RESEARCH

An Genetic Algorithm Approach for Profiling Computational Performance Measures

Matheus S Lima

Correspondence: matheusslima@yahoo.com.br
Department of Computer Science, Federal University of Sao Carlos, Sao Carlos, Brazil
Full list of author information is available at the end of the article

Abstract

This paper present an Genetic Algorithm(GA) approach for clustering data metric of computational performance measures collected from vmstat and sar tools. The proposed work models the genes, chromosomes, species and environment based on the dataset and presents an algorithm to analyze patterns and classify the records. The proposed method submits the performance information to an N-Dimensional Histogram in order to obtain the distribution of data that is used as input to the cluster initialization. The individual from each specie undergo successive crossover, mutation and selection operations to improve and evolve the initial population to a given environment state. The fitness-function is determined by the N-Dimensional Euclidean distance. The selection method is based on the Roulette-Wheel Selection, Elitist Selection and Truncation Selection. The result's presented were obtained from seven test scenarios.

Keywords: Genetic Algorithm; Performance Analysis; Clustering; Profiling

Introduction

Performance

The execution of any software requires different types and amounts of hardware resources and this consumption strongly varies and depends of many factors like the nature of the application(IO-bound, CPU-bound, Memory-bound or Network-bound), the hardware capabilities itself and the user's interaction with it. Then performance analysis of the Operating System and the Applications running on it is of crucial importance since it directly impacts the services response and therefore the user's experience. The usage profile study allows for the optimization of the Operating System, Hardware and Software Applications due to custom changes made in any of them, this is know as Performance Tunning. The Performance Tunning results in an important improvement that allows a system to accepts an higher load, since it eliminates or reduces the impact of a part of the system which is responsible for the overall decreasing performance, know as bottleneck. There is basically four major profiles that can affect system performance(Corrigan 1996):

- CPU-bound – Occurs when OS and/or Software Applications are making too many request to the CPU or there isn't adequate amount of CPU available.
- Memory-bound – Occurs when there isn't enough memory available forcing the OS to do excessive paging and swap.
- Disk-bound – Occurs when the IO rate is above the capacity of the disk.

- Network-bound – Occurs when the traffic is too high or too many network collisions happen.

One of the most common ways to improve the Linux based OS distributions are the modifications of the Kernel parameters. It allows for example the change in the memory swappiness behavior, memory page size, etc(Ciliendo and Kunimasa 2007). Changes in the hardware can also improve performance, since applications like large databases are usually IO intensive. Solid-state Drives(SSD) in general have a better performance then Hard Disk Drivers(HDDs) for this scenario(Lee et al. 2008). Many tools are available for Unix based Operating Systems like Linux that allows the performance's inspection such as vmstat, iostat, top, sar, ps, etc ... In this paper we focus on vmstat and sar. Vmstat offers a variety of data from CPU, Memory, Swap, IO, System and Processes(Ware and Frédérick 2014)

Procs

r: The number of processes waiting for run time.

b: The number of processes in uninterruptible sleep.

Memory

swpd: the amount of virtual memory used.

free: the amount of idle memory.

buff: the amount of memory used as buffers.

cache: the amount of memory used as cache.

inact: the amount of inactive memory.

active: the amount of active memory.

Swap

si: Amount of memory swapped in from disk (/s).

so: Amount of memory swapped to disk (/s).

IO

bi: Blocks received from a block device (blocks/s).

bo: Blocks sent to a block device (blocks/s).

System

in: The number of interrupts per second, including the clock.

cs: The number of context switches per second.

CPU (These are percentages of total CPU time.)

us: Time spent running non-kernel code. (user time, including nice time)

sy: Time spent running kernel code. (system time)

id: Time spent idle.

wa: Time spent waiting for IO.

st: Time stolen from a virtual machine.

Sar also offers many system activity information. In this paper we use only the network traffic information(Godard 2014):

Network

rxkB/s: Total number of kilobytes received per second.

txkB/s: Total number of kilobytes transmitted per second.

Genetic Algorithm

Genetic Algorithms(GA) are based on the theory of natural selection and genetics(Mitchell 1998). It's applied in optimization problems in order to find better solutions using an heuristic approach. It relies on the concept of *Heredity* from which children receive genetic characteristics of its parents, that allowed then to survive and reproduce in the environment; *Variation* which accounts for the genetic variability among populations and are the decision factor if an given individual will survive/reproduce or not in the environment. This concept is constituted by two process: Crossover (Responsible for the exchange between genetic materials from the parents) and Mutation(Responsible for small random changes in the children genetic material); *Selection* is the mechanism in which an individual from the population is evaluated against the environment. If a given individual is more fit than another one in an specific environment, it has more chances that his offspring's will survive and proliferate. GA applies this ideas in the following order:

Initiate Population

In this step an initial population is generated. Different approaches exist to create them and the most common is to randomly create several individuals. Other would be the pre-analysis of the given dataset in order to create individuals that are more similar to the sample. This has the advantage of reduce the amount of interactions required by the algorithm to produce a valid solution and also improving their quality. Therefore this is the option chose in this work.

Selection

This is represented by an Fitness Function which is used as an measure of how fit the individual is against the problem. If the solution represented by the individual is good it has more chances to survive and reproduce, spreading its genes to future generations. Each different problem requires an unique Fitness Function since it evaluates an very specific type of data. For example, numeric data have different properties than categorical data and therefore require an unique approach(Roy and Sharma 2010).

Crossover

This step exchanges information between two parent solutions that will be inherited by the child's.

Mutation

Introduces small changes in the child's genes, that may produces evolutionary advantages.

Termination

At this moment, the solutions generated are checked against some termination criteria and the algorithm stops. This may considers how good the solutions are or if the algorithm has executed at an fixed amount of time, etc.

Cluster Analysis and Profiling

Performance analysis tools like vmstat and sar produces an snapshot of the computational resources been used at a given time. This output can be stored using tools like ctracker(Lima 2014) and them be further analyzed to find patterns. The profiles that are founded based on this patterns allows for the detection of bottlenecks, shown when a unexpected high usage of a resource happens and detect attack/intrusion attempts like those using Denial-of-Service techniques. This information is very important for System Administrators and IT Managers, that constantly need to guarantee the Quality of Service (QoS). Since a lot of data can be generated every second, the amount of record's obtained grows quickly and the activity to manage the profiles and identify clusters within those data by an human operator becomes an difficult task.

In order to address this issue an Genetic Algorithm(GA) approach is proposed by this paper. The GA method generate solutions to this Cluster Analysis Problem(Maulik and Bandyopadhyay 2000) which consist in group a set of data that are similar to each other following an given computational performance pattern. Others approaches exist to this type of problem and one example is the K-Means algorithm that can also generate valid solutions(MacQueen, 1967).

Materials & Methods

The Modeling subsection shown how the performance data collected is interpreted as a set of properties used by the GA to evolves and adapts over time. For the Pre-Processing subsection, we shown the required data treatment that improves the understanding of the data distribution. In the Algorithm subsection, we discuss how the GA steps are applied. Finally, at the Results subsection the output for all test scenarios are presented.

Modeling

Genes

The CPU values of Idle, Waiting, Non-Kernel code and Kernel code, for example are, each one, a gene as shown in Figure 1. The same applies for other values obtained from vmstat and sar such as Memory, Swap, System, Processes, Kernel, Network, etc ...

Chromosome

The values of the vmstat and sar at a given time represent a Chromosome which consist of a group of all genes. They represent individuals of a population as shown in Figure 2.

Figure 1 Gene Gene representation based on vmstat and sar values.

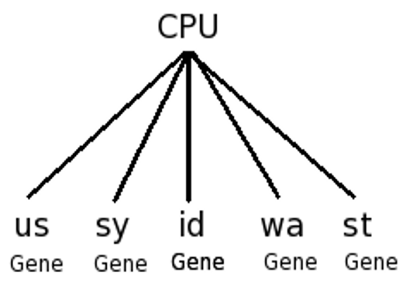
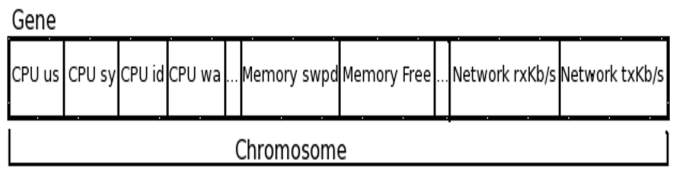


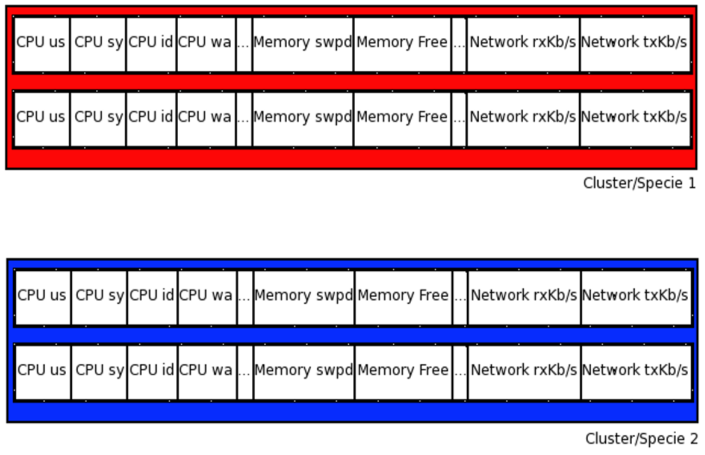
Figure 2 Chromosome Chromosome representation as the set of genes.



Cluster/Specie

A Cluster or Specie in this work is represented by an group of individuals that have very unique characteristics and can reproduce only between the same group and therefore there is no exchange of the genetic code outside this group as shown in Figure 3.

Figure 3 Cluster/Specie Cluster/Specie design with an set of individuals.



Environment
The environment is defined as the dataset which stores all the vmstat and sar output obtained from an given time interval. An environment state is defined as a single record from this dataset as shown in Figure 4.

Pre-Processing

In order to reduce the amount of iterations required by the algorithm to produce good solutions and also improve the overall solution quality, we used an N-Dimensional Histogram to obtain the distribution of values across the dataset, that are then used as an range when randomly generating the values of the Initial Population. This approach allows for gene values that are more close to the values within the dataset.

There is also the possibility to randomly initiate the gene values without any range, but this has downside of increase the amount of iterations required to the algorithm to converge and also resulting in quality lost.

Algorithm

Initiate Population
Each specie/cluster has an group of individuals that are initialized using the range obtained from the N-Dimensional Histogram. The amount of individuals(population size) are fixed as well as the amount of species(cluster size) that will exist and compete. This implies that from all the bins from the Histogram, just a few will be used to initiate the species and therefore it's individuals.

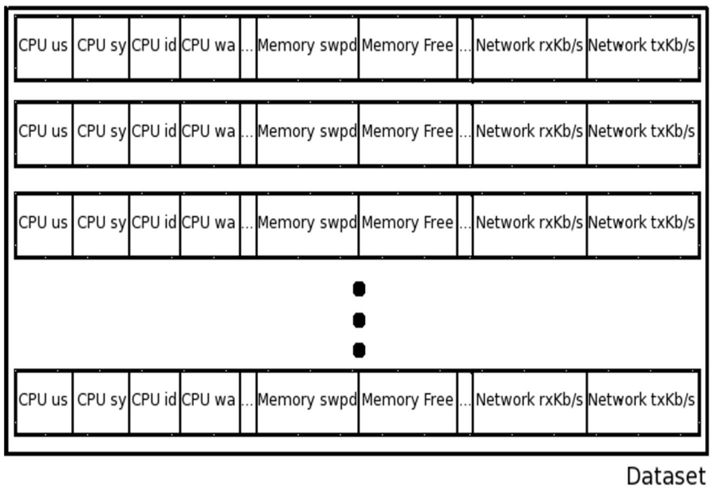


Figure 4 Enviroment vmstat and sar output stored in the dataset.

Crossover

The concept applied in this paper is based in the phenomenon know as heterogamy, in which some species alternates between sexual and asexual strategies of reproduction depending on environmental conditions(Cole and Sheath 1990). Our approach generalizes this concept and the parent can generate children from both ways within the same generation. The genetic code is exchanged by the sexual reproduction, as shown in Figure 5 and clones are created by asexual reproduction. The Crossover operation creates an amount of individuals equals to the square root of the population size for each cluster.

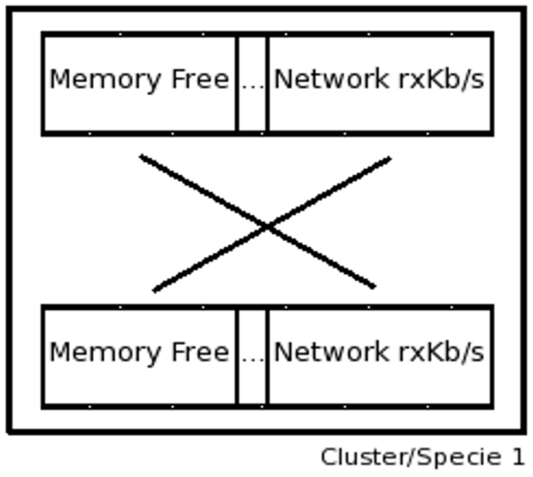
Mutation

Each gene has a fixed probability of suffer mutation, this is know as mutation rate(Thierens and Dirk 2002). If the mutation happens, the gene can be shifted and be added or subtracted by an random value chose from an fixed range(Shift Range). The choice of both values, Shift Range and Mutation Rate, need to be choose carefully. An high value may cause severe damage to the individual genetic code, damaging the quality of the solution and therefore causing the individual to not survive and proliferate.

Fitness Function and Selection

Each children from each specie is compared against the environment state at a given time. The Fitness-Score is calculated as the N-Dimensional Euclidean distance between the individual and the environment. The individuals that are close to the

Figure 5 Crossover operation Crosover swapping the genes of two parents



environment are the ones with better chances of survive. The distance values are sorted by distance, with the closest ones having the best probability to proliferate.

In order to choose the individuals that are most fit, an fixed percentage of the children sample are chosen as the closes N-Dimensional Euclidean distance to create the Mating Pool. This process of reducing the amount of possible solutions is know as Truncation Selection.

The probability of each cluster/specie is calculated by the division of the amount of individuals from the Mating Pool that exist from each cluster, by the total amount of individuals in the Mating Pool.

The Selection method used by this work is know as Roulette Whell (Lipowski and Lipowska 2001). The probabilities for each cluster is the chance of the cluster been chosen in a random selection. This method guarantee that clusters with higher probabilities will have better chances that his individuals will reproduce and also don't completely eliminate the chance of clusters with lower probability been chose . This is important since they also can have individuals with genes that are a good fit to the environment(Shiffman 2012).

The cluster chosen by the Roullete Whell Method is updated with the winner children set. At this moment an fixed amount of the closest children are choose from the winner cluster, following the population size value. The children of the other parents of the loser clusters/specie are discarded and the parent's remain, as the individuals of the loser cluster/species. The importance of keep the parent's value for the the species that looses the competition is because they can be selected in the future, or were selected in the past, for an given environment as the most fitted. This avoids the Initialization of the cluster again, since the current values can still have an evolutionary advantage for a different environment set of values. This is know as Elitism Selection (Baluja et al. 1995).

If the winner specie at this iteration is the same as the previous iteration for a given environment state, then a new environment is picked and Crossover and Mutation undergo. If the winner specie is different from the previous one, then the current environment is maintained and new child's will try to find an better solution.

The algorithm stops when all environment states were evaluated and classified.

The proposed method is shown in Algorithm 1.

Results

The study presented by this work evaluate a dataset of 14465 performance records, obtained by the vmstat and sar tools. In order to plot the cluster's classification output in a 2D chart, we used a simplified chromosome with only two genes: CPU Idle(id) and Memory free. We performed seven test's scenarios. For scenarios 1, 2,3, 4, 5 and 7 the 2-Dimensional Histogram with the highest values are shown in Table 1:

Table 1 Higher Histogram bins.

CPU Idle Interval(%)	Memory free Interval(Mb)	Number of records
80.0 - 100.0	736.600 - 909.0	12640
0.0 - 20.0	47.0 - 219.400	1544
0.0 - 20.0	219.400 - 391.800	199
60.0 - 80.0	736.600 - 909.0	36

For scenario 6, the 2-Dimensional Histogram values are shown in Table 2:

Table 2 Higher Histogram bins.

CPU Idle Interval(%)	Memory free Interval(Mb)	Number of records
90.0 - 100.0	736.600 - 822.799	6323
90.0 - 100.0	822.799 - 909.0	5822
0.0 - 10.0	47.0 - 133.199	1439
80.0 - 90.0	822.799 - 909.0	468

The results are shown in Table 3:

Table 3 Higher Histogram bins.

	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7
Number of bins	5	5	5	5	5	**10**	5
Number of Cluster/Specie	4	4	4	**2**	4	4	4
Population size per Cluster/Specie	4	4	**10**	4	4	4	4
Sample % for Truncate Selection	20%	20%	20%	20%	20%	20%	**5%**
Shift Range	[0-2]	[0-2]	[0-2]	[0-2]	**[0-80]**	[0-2]	[0-2]
Mutation Rate	10%	**80%**	10%	10%	10%	10%	10%
Number of Iterations	20421	**25667**	**25212**	**16224**	**25213**	**26976**	**22545**
Children per generation	64	64	400	32	64	64	64
Children per Cluster/Specie	16	16	100	16	16	16	16
Number by Cluster/Species							
k0	10548	5941	8660	12706	6507	4737	6459
k1	1583	1474	1544	1759	2213	4925	1527
k2	186	292	223	X	3839	1757	240
k3	2148	6758	4038	X	1906	3046	6239

The total population for all iterations is shown in Table 4:

Table 4 Population Size

Scenario	Population Size
4	519168
1	1306944
7	1442880
5	1613632
2	1642688
6	1726464
3	10084800

Discussion

Scenario 1 is our Control Group and will be used as model against the other scenarios. It shows the best balance between all variables used by our test cases and have resulted in an better classification quality.

In Scenario 1 even tough Cluster/Specie 3 has it's Initial Population created from the histogram bin with the small amount of samples(36), it manages to classify 2148 records. The reason for this is the similarity between the 'Network free' and the 'CPU Idle' genes for Cluster/Specie 0 and Cluster/Specie 1. After several generations and iterations, they converge to have closes values and therefore competing more aggressively to the environment. Cluster/Specie 1 and Cluster/Specie 2 indeed are initialized using the same histogram bin range from 'CPU Idle', but since 'Memory Network' ranges are not sufficient close for both of them, they do not converge like Cluster/Specie 0 and Cluster/Specie 3.

Table 1 shows that the increase to 80% for the Mutation Rate from Scenario 2 results in an increase number of iterations required to the algorithm to finish. This happens because since mutations happens more often, the algorithm needs more iterations to converge. This increases the benefits to Cluster/Specie 3, creating an evolutionary advantage against the other species, although it still compete aggressively with Cluster/Specie 0.

Scenario 3 increases the population size for each Cluster/Specie to 10, creating 100 offspring's per generation with an total of 400 children per iteration against 64 for Scenario 1. This increases the number of required iterations and calculations to converge as well. The classification for this scenario is similar to the behavior show for Scenario 2, although it classify more records as Cluster/Specie 0, similarly to Scenario 1. Since there is an increase in the number of iterations, this scenario tends to approximate Cluster/Specie 0 and Cluster/Specie 1 to similar values.

Scenario 4 requires less iterations than any other scenario. The reason for this is that once there is less Cluster/Species there will be less children to be evaluated taking less time to converge. Although there is less children for each generation, the amount of children that were selected is enough to have a good genetic variability and don't excessive cause overlap in the result.

Scenario 5 shows that the increase of the range interval for the Shift Range to [0-80] severely damages the genes and consequently the chromosome. Therefore, the classification shows poor results. This is because the mutations don't represent any advantage, causing the Clusters/Species to excessively overlap.

Scenario 6 shows that an increase in the number of bins(and it's consequent small interval's) damages the final result causing excessive cluster overlap, if the number of clusters don't increase as well. The reason for this is that since there is not a enough number of clusters the Initial Population is initiated with an range that is not similar enough to the dataset. This also increases the number of iterations.

Scenario 7 decreases the Sample Percentage For Truncate Selection and this increases the number of required iterations. The reason for this is since less children are selected to the Roulette-Wheel method(Even less the Scenario 4), the competition between Cluster/Specie 0 and Cluster/Specie 1 becomes more aggressive, requiring more iterations and more offspring's to occur in order to decide the winner specie. This is show by the amount of records classified as Cluster/Specie 0 and Cluster/Specie 1 been similar. Also the genetic variability is apparently not damaged by the decrease in the Sample Percentage For Truncate Selection variable.

Figures 6, 7, 8 and 12 shown that Species 1 and 2 represents records with CPU Idle at 0% of the time and also that free Memory is low or, for Specie 1, zero. This profile indicates that the computation resources available are being exhausted and optimization may be required. Although, Specie 0 and 3 shows that, in general, the machine spend most of it's time with CPU Idle around 80% to 100% and more than 600Mb of free Memory. Figure 9 shown a similar result, polarizing the output between two patterns with an high and low resource consumption. Figure 10 and 11 excessively overlap the cluster/species and therefore damaging the results.

Population Size and Execution time

The total population size is calculated as the value of the *Number of Iterations* multiplied by the value of the *Children per Generation*. This size is directly proportional to *Number of Cluster/Specie* and to the *Population Size per Cluster/Specie*. Once this values changes, the amount of children created for each generation also changes and therefore, impacting in the time complexity of the algorithm. Crossover operation, sort function and Euclidean Distance calculation heavily affects the time required to complete those steps, as show in Table 4.

Scenario 4 with only 2 Clusters/Species produces 519168 children, finishing it's execution more quickly then any other scenario, this occurs because there is less solutions to be evaluated. Scenario 3 behaves in the opposite way with the biggest *Population Size per Cluster/Specie* and consequently having by far(10084800 children) the slowest execution. Although Scenario 5 and Scenario 6 generates cluster solutions with excessive overlap, as shown in Figure 10 and Figure 11, they have an close total amount of children to Scenario 2, which produces an good profiling as shown in Figure 7. Considering all the Scenarios presented by this paper, Scenario 1 require less time and is still able to profile the dataset with an good quality and variability. Although Scenario 4 executes more quickly than Scenario 1, it has less profiles to detect and ,as a consequence, deriving less information.

This demonstrate that the number of iterations alone is not an indicator of the quality of the solution and the input values associated with it can then increase or decrease the required execution time and output quality.

Conclusions

In this paper, a method based on Genetic Algorithm to classify and detect cluster's of computational performance profiles is proposed. The results shown in Figure 6, Figure 7, Figure 8, Figure 9 and Figure 12 obtained from the test Scenarios 1, 2, 3, 4 and 7 respectively, demonstrates the effectiveness of the profiling and results in an proper segmentation for the dataset. To perform the clustering, an Modeling for the genes, chromosome, clusters/species are designed and the dataset is pre-processed using an N-Dimensional Histogram in order to reduce the amount of iterations required by the algorithm and also to enhance the quality of the clustering output. The algorithm creates an Initial Population based on the ranges and data distribution from histogram and an series of crossover, mutation and selections operations are performed in order to assign to each record the proper cluster/specie. The quality of the method proposed strongly relies on the input values of the variables. Figure 10 and Figure 11 obtained from the Scenarios 5 and 6 respectively, shown that this can damage the results and are an direct consequence of the values choose to the Shift Range and the number of Bins. Shift Range particularly is very important since it affects how much the genetic code chance in the mutation. If the mutation is severe, instead of helping it's survival it can lead to his extinction since he is not fit to the environment. Scenario 3 shown that the increase in the population size not necessarily will result in an increase in the quality of the clustering, as show in Figure 8. Also, Scenario 2 show that an increase in the Mutation Rate to 80% of the genes alone is not enough to cause damage to the classification. Scenario 4 shown that decreasing the amount of cluster directly impacts the number of iterations and this not necessarily affects genetic variably and the quality of the output result. An similar statement can be made for Scenario 7 but in this case an increase in competition between the species is observed. It was also observed that overtime individuals from the each population that have similar characteristics tends to became more similar between each other since they compete more aggressively to be more adapted to the environment. We have also demonstrate that the execution time required by the method presented in this work is directly proportional to the population size for each cluster/specie. Scenarios 3 and Scenario 4 shown that if the amount of children per iteration is high or low, the algorithm will require more or less operations to end, respectively. The results shown that the information obtained by this method can improve the understanding of the computational performance and therefore helping the user to detect abnormalities or unexpected events affecting the system. This information can also help the user to better adapt the system to a specific workload and eliminate bottlenecks.

Future Work

Future work include implementing methods to optimize the choice of the variables input and therefore obtaining an result that requires less iterations and an better classification of the records. The algorithm proposed by this paper will also be modified to use the dataset as an Training Set to eliminate the histogram calculation and different types of data originated by other types of metrics sources will be explored.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
MSL designed and implemented the GA Algorithm showed in this paper as well performed the tests and the data analysis of the sample.

References

Corrigan, P. (1996). ORACLE performance tuning, 2nd edn. " O'Reilly Media, Inc.".

Ciliendo, E., & Kunimasa, T. (2007). Linux performance and tuning guidelines. IBM, International Technical Support Organization.

Lee, S. W., Moon, B., Park, C., Kim, J. M., & Kim, S. W. (2008, June). A case for flash memory ssd in enterprise database applications. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 1075-1086). ACM.

Frédérick F, Ware H(2014). vmstat. In: Section 8. Linux man page. Available via die.net. http://linux.die.net/man/8/vmstat. Acessed 12 Nov 2014.

Godard S(2014). sar. In: Section 1. Linux man page. Available via die.net. http://linux.die.net/man/1/sar. Acessed 12 Nov 2014.

Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.

Roy, D. K., & Sharma, L. K. (2010). Genetic k-Means clustering algorithm for mixed numeric and categorical data sets. International Journal of Artificial Intelligence & Applications, 1(2), 23-28.

Lima, M (2014). Cloud Tracker(ctracker): Cloud Monitoring Tool. http://cloudtracker.org/. Accessed 12 Nov 2014.

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. Pattern recognition, 33(9), 1455-1465.

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

Cole, K. M., & Sheath, R. G. (Eds.). (1990). Biology of the red algae. Cambridge University Press.

Thierens, D. (2002, May). Adaptive mutation rate control schemes in genetic algorithms. In Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on (Vol. 1, pp. 980-985). IEEE.

Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. Physica A: Statistical Mechanics and its Applications, 391(6), 2193-2196.

Shiffman, D. (2012). The nature of code:[simulating natural systems with processing]. Selbstverl..

Baluja, S., & Caruana, R. (1995, May). Removing the genetics from the standard genetic algorithm. In ICML (pp. 38-46).

Algorithm 1 Genetic Algorithm to Clustering Perfomance Measures

```
$Dataset$ ⇐getDataSource()
$BinSize$ ⇐5
$ClusterSize$ ⇐4
$PopulationSize$ ⇐4
$SamplePercentageTruncateSelection$ ⇐20%
$ShiftRange$ ⇐[0-2]
$MutationRate$ ⇐10%
$ClusterEnvArray$ ⇐setZerosToArray(Dataset.getLength())
$IndexCluster$ ⇐0
$BinRangeArray$ ⇐sort(getHistogram(BinSize, Dataset))
$HigherBinIntervalRangeArray$ ⇐getHighBins(ClusterSize, BinRangeArray)
$PopulationArray$ ⇐InitiatePopulation(HigherBinIntervalRangeArray, PopulationSize, ClusterSize)

for $Environment$ in $Dataset$ do

    $Start$ ⇐True
    $NewState$ ⇐0
    $OldState$ ⇐0

    while ($NewState$ ≠OldState) or ($Start$ ==True) do

        $Start$ ⇐false
        $ChildArray$ ⇐Crossover(Population)
        $ChildArray$ ⇐Mutate(ChildArray, ShiftRange, MutationRate)
        $OldState$ ⇐ClusterEnvArray[IndexCluster]

        {Selection Operations}

        $MatingPoolArray$ ⇐sort(EuclidianDistance(ChildArray,Environment)

        {Truncation Selection - Preserve the 20% of the children population that are most fit to the
        Environment State.}
        $MatingPoolArray$ ⇐Truncate(SamplePercentageTruncateSelection, MatingPoolArray)

        $SpecieProbabilitiesArray$ ⇐CalculateClusterProbabilities(MatingPoolArray)
        $WinnerSpecie$ ⇐Roulette-Whell(SpecieProbabilitiesArray)

        for $i = 0$ to $ClusterSize$ do

            {Elitism Selection - Preserve population from specie that loose.}
            {Update the winner obtained by the Roulette Wheel Selection.}

            if $WinnerSpecie$ ==i then

                $PopulationArray[WinnerSpecie]$ ⇐getIndividuals(MatingPoolArray[WinnerSpecie],
                PopulationSize)

            end if

        end for

        $ClusterEnvArray[IndexCluster]$ ⇐WinnerSpecie
        $NewState$ ⇐WinnerSpecie

    end while

    $IndexCluster$ ⇐IndexCluster+1

end for
```

Figure 6 **Scenario 1** Clustering output

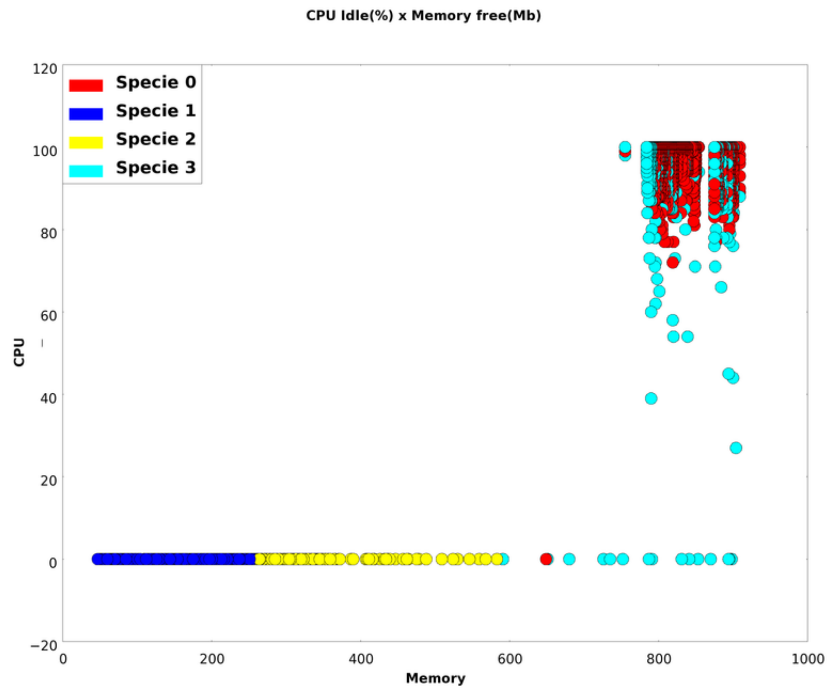


Figure 7 **Scenario 2** Clustering output

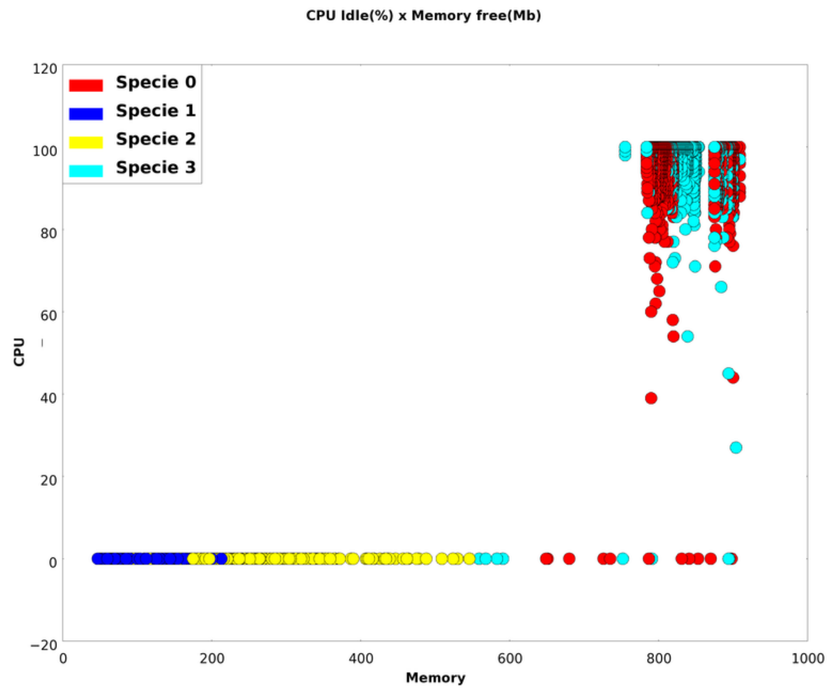


Figure 8 Scenario 3 Clustering output

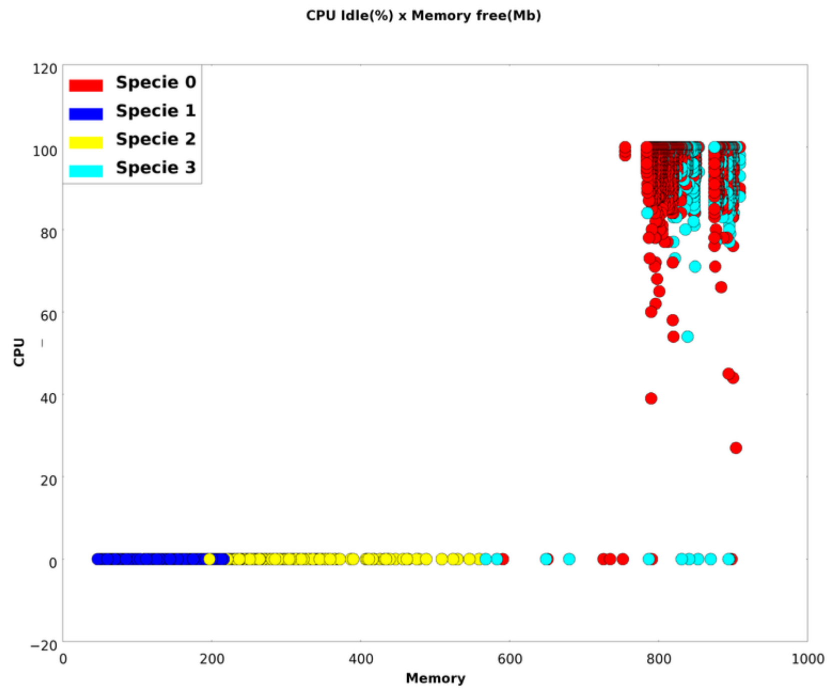


Figure 9 Scenario 4 Clustering output

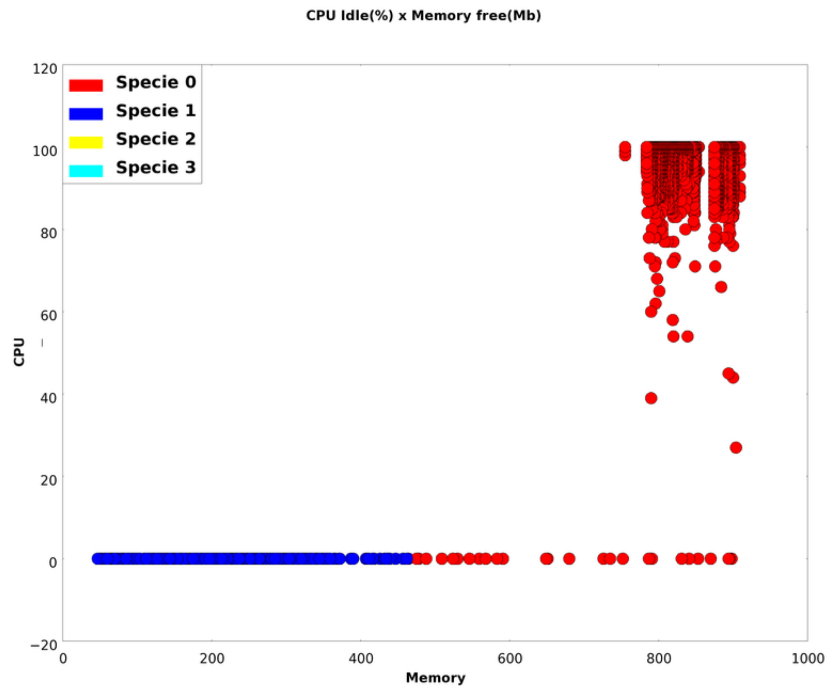


Figure 10 Scenario 5 Clustering output

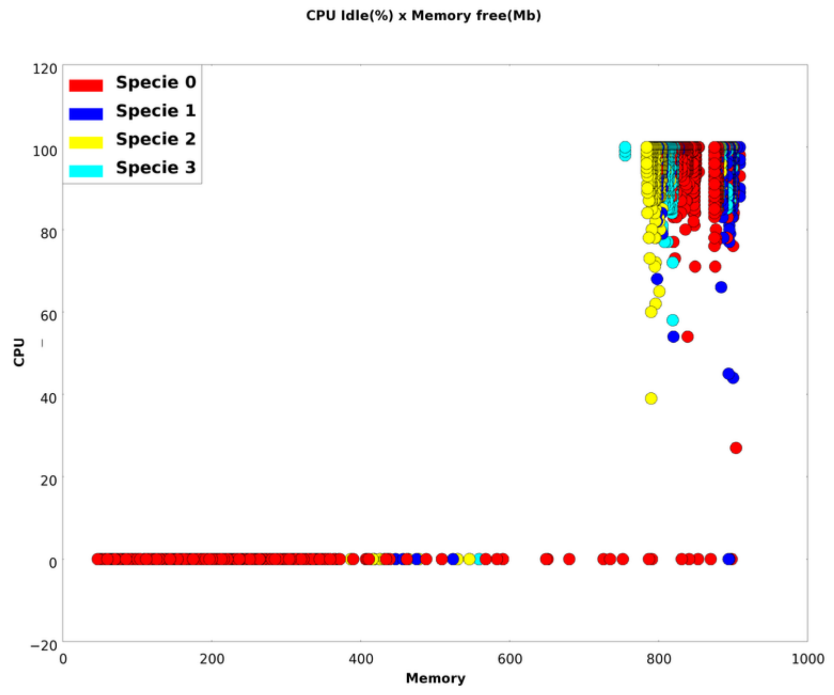


Figure 11 Scenario 6 Clustering output

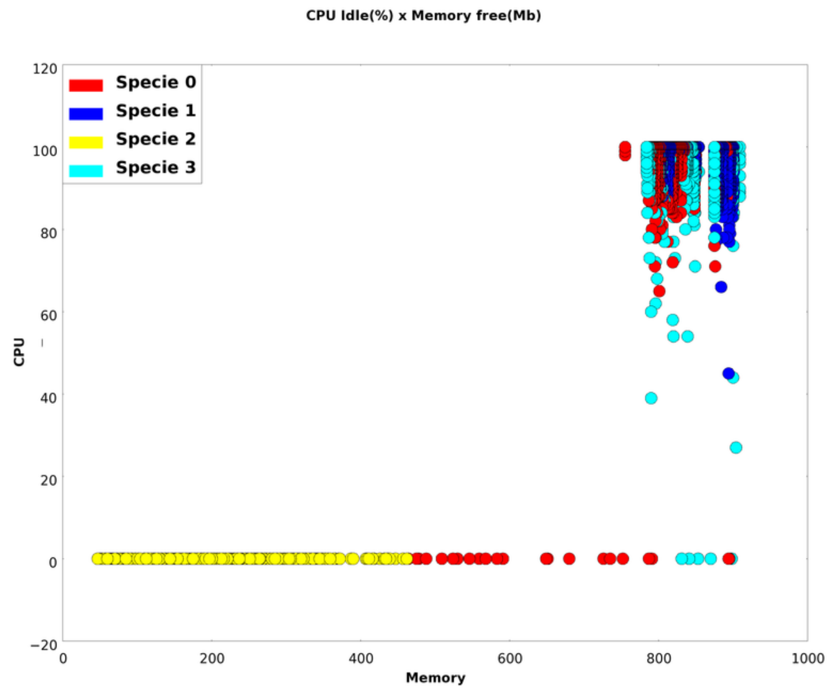
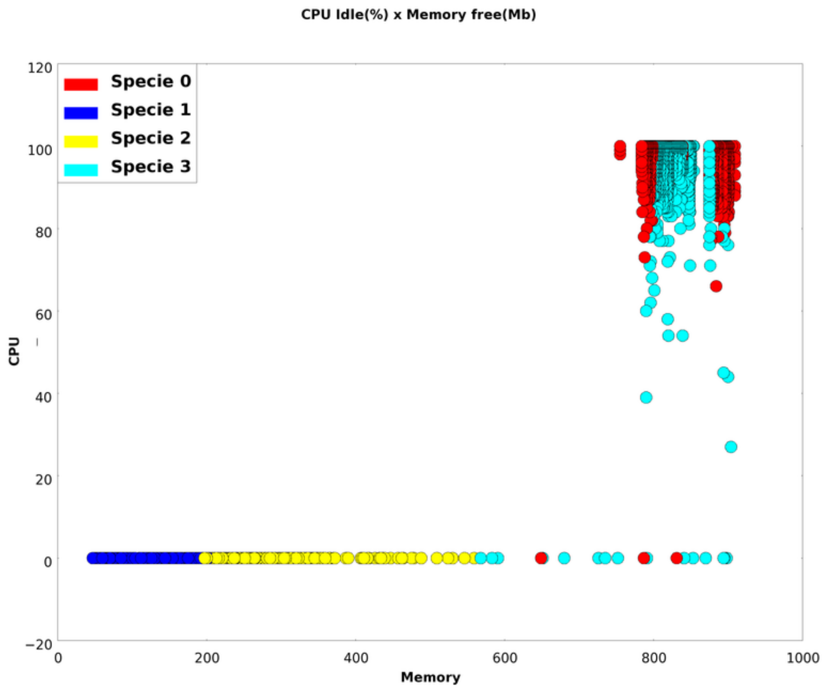


Figure 12 Scenario 7 Clustering output


CPU Idle(%) x Memory free(Mb)
Specie 0
Specie 1
Specie 2
Specie 3
CPU
Memory