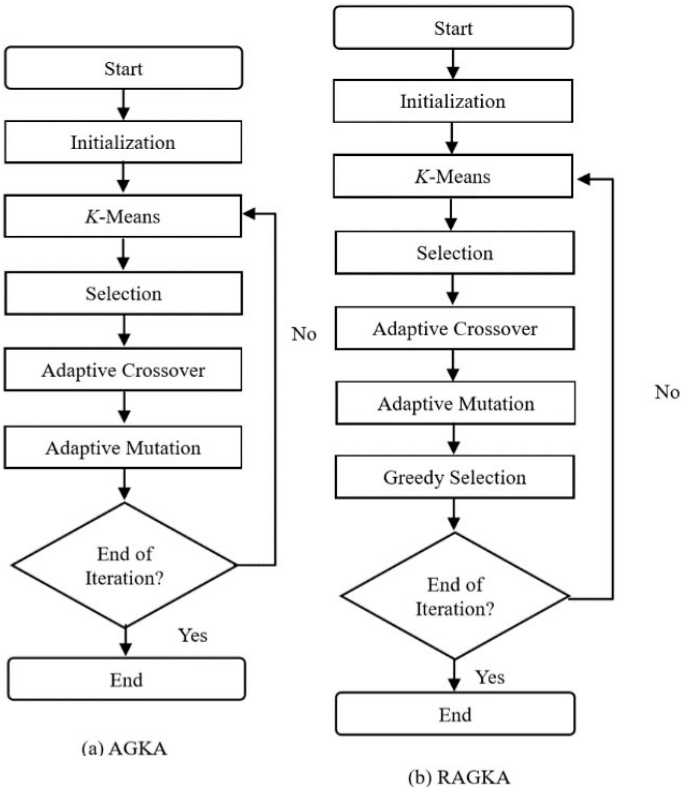


Fig. 5. (a) AGKA and (b) RAGKA Flowchart

## IV. Experimental Results

### A. Dataset and Parameter Settings

TABLE I Dataset

| Dataset | Patterns | Clusters | Attributes |
|---|---|---|---|
| Sonar | 208 | 2 | 2 |
| Haberman | 306 | 2 | 3 |
| Parkinson | 195 | 2 | 22 |
| Iris | 150 | 3 | 4 |
| Wine | 178 | 3 | 13 |
| Glass Identification | 214 | 7 | 9 |
| Yeast | 1484 | 10 | 8 |

Table I shows datasets used for this study. Column "Cluster" indicates number centroid, while column "Attributes" denotes length of data. In this study, the number pattern is 150 to 1484, number clusters is 2 to 10, and number attributes is 2 to 22.

TABLE II Parameter Setting of RAGKA

| Experiment | Parameter |
|---|---|
| Chromosome | 10 |
| $p_c$ | 0.1 |
| $p_m$ | 0.5 |
| Iterations | 100 |
| Experiment | 10 |

In this study, we used 10 chromosomes and 100 iterations, then the experiments run for 10 times. For genetic algorithm, the parameter setting for probability crossover ($p_c$) is 0.1 and probability mutation ($p_m$) is 0.5 as shown in Table II

### B. Analysis Results

The experiment compared GKA, AGKA, and RAGKA are provided from seven datasets (Sonar, Haberman, Parkinson, Iris, Wine, Glass Identification, and Yeast). Then assessed is statistical value how close the value of one iteration with the average of iteration (standard deviation), average accuracy and best fitness. The results of implementation have shown, from seven datasets. Table III shows the performance of comparation of RAGKA, GKA and AGKA algorithms. The result shows, GKA have a standard deviation that is much different from the AGKA and RAGKA as a result of GKA based on a random value of genetic concept itself, but for the standard deviation AGKA improved because of genetic maintain the best fitness value. For the best improvement we use RAGKA, because RAGKA have a good performance than GKA and AGKA for all dataset because the selection of best fitness value which is very tight; but for computation time RAGKA is slower than GKA and AGKA (see Fig. 6).

Computation is slower due to the process therein are counted one by one and compared with previous results on every process after crossover, mutation and performed after genetic processes. As shown in Fig. 6, RAGKA shows good performance and fast solution than GKA and AGKA. For big data, GKA got stuck at sub-optimal solutions in fourth iteration. By using Adaptive Genetic Algorithm, the results get improvement than GKA in thirty-second iteration best fitness began to rise until fifty-third iteration, and then start converging. AGKA get improvement result because probability of each cluster to crossover and mutation depends on their fitness value. Therefore, the best solutions are kept for comparison. Too improve the best solution, RAGKA using Greedy Selection. This step will compare between fitness value of K-Means and fitness value faster using Adaptive Genetic Algorithm, the best solution is retained, and RAGKA start convergent in sixteenth iteration.

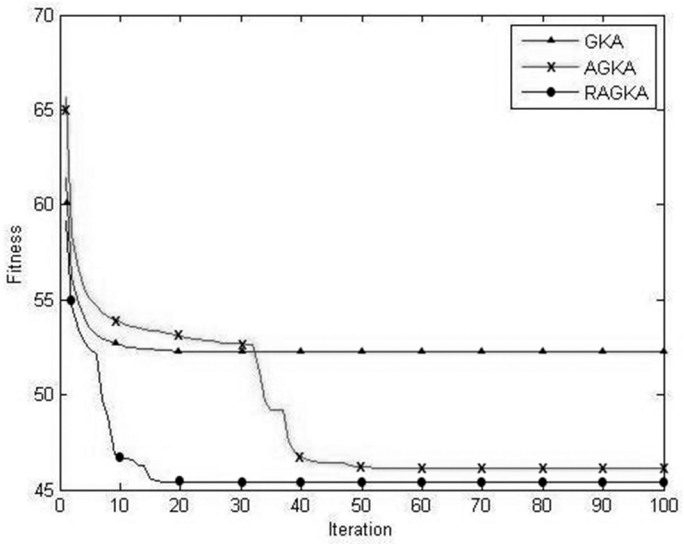


Fig. 6. Diagram Algorithm Analysis for Data Yeast

## V. Conclusion

Inspired by the algorithm to maintain the best fitness value, this paper presents a model robust algorithm for solving the problem of clustering, namely Robust Adaptive Genetic *K*-Means, Algorithm, called RAGKA has been shown in this paper. At first solving the problems of clustering using *K*-Means, but *K*-Means are often trapped local optimum, then *K*-Means combined with GA. When using the *K*-Means with GA still has shortcomings because the optimal results can still be replaced because the probability of crossover and mutation fixed. Therefore, we are trying to use the GA that maintain best fitness value (AGA), and the results are starting to improve, but for some of data is still not optimal, and therefore we try to maintain the best fitness value of the results of the iteration with greedy selection, and the result was increasing rapidly. The results can be seen in Table III, RAGKA to have an accuracy of better than GA with *K*-Means and AGA with *K*-Means, because RAGKA maintain the best fitness value. For further research every best fitness value should be maintained, because RAGKA have optimal value by maintaining the best fitness value.

## References

[1] Guliani, S. (2015). An Enhanced Clustering Algorithm by Comparative Study on K-Means Algorithm, 4(06), 193–196. http://dx.doi.org/10.17577/IJERTV4IS060271

[2] Karaboga, D., & Ozturk, C. (2011). A novel clustering approach: Artificial Bee Colony (ABC) algorithm, 11, 652–657. http://doi.org/10.1016/j.asoc.2009.12.025

[3] Subbalakshmi, C., Rao, P. V., & Mohan, S. K. (2014). Performance Issues on K-Mean Partitioning Clustering Algorithm, 41–51.

[4] Vora, P., & Oza, B. (2013). A Survey on K-mean Clustering and Particle

Swarm Optimization, (3), 24– 26.
[5] Zhang, C., Ouyang, D., & Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems With Applications*, 37(7), 4761–4767. http://doi.org/10.1016/j.eswa.2009.11.003
[6] Cole, R. M. (1998). Clustering with genetic algorithms. University of Western Australia.
[7] Zalik, K. R. (2008). An efficient k-means clustering algorithm, 29, 1385–1391. http://doi.org/10.1016/j.patrec.2008.02.014
[8] Shelokar, P. S., Jayaraman, V. K., & Kulkarni, B. D. (2004). An ant colony approach for clustering, 509(December 2003), 187–195.
[9] Zhao, B. J. (2007, August). An ant colony clustering algorithm. *In Machine Learning and Cybernetics*, 2007 International Conference on (Vol. 7, pp. 3933-3938). IEEE.
[10] Santhanam, T; Padmavathi, M.S. (2015). Application of K-Means and Genetic Algorithms for Dimension Reduction by Integrating SVM for Diabetes Diagnosis. *Procedia Computer Science*. Vol. 47. Page: 76-83. Doi: 10.1016/j.procs.2015.03.185.
[11] Nazeer, K. A. A., & Sebastian, M. P. (2009). Improving the Accuracy and Efficiency of the k-means Clustering Algorithm, I, 1–5.
[12] Arvi, J. K., Anti, P. F. R., & Nevalainen, O. (2003). Self-Adaptive Genetic Algorithm for Clustering, (2), 113–129.
[13] Lin, H., Yang, F., & Kao, Y. (2005). An Efficient GA-based Clustering Technique, 8(2), 113–122.
[14] Ahmad, A. (2007). A k-mean clustering algorithm for mixed numeric and categorical data, 63, 503–527. http://doi.org/10.1016/j.datak.2007.03.016
[15] Chen, Z. (2009). K-means Clustering Algorithm with improved Initial Center. http://doi.org/10.1109/WKDD.2009.210
[16] Dehariya, V. K., Shrivastava, S. K., & Jain, R. C. (2010). Clustering Of Image Data Set Using K-Means And Fuzzy K-Means Algorithms.
[17] June, C., & Singh, R. V. (2011). IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011 Data Clustering with Modified K-means Algorithm, 717–721.
[18] Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique, 33.
[19] Srinivas, M., & Patnaik, L. M. (1994). Adaptive Probabilities of Crossover and Mu tation in Genetic Algorithms, 24(4), 656–667.
[20] Wulandhari, L. A., Wibowo, A., & Desa, M. I. (2015). Condition diagnosis of multiple bearings using adaptive operator probabilities in genetic algorithms and back propagation neural networks. *Neural Computing and Applications*, *26*(1), 57-65.
[21] Cai, Q., Gong, M., Ma, L., Ruan, S., Yuan, F., & Jiao, L. (2015). Greedy discrete particle swarm optimization for large-scale social network clustering. *Information Sciences*, 316, 503-516.
[22] Sheikh, R. H. (2008). Genetic Algorithm Based Clustering: A Survey, 2(6), 314–319. http://doi.org/10.1109/ICETET.2008.48
[23] Mitchell, M. (1996). An Introduction to Genetic Algorithms. Page: 124-130. ISBN: 0-262-133164.
[24] Huang, C., & Wang, C. (2006). A GA-based feature selection and parameters optimization for support vector machines, 31, 231–240. http://doi.org/10.1016/j.eswa.2005.09.024

**TABLE III PERFORMANCE OF RAGKA**

| Dataset | | GKA | AGKA | RAGKA |
|---|---|---|---|---|
| Sonar | Best Fitness | 280.5696 | 280.5340 | **280.5340** |
| | Average Fitness | 280.6561 | 280.5718 | **280.5542** |
| | Standard Deviation | 0.1727 | 0.1038 | **0.0178** |
| | Average Time | **2.0887** | 2.0972 | 2.1198 |
| Haberman | Best Fitness | 30507.0208 | 30507.0208 | **30507.0208** |
| | Average Fitness | 30549.1299 | 30508.1490 | **30507.1219** |
| | Standard Deviation | 48.1934 | 2.3786 | **0.1306** |
| | Average Time | **2.8601** | 2.8730 | 2.9498 |
| Parkinson | Best Fitness | 1165833.2846 | 1165833.2846 | **1165833.2846** |
| | Average Fitness | 1327184.9552 | 1273944.6981 | **1201307.1043** |
| | Standard Deviation | 48.1934 | 2.3786 | **0.1306** |
| | Average Time | **1.9560** | 1.9679 | 1.9729 |
| Iris | Best Fitness | 78.9408 | 78.9408 | **78.9408** |
| | Average Fitness | 78.9425 | 78.9421 | **78.9408** |
| | Standard Deviation | 0.0022 | 0.0020 | **0.0000** |
| | Average Time | **1.8189** | 1.8313 | 1.8857 |
| Wine | Best Fitness | 2370689.6868 | 2370689.6868 | **2370689.6868** |
| | Average Fitness | 2396976.2513 | 2396143.0332 | **2396143.0332** |
| | Standard Deviation | 83125.4159 | 80490.5487 | **80490.5487** |
| | Average Time | **2.1282** | 2.1704 | 2.1964 |
| Glass Identification | Best Fitness | 315.4894 | 292.3537 | **292.2628** |
| | Average Fitness | 387.6908 | 354.5441 | **303.9549** |
| | Standard Deviation | 88.2767 | 77.3931 | **12.1029** |
| | Average Time | **4.2111** | 4.2696 | 4.2953 |
| Yeast | Best Fitness | 45.7754 | 45.4305 | **45.3809** |
| | Average Fitness | 49.2993 | 46.4659 | **45.7348** |
| | Standard Deviation | 3.4327 | 0.6798 | **0.1536** |
| | Average Time | **37.9817** | 39.0110 | 39.4703 |