# Comparison of Strategies Based on Evolutionary Computation for the Design of Similarity Functions

A. Fornells Herrera [a] J. Camps Dausà [a] E. Golobardes i Ribé [a] J.M. Garrell i Guiu [a]
[a] *Researching Group in Intelligence Systems - http://www.salleURL.edu/GRSI*
*Enginyeria i Arquitectura La Salle, Universitat Ramon Llull*
*Quatre Camins 2, 08022 Barcelona.*
*E-mail: {afornells, joanc, elisabet, josepmg}@salleURL.edu*

**Abstract**. One of the main keys in case-based reasoning system is the retrieval phase, where the most similar cases are retrieved by means of a similarity function. According to the problem, the similarity function must be selected and adapted depending on the characteristics and properties of the problem's domain. The goal of this article is to present a platform called BRAIN, which incorporates strategies based on different evolutionary approaches to design similarity functions ad hoc for a domain to be used in a case-based reasoning system. The strategies are based on Genetic Programming and Grammar Evolution approaches. Both are applied to different data sets to study the influence of their characteristic in the accuracy rate and in the execution time.

**Keywords.** Reasoning Models, Machine Learning, Similarity Function, Case Base Reasoning, Genetic Programmming, Grammar Evolution

## 1. Introduction

Case-Based Reasoning (CBR) [1] tries to solve new problems using others previously solved. When talking about improving accuracy rates, traditionally the focus lands on the similarity function. It allows to compute the degree of similarity between the previously solved cases and the new case, and using the most similar cases CBR proposes a possible solution. Therefore, the accuracy rate is highly influenced by the similarity function. General purpose similarity functions are normally adapted and adjusted according to the problem's domain, but it is a complex task due to the complexity of the real domains. If the similarity function can not model the problem's domain, the accuracy rates will usually not be the desirable. Also, adding specific knowledge to these functions is very complex.

Genetic Programming (GP) [2] is a well-known variant of Evolutionary Computation (EC) designed to find programs or optimize functions. It seems reasonable to think that using CBR and GP together in hybrid system it is possible to find a similarity function ad hoc for a domain. The hybrid can use the GP to generate functions, and the CBR to evaluate their reliability. However, the searching space is infinity and it could be impossible to find the desirable solution. For this reason, specific knowledge of the domain

can be added to GP's individuals in order to reduce it, but it is very complex to add them because GP does not allow it in a natural way. On the other hand, Grammar Evolution (GE) [3] allows finding programs like GP with the ability of using a Backus Naur Form (BNF) grammar to lead the searching process. Therefore, it is easy to add knowledge because it only implies modifying the grammar. The goal of this article is to compare between GP-CBR [4] [5] and GE-CBR [6] approaches, comparing the accuracy rates of the similarity functions found, and the execution time needed to find a solution. We also measure the significance of the results in relation to general purpose similarity functions using different configurations and data sets.

The article is organized as follows. Section 2 surveys related work using evolutionary computation approaches to improve the retrieval phase of CBR. Section 3 presents the main ideas of CBR, GP and GE to introduce their interconnections later. Section 4 summarizes the experiments. Finally, we end with a discussion on the related work and an outlook on future research issues.

## 2. Background Work

The most used similarity functions are Minkowski [7], Mahalanobis [8], Camberra, Chebychev, Quadratic, Correlation, and Chi-square [9], Hiperrectangle based functions [10] or Heterogenous distance functions [11]. The CBR's results obtained can be improved if CBR uses an adjusted similarity function.

Several strategies based on evolutionary computation are used in order to make this 'adaptation' possible. For example, Genetic Algorithms (GA) [12] can be used as a weighting algorithm [13] [14] or as a feature selection algorithm [15]; GP can be used as a features extraction algorithm as in [16] [17].

Our research group works in breast cancer diagnosis [18]. At the moment, in one of the projects of this area we are developing a tool to obtain mammography images by content to help experts in the diagnosis of breast cancer. One of the aims of the project (TIC2002-04160-C02-02) is to define the strategy for retrieving the most similar patient's records. In [4] and [5] we propose an automatic system for discovering similarity functions using a GP-CBR approach, in breast cancer and synthetic problems respectively. The results were good, but it was very complicated to add specific knowledge to improve the searching process. In [6] we propose a strategy based on the GE-CBR approach in order to introduce experts specific knowledge, and consequently improve the searching process.

## 3. BRAIN: The Framework

BRAIN (hyBRid system to find And Improve similarity fuNctions) is a framework developed in order to define/optimize similarity functions ad hoc for a domain, using strategies based on evolutionary computation (GP and GE). The adjusted similarity function allows CBR to improve the accuracy rates. First we review CBR, GP and GE approaches, and later we present their integration.

### *3.1. Case-Based Reasoning*

CBR uses a human-inspired philosophy: it tries to solve new cases using previously solved cases. The process of solving new cases also updates the system providing new information and knowledge. This new knowledge can be used for solving other future cases. The basic method can be easily described in terms of its four phases [1]. The first phase retrieves the most similar solved cases contributing to new cases using a similarity function (i.e. Euclidean's metric). Then, in the second phase, the system tries to reuse the solutions from the previously retrieved cases to solve the new case. Next, the third phase revises the proposed solution. Finally, the fourth phase retains the useful information when the new case is solved.

### *3.2. Genetic Programming*

GP [2] is a machine learning technique based on EC. A population of individuals are evolved through generations, where in each generation the individuals are evaluated, selected, recombined and mutated. Finally, a set of them are used to build the population of the next generation. The main characteristic of GP is that its individuals are programs represented in tree form, which can be executed directly. Consequently, all the EC's operators must be adapted.

This special representation in tree form requires the definition of several arguments: Terminal Nodes (the program's variables and constants), Functions Nodes (operations applied in the program), Fitness Function (the process to evaluate a program), and finally the Ending Conditions (they indicate when the execution will stop. All these definitions are very important to warrant the Closure, Sufficiency and Universality principles of GP [2].

### *3.3. Grammar Evolution*

GE [3] is a machine learning technique based on EC, where a BNF grammar is used in a genotype to phenotype mapping process in order to transform the individual (represented by an array of bits) into a executable program or function. GE assigns a fitness value to the individual according to the individual's execution.

The BNF grammar is composed by a tuple {N, T, P, S}. 'N' and 'T' represent the set of non-terminals and terminals respectively, 'S' is the starting of the production, and 'P' defines the rules for each production of non-terminals. At the beginning of the mapping process, each individual has a program represented by the non-terminals of the starting production. The first step is clustering the bits of the individuals in integers of 'X' bits called codons, where 'X' depends of the production with more rules. Next, the non-terminals are replaced by the elements of the rule selected by Eq. 1 using the codons of the individuals. This process is repeated until all the elements of the program are terminals, and therefore the program can be executed. If the codons have run out and the mapping process has not ended then a wrapping operator is applied. It means that codons are reused again from the beginning.

$$new\ rule\ selected = MOD \frac{codon}{\#\textit{rules of the non-terminal}} \tag{1}$$

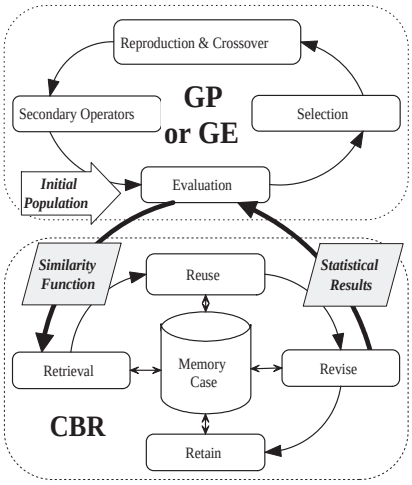


**Figure 1.** Global interaction

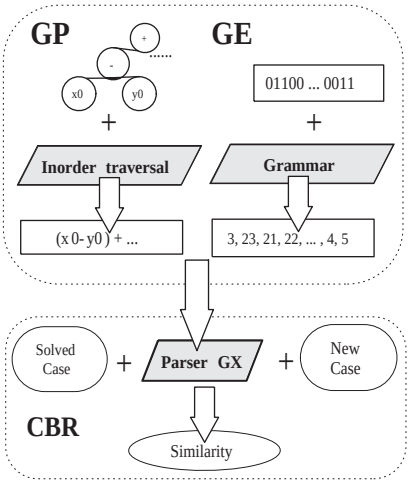


**Figure 2.** Translation process

### *3.4. Integration of GE, GP and CBR in BRAIN*

BRAIN is an automatic system to find similarity functions for complex data. We integrate a system for exploring solutions (GP or GE) and another for evaluating the solutions proposed within an only framework (CBR). The strategy based on GP and CBR [4] allows finding similarity functions without using any specific knowledge of the domain. However, sometimes this knowledge can be used to improve the searching process. In this situation, the strategy based on GE and CBR [6] uses this knowledge to lead the searching process.

Both strategies basically have the same phases (Figure 1), the main differences are in the way they apply the operators (because they use different representations) and transfer the individual to the CBR to evaluate it. There are basically two steps (Figure 2): First, the individual is transformed into a linear expression. Next, CBR applies a parser (parserGP or parserGE) in order to obtain the similarity between the new case and the previously solved.

In the case of GP-CBR, the individual (represented in tree form) is translated into a linear expression by means of an inorder traversal, not needing any more translating. In the case of GE-CBR the process is a little more complex because a genotype to phenotype mapping process must be applied, using the codons of the individual and the grammar in BNF previously defined. At the end of this mapping, the program is represented by a set of intermediate codes, where each code represent one possible terminal's value (constants, variables, operations). When CBR has executed all the test cases, it generates a set of statistics: % of correctly classified, % of unclassified, % of sensitivity and % of specificity. If CBR is executed in a cross-validation mode, the means and the standard deviation of the previous parameters are also obtained. GP and GE execute Eq. 2 to get the individual's fitness. $w_i$ allows to adjust the fitness function according to the problem. Nevertheless, other statistics can be used to compute the fitness.

$$fitness = w_1 \%\textit{sensitivity} + w_2 \%\textit{specificity} - w_3 \%\textit{unclassified} \qquad (2)$$

## 4. Experiments: Comparative, Results and Discussions

In this section we study the results using general purpose similarity functions, in relation to the GP-CBR and GE-CBR approaches. Table 2 and figure 3 summarize all the configurations applied, and table 1 the data sets used. HS and SO came from the UCI Repository [19] and MF and TA from our own repository. All of them have a different number of features to analyse its influence on the strategies.

| Data Set | Features | Type | # Classes | # Samples Train | # Samples Test |
|---|---|---|---|---|---|
| Tao (TA) | 3 | Numeric | 2 | 1700 | 800 |
| Heart-Statlog(HS) | 15 | Numeric | 2 | 243 | 27 |
| Mammography (MF) | 22 | Numeric | 2 | 196 | 20 |
| Sonar (SO) | 61 | Numeric | 2 | 187 | 21 |

**Table 1.** Characteristic of the data sets used in these experiments

| Approach | Arguments | Values |
|---|---|---|
| CBR | Function | Clark, Cosinus, Minkowsky (r=1, 2, 3) |
| | Weighting | Without Weighting (WW), Principal Component Analysis (PCA), Sample Correlation (SC) |
| | K-NN | 1, 3, 5 |
| GP & GE | Population | 500 |
| | Ending Conditions | 200 generations or 0.95% of the ideal fitness |
| | Terminals | $x_0 \dots x_{N-1}, y_0 \dots y_{N-1}, +, -, *, /, \|\ \|, \ ^2, \sqrt{}$ |
| | Operators | Prob. Crossover (0.8), Prob. Reproduction (0.2), Prob. Mutation (0.3), |
| | Selection | Tournament-2 (TS) and Rank Selection (RS) |
| | Evaluation of the individuals | CBR in 10-fold stratified cross-validation mode using the Eq. 2 with different $w_i$ values ($\sum w_i = 1$). |
| | Initialization | Grow (GI), Full (FI), Ramped half and half (RI) |
| | Replacement | Steady-State (SR), Generational (GR) |
| Only GP | Max. deep tree | 7 levels |
| Only GE | # Codons | 10 codons by attribute and Wrapping can be applied two times |

**Table 2.** Configurations of CBR, GP and GE

**BNF Grammar G**={N, T, S, P}

**N** = {<expr>,<op_binary>,<op_unary>,<var>, <op_bis>,<constants>}

**T** = {$x_0 \dots x_{N-1}, y_0 \dots y_{N-1}, +, -, *, /, abs, \ ^2, \sqrt{}$ }

**S** = <expr> / (#Features used)

**P** = <expr> ← (<expr> <op_binary> <expr>)
← <op_unary> (<expr>)
← <constants> * (<var>)

<op_binary> ← + | - | * | / | %

<op_unary> ← $abs \mid {}^2 \mid \sqrt{}$

<var> ← $x_0$ <op_bis> $y_0 \mid \cdots \mid x_{P-1}$ <op_bis> $y_{P-1}$

<op_bis> ← + | - | * | / | %

<constants> ← 0 | 0.1 | ... | 1

**Figure 3.** Grammar defined in GE to map the individuals into functions

| Problem | Function | Configuration | %Sensitivity | %Specificity | %Accuracy |
|---|---|---|---|---|---|
| TA | Clark | K-NN=3, PCA | 95.1(2.1) | 96.1(2.7) | 95.3(1.4) |
| | Cosinus | K-NN=1, WW | 00.0(0.0) | 50.0(0.26) | 50.0(0.3) |
| | Mink.(r=1) | K-NN=5, WW | 96.4(2.2) | 96.5(2.1) | 96.4(1.5) |
| | Mink.(r=2) | K-NN=5, WW | 96.7(2.4) | 96.5(1.9) | 96.6(1.4) |
| | Mink.(r=3) | K-NN=5, PCA | 96.9(2.8) | 96.4(2.1) | 96.6(1.6) |
| | GP+CBR | RI, GR, RS, w{.4,.4,.2} | 94.2(1.7) | 96.1(2.1) | 95.7(2.1) |
| | **GE+CBR** | **FI, SR, TS, w{.4,.4,.2}** | **97.6(0.9)** | **95.3(1.5)** | **96.8(1.4)** |
| HS | Clark | K-NN=3, SC | 44.8(1.1) | 0 (0) | 44.44(0) |
| | Cosinus | K-NN=3, PCA | 28.3(32.5) | 53.5(3.4) | 51.1(6.15) |
| | Mink.(r=1) | K-NN=3, SC | 81.1(9.9) | 82.8(6.5) | 81.4(6.4) |
| | Mink.(r=2) | K-NN=3, PCA | 81.2(8.2) | 78.9(9.8) | 79.2(6.8) |
| | Mink.(r=3) | K-NN=5, PCA | 81.6(9.5) | 79.2(8.6) | 80.0(9.6) |
| | GP+CBR | RI, GR, RS, w{.4,.4,.2} | 65.8(9.6) | 82.6(8.5) | 75.18(9.4) |
| | **GE+CBR** | **RI, GR, TS, w{.4,.4,.2}** | **85.1(8.5)** | **85.8(5.4)** | **85.2(6.4)** |
| MF | Clark | K-NN=3, SC | 61.2(9.7) | 70.5(7.1) | 65.7(8.6) |
| | Cosinus | K-NN=3, PCA | 10.0(30.0) | 56.2(2.5) | 56.4(4.9) |
| | Mink.(r=1) | K-NN=3, PCA | 62.3(6.9) | 73.7(11.3) | 68.1(10.0) |
| | Mink.(r=2) | K-NN=5, PCA | 62.0(8.6) | 72.3(8.7) | 67.1(12.4) |
| | Mink.(r=3) | K-NN=3, WW | 61.5(7.3) | 71.1(7.2) | 66.6(8.12) |
| | GP+CBR | RI, GR, RS, w{.4,.4,.2} | 67.7(9.3) | 61.8(6.8) | 64.2(7.2) |
| | **GE+CBR** | **RI, SR, TS,w{.4,.4,.2}** | **61.6(7.5)** | **81.3(10.9)** | **69.0(9.6)** |
| SO | Clark | K-NN=3, SC | 77.4(6.6) | 94.6(6.5) | 82.6(8.9) |
| | Cosinus | K-NN=3, PCA | 49.7(13.1) | 42.2(8.2) | 45.1(8.3) |
| | **Mink.(r=1)** | **K-NN=1, SC** | **88.0(7.4)** | **91.2(8.1)** | **88.9(7.7)** |
| | Mink.(r=2) | K-NN=1, SC | 88.2(6.3) | 88.5(11.7) | 87.9(11.4) |
| | Mink.(r=3) | K-NN=1, WW | 86.0(9.2) | 88.1(10.6) | 86.5(12.1) |
| | GP+CBR | RI, GR, RS, w{.4,.4,.2} | 75.2(11.2) | 71.3(9.1) | 74.1(9.6) |
| | GE+CBR | RI, SR, TS, w{.2,.2,.6} | 85.7(9.1) | 89.2(8.5) | 86.7(8.5) |

**Table 3.** Best configurations for each problem

Table 3 summarizes the best configurations for each problem using the general purpose similarity functions and those generated by GP-CBR and GE-CBR. The exploration of the searching space has been done using the same number of generations and individuals. The results show that problems with little volume of features get the best results (TA, HS or MF), so if the problem has a lot of features (SO) the results are not the best. This is because GP and GE would need to explore the searching space a lot more. For this reason, problems with a higher volume of features should use a bigger population and apply the cycle during more generations. Nevertheless, the results obtained in SO are good.

Another important aspect is the execution time. All the configurations have been simulated using a cluster composed by 6 computers (P-IV 2.6Ghz with 1 GB of RAM) managed by *OpenMosix* [20]. The most expensive operation is the evaluation of one individual in the CBR, which is influenced by the number of features and the train and test samples (Table 1). Table 4 resumes the time needed to evaluate one individual and find the solution for each problem with GP and GE. TA's problem has the highest evaluation time (because it has a lot of samples) but it ends before the others. This is because TA has only two features, which makes it easier to find the solution than in other problems with more features. Therefore, the number of features is a complexity factor.

| Problem | T. evaluation in GP | T. total in GP | T. evaluation in GE | T. total in GE |
|---|---|---|---|---|
| TA | 1.2 sec | 173 min | 0.09 sec | 16 min |
| HS | 0.13 sec | 250 min | 0.01 sec | 23 min |
| MF | 0.16 sec | 300 min | 0.01 sec | 29 min |
| SO | 0.24 sec | 370 min | 0.02 sec | 38 min |

**Table 4.** $\overline{T_{evaluation}}$ of one individual and $\overline{T_{execution}}$ by the GP and GE

Finally, we expose the similarity functions found using the GP+CBR (TA - Eq. 3 and HS - Eq. 4) and GE+CBR (TA - Eq. 5 and HS - Eq. 6) approaches. The others are not showed because they are too much long.

$$f(case\_x, case\_y) = \sqrt{(Y_1 + Y_1) - (X_1 - (Y_1 - X_2))} \tag{3}$$

$$f(case\_x, case\_y) = (X_2 X_{11} \sqrt{Y_9})^2 - ((X_5 + X_4)\sqrt{X_{12}})) \tag{4}$$

$$f(case\_x, case\_y) = 0.5(X_1 - Y_1) + 0.5(X_0 - Y_0) + \sqrt{||(X_1 - Y_1)||)} + \sqrt{||0.9(X_0 - Y_0)||} \tag{5}$$

$$f(case\_x, case\_y) = ||0.3(X_2 - Y_2) + 0.1(X_{11} - Y_{11}) + 0.5(X_{12} - Y_{12})|| \tag{6}$$

## 5. Conclusions and Further Work

GP and GE are approaches used in order to find programs. GP individuals can be executed directly but their management is more complex than GA individuals. Even though GE individuals need a genotype to phenotype mapping process to execute them their management is easier than GP. If we want to define a similarity function, some rules such as 'only operations between equal attributes are allowed' or 'each attribute has different influence' are very important. For this reason, the results in the last section show that the GE-CBR approach improves the results better than GP-CBR. Also, GE-CBR gets better results than CBR with general purpose similarity functions if GE-CBR is well trained. The time needed to find a solution is lower in GE-CBR than GP-CBR because the management of individuals is very expensive in GP due to the representation in tree form.

The main further works to follow are (1) defining other ways to evaluate the individuals in order to find more robust similarity functions, (2) adding GA improvements to the population of individuals, and (3) incorporating mechanism based on relevance feedback [21] in order to avoid the overfitting of functions.

## Acknowledgments

The authors acknowledge the support provided under grant numbers TIC2002-04160-C02-02, TIC2002-04036-C05-03, 2005FIR 00237 and 2002 SGR-00/55. Also, we would like to thank Enginyeria i Arquitectura La Salle for their support to our research group.

## References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundations issues, methodological variations, and system approaches. *IA Communications*, 7:39–59, 1994.

[2] J. R. Koza. *Genetic Programming. Programing of computers by means of natural selection*. MIT Press, 1992.

[3] C. Ryan, J. J. Collins, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In Wolfgang Banzhaf and Riccardo Poli, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391, pages 83–95, Paris, 14-15 1998. Springer-Verlag.

[4] E. Golobardes, M. Nieto, M. Salamó, J.Camps, G. Calzada, J. Martí, and D. Vernet. Generació de funcions de similitud mitjançant la programació genètica pel raonament basat en casos. *CCIA*, 25:100–107, 2001.

[5] J. Camps, J.M. Garrell, E. Golobardes, and D. Vernet. Diseño de funciones de similitud para el razonamiento basado en casos usando programación genética: estudio con problemas sintéticos. *MAEB*, pages 409–416, 2003.

[6] A. Fornells Herrera, J. Camps Dausà, and E. Golobardes i Ribé. Incorporación de conocimiento en forma de restricciones sobre algoritmos evolutivos para la búsqueda de funciones de similitud. *MAEB*, pages 397–404, 2005.

[7] B. Bachelor. Pattern recognition: Ideas in practice. *New York: Plenum Press*, pages 71–72, 1978.

[8] Morton Nadler and Eric P. Smith. Pattern recognition engineering. *New York: Wiley*, pages 293–294, 1993.

[9] R. Michalski, R. Stepp, and E. Diday. A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts. *Progress in Pattern Recognition, Vol. 1, Laveen N. Kanal and Azriel Rosenfeld (Eds.). New York: North-Holland*, pages 33–56, 1981.

[10] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309, 1991.

[11] D.R. Wilson and T.R. Martinez. Improved heterogeneus distance functions. *Journal of Artificial Intelligence Research (JAIR)*, 6:1–34, 1997.

[12] D. E. Goldberg. *Genetic Algorithm in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[13] L. Kuncheva. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters, Special Issue on Genetic Algorithms*, 16:809–814, 1995.

[14] J. Kelly and L. Davis. Hybridizing the genetic algorithms and the k nearest neighbors. *Proceedings of the 4th international conference on genetic algorithms*, pages 337–383, 1991.

[15] J. Jarmulak, S. Craw, and R. Crowe. Genetic algorithms to optimise cbr retrieval. In *EWCBR '00: Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning*, pages 136–147, London, UK, 2000. Springer-Verlag.

[16] M.L. Raymer, W.F. Punch, E. Goodman, and L.A. Kung. Genetic programing for improved data mining: An application to the biochemistry of proteins interactions. *Proceedings of the firt annual conference*, pages 375–380, 1996.

[17] M. Ahluwalia and L. Bull. Coevolving functions in genetic programming: Classification using k-nearest-neighbour. *In Proceedings of the genetic and evolutionary computation conference*, pages 947–952, 1999.

[18] E. Golobardes, X. Llorà, M. Salamó, and J. Martí. Computer aided diagnosis with case-based reasoning and genetic algorithms. *Journal of Knowlegde Based Systems*, pages 45–52, 2002.

[19] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[20] Open source linux cluster project. http://openmosix.sourceforge.net/.

[21] H. Zhang, L. Wenyin, and C. Hu. ifind: A system for semantics and feature based image retrieval over internet. In *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*, pages 477–478. ACM Press, 2000.