Using Genetic Algorithms to Discover Selection Criteria for Contradictory Solutions Retrieved by CBR

Costas Tsatsoulis and Brent Stephens

Information and Telecommunication Technology Center
Department of Electrical Engineering and Computer Science
The University of Kansas
tsatsoul@ittc.ku.edu

Abstract. In certain domains a case base may contain contradictory but correct cases. The contradictory solutions are due to known domain and problem characteristics which are not part of the case description, and which cannot be formally or explicitly described. In such situations it is important to develop methods that will use these criteria to select among the competing solutions of the matching cases Our domain of application was the assignment of billing numbers to the shipment of goods, and the case base contained numerous cases of similar or even identical problems that had different solutions (billing numbers). Such contradictory solutions were correct and an outcome of domain constraints and characteristics that were not part of the cases and were also not formally known and defined. It was assumed that the frequency with which a solution appeared among the retrieved cases and the recency of the time the solution had been applied were important for selecting among competing solutions, but there was no explicit way for doing so. In this paper we show how we used genetic algorithms to discover methods to combine and operationalize vague selection criteria such as "recency" and "frequency." GAs helped us discover selection criteria for the contradictory solutions retrieved by CBR retrieval and significantly improved the accuracy and performance of the CBR system.

1 Introduction

When selecting case-based reasoning (CBR) for use in a particular domain, it is assumed that similar problems lead to similar solutions, so previous experience can be reused in the formulation of a new solution. Additionally, CBR systems assume that the case base used to describe the previous experiences is consistent and correct and that knowledge represented by it remains accurate. The ability of CBR systems to reason accurately is lost if these assumptions do not hold.

Recent work in CBR is looking into developing techniques to resolve some of the problems associated with incorrect, conflicting and noisy cases by performing maintenance on a case base (among many: [8], [17], and [20]). The motivation for doing maintenance is to reduce the size of the database, maintain consistency,

and maximize domain coverage. This is done by strategically eliminating or combining cases to get rid of redundancy or inconsistency.

There are domains, though, where traditional case base maintenance techniques are not appropriate, and may lead to loss of important knowledge and information. Our work focuses on domains where the case base is large (hundreds of thousands of cases), very redundant, and contains contradictory cases. The redundancy needs to be preserved since it is a proxy of "best practices." In other words, the frequency by which an action was taken in a certain situation is an important indicator of whether this action should be chosen in future situations. The contradictory cases can be due to data input errors, but, often, they represent cycles in best practices. If the actions taken in a certain situation depend on business or other cycles, then two contradicting cases may each be correct at appropriate times. If the attribute that distinguishes between the cases is inaccessible, then both cases will be retrieved, but there is no way to select between them.

In such domains, a CBR system would retrieve a large number of cases proposing different solutions or actions, and a simple pruning based on similarity is not an appropriate selector of the best case. One possible solution to this problem is to let a user decide between the cases retrieved. This has several limitations: first, since too many cases are retrieved, the user has to analyze all of them; second, the accuracy of the case selection process is limited by the ability of the user, so an expert user may be needed; lastly, the overall workload on a user may actually increase when using the system, because of the need to perform manual case selection.

In the work presented in this paper we used the characteristics of the case base (redundancy and cyclic nature of actions) to develop an automated way to select the best case. We experimentally examined a variety of equations that combine the frequency, recency, and similarity of a case to identify the best possible way to combine these characteristics to improve retrieval. We used genetic algorithms (GAs) to find the parameters of these equations, and experimented with two different fitness metrics for the GA.

GAs have been used in previous CBR systems either as loosely coupled preprocessors (e.g. to find the weights of similarity functions) or as tightly coupled CBR components (e.g. to perform case adaptation). Conversely, CBR has been used to seed GAs to reduce the amount of search and the number of generations. Much of this work is discussed in Section 6, "Previous Research."

We applied our system to a large case base of billing records for shipments to clients. The shipment characteristics were used by the CBR system to identify similar cases and to assign the appropriate billing code. In situations where simple CBR retrieval resulted in conflicting billing codes, the application of the GA-derived selection equations resulted in selecting the correct solution out of the competing cases in 80–90% of the time. In the best case, simple CBR retrieval found a unique billing code in approximately 45% of the problems and the GA-derived selection criteria found the correct code in approximately 30% of the problems, resulting in a combined 75% accuracy; the remaining 25% of

the problems remained unsolved. Our experiments showed that the integration of standard CBR retrieval with retrieval using the GA-generated case selection criteria consistently and dramatically increased the accuracy of the selected billing codes (e.g., from 45% to 75%). This, in turn, indicated that: first, in case bases which contain redundant and contradictory information that is due to case characteristics that are not part of the case (in other words, the contradictory information is not incorrect), the frequency, recency and similarity of a case can be combined to improve retrieval; and, second, GAs are an appropriate method to identify the parameters used in combining these values in an efficient manner.

2 Domain of Application

The actual problem our research addressed was to use CBR to identify the billing code for shipments by the Burlington Northern and Santa Fe Railway Company (BNSF) [1]. The billing code in turn identifies the exact division of a particular company that should be billed for a shipment. The billing is done by BNSF employees except in cases where, for a variety of reasons, they are not sure which billing code to apply. Also, in some cases, the employees enter the incorrect code, the bill is rejected by the client, and returns to the BNSF accounting system for correction. Our system would assign the correct billing code to shipments that either had no code or were returned because they had the wrong code. According to BNSF there are approximately 5000 such billing statements per month.

Each shipping record consists of a very large number of attributes describing originating location, destination, transit points, type of cargo, type of car used, and other domain-specific information totaling over 700 attributes. BNSF maintains a multi-terabyte database of paid bills that keeps growing at approximately 600,000 billing records per month.

The database of paid bills is mostly correct, but there are records in it that contain incorrectly entered information and billing codes. Since the database is maintained over a long period of time, certain billing codes have changed, and these changes lead to billing codes that are no longer accurate and should not be used. Finally, during certain periods of the year it is possible that the billing codes will change, so that billing numbers may no longer be correct.

The high-level goal of our CBR system was fairly simple: given a new shipment, select the appropriate billing code by matching it against the database of previous billing cases (i.e. paid bills). The difficulty was that in so doing our system would retrieve many billing codes from cases that were either old, incorrect, or not valid for the current cycle of operations. When discussing the problem with the BNSF billing experts they indicated that there were two major selection criteria they would use to resolve any conflicts: the *volume* or *frequency* and the *recency* of a billing number. In other words, how many times a billing number had been used in the past coupled with how recently the number had been used.

[1] BNSF operates a railroad network with 33,000 route miles in North America

The combination of these two parameters is not straightforward, though. Cyclic billing means that some recent billing records would be correct, but have much less frequency than older ones that were generated over a long period of time. On the opposite end, a recent record may be simply incorrectly entered by BNSF personnel and should not necessarily override voluminous but older records.

Traditional case base maintenance techniques were of no help, since the conflicting cases were, in most situations, correct for different billing cycles and conditions, and such cycles and conditions were not consistent across all customers, nor were they always dependent on available information (for example, a customer moved temporarily to a new address, requiring a new number, and then moved back to their original offices, requiring the old number, but this happened in no predictable fashion). The case base did not have obsolete information that could be simply pruned; it contained current, sometimes conflicting, billing cases. To capture all rules and information regarding the definition of billing cycles and changes would be a very expensive and error-prone exercise.

3 CBR System

The basic CBR system for retrieval of the most similar cases used a simple, weighted nearest-neighbor matching of 20 features of the shipping record that required a billing number and the cases in the billing record case base. Weights were determined by BNSF experts of the domain, who also determined the attributes that were used in the matching process. The threshold over which a case was considered similar was varied experimentally between 1 and 0.9 in increments of 0.02 to identify any trends in retrieval. In this particular application varying the similarity threshold led to trivial and expected results: a lower threshold retrieved a larger set of similar cases and had fewer inputs that retrieved no similar cases.

Once the initial matching is complete, a billing code is assigned only if all retrieved cases above the cut-off threshold have the same billing code (something that happens in approximately 45% of the problems submitted to the CBR system). Otherwise the cases are analyzed by additional processing which uses the GA-created selection criteria, as described next. Billing problems that retrieve no similar cases are left for users to determine the appropriate billing code.

4 Generating Weighted Solution Selection Criteria

When CBR retrieved different billing codes, leading to conflicts as to the solution the system should select, our system needs to decide which of the billing codes to pick. As mentioned, the selection of the most appropriate case should be based on a combination of the recency of the billing case, its frequency (how often it has been used), and its similarity to the current billing problem. At issue was how to combine these parameters in a way that led to the selection of the correct

billing case. We experimented with different equations and weighted parameters using genetic algorithms (GAs).

Before we describe the equations we used, we need to define two parameters: *frequency* and *recency*. Frequency is simply the relative frequency of a billing number in the cases retrieved. So, for example, if we retrieve 100 cases that are over the matching threshold, and 60 of them have the same billing number 12345, then the frequency of 12345 is 0.6. To compute the recency of a billing number we first need to define the cut-off distance, in other words the number of days after which a billing number is considered outdated. Each day between the current date and cutoff date is assigned a number between 0 and 1, computed linearly over the time interval. So, for example, if the cutoff date is 5 days in the past, the current date has value 1.0, yesterday has value 0.8, two days ago is 0.6, etc. The recency of a billing number is the normalized sum of the recency value of each case retrieved that has this number.

We defined six different weighted solution selection metrics:

1. A linearly weighted frequency: $\alpha \times frequency - \beta$
2. An exponentially weighted frequency: $\alpha \times e^{-(\beta(1-frequency))} + \gamma$
3. A step function for frequency: $\left\lceil \frac{frequency}{1/\alpha} \right\rceil \times \beta$
4. A linearly weighted recency: $\alpha \times recency - \beta$
5. An exponentially weighted recency: $\alpha \times e^{-(\beta(1-recency))} + \gamma$
6. A step function for recency: $\left\lceil \frac{recency}{1/\alpha} \right\rceil \times \beta$

We also defined two simple retrieval methods to be used as comparisons:

1. Most recent: Select the billing number that is in the most recent case
2. Most frequent: Select the billing number that appears most often in the whole set of retrieved cases

Next we used GAs to find the parameters of the six equations described above. We ran nine different experiments, one each for a different maximum similarity threshold (1, 0.98, 0.96, 0.94, 0.92, and 0.90). We repeated our experiments using two different fitness functions, to investigate the effect of the fitness function on the selection of the appropriate solution. The first fitness function was based only on the percentage of the problems that were solved correctly (i.e. were assigned the correct billing number). The second fitness function combined the percentage of the problems solved correctly with the difference between the matching value of the highest-ranked case that had the correct solution and the matching value of the highest-ranked case that had an incorrect solution; the goal was to have high accuracy but also to punish incorrect solutions that matched highly. Figure 1 shows the overall solution accuracy for the two fitness functions; since the fitness function that takes only percent correctness into account performed significantly worse, we will only discuss our results using the second fitness function.

The genetic algorithm was run for 1000 generations per experiment. Each generation population consisted of 200 individuals. The likelihood for crossover

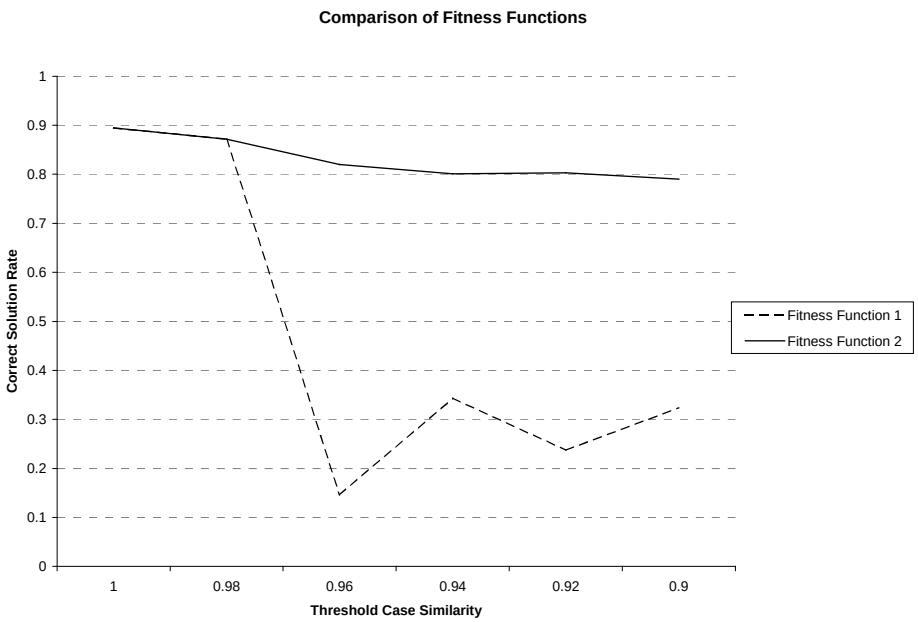

Fig. 1. Comparison of the overall results of the two fitness functions. "Fitness Function 1" uses only the percentage of correct solutions and performs significantly worse as the matching threshold for case selection is lowered.

was 99 percent, and the likelihood for mutation was 1 percent. The parameters we wanted to determine where encoded as bit strings. For example, for the exponentially weighted frequency equation, the GA tried to find the parameters α, β, an γ; for the linearly weighted recency equation, the GA tried to find the parameters α, β, and the cutoff date distance; etc. We used a case base of 500 cases for training and another one of 500 cases for testing. The training cases were again divided into five sub-sets of 100 cases each. The GA switched between training sub-sets for each generation, in an effort to avoid overfitting the solution.

The numbers generated were floating point with resolution of 0.001. For the exponential formulas we used 19 bits to encode the α parameter, 17 bits to encode the β parameter, and 20 bits to encode the γ parameter. For the linear formulas we used 17 bits to encode the α parameter and 10 bits to encode the β parameter. There are integer parameters, too, for example the date length is encoded in 9 bits (the date length is the number of days in the past when recency is set to zero), and the step number and step height each are encoded in 10 bits. The parameters for each equation are stored in a single chromosome of maximum length 90 bits. As already mentioned, the fitness function combined the percentage of the problems solved correctly with the difference between the matching value of the highest-ranked case that had the correct solution and the matching value of the highest-ranked case that had an incorrect solution.

For example, the GA found the following parameter values for the exponentially weighted recency equation when the similarity threshold was set to 0.90: cutoff date distance= 17 days; equation: $0.089 \times e^{-(0.053(1-recency))} + 0.085$

Next, we used the same GA process to discover the parameters of a *combination formula*, $\sum_i^8 w_i \times f_i$ where w_i are the parameters and f_i are the outputs of the six formulas and the two simple solution selection methods ("most recent" and "most frequent"). The weight for each formula was encoded as a 10-bit string in the GA.

5 Results

After running the GAs, for each similarity threshold we had three formulas that weighted recency, three that weighted frequency, and one formula that combined the previous six and the two simple selection criteria of "most recent" and "most frequent." These nine selection criteria were tested on the 500 problems of our test set, and the results are shown on Figures 2 and 3 (note that three criteria, most frequent solution, and step and exponential weighted frequency, had identical results). We ran three different experiments with the same test set and all resulted in similar equations and the results were within one percentage point. While more experiments may be needed, our initial results indicate that the formulas generated by the GAs are stable and produce almost identical results. The results we are showing in the following graphs are from a single experiment.

Figure 2 shows the percentage of correct solutions (billing codes) that were identified by the nine different selection criteria as a function of the similarity threshold used. The combination formula does the best, and is the most consistent across changing threshold values. Note that the ratio of correct solutions is computed as a percentage of the solutions that the system found, and does not include situations where no solution could be found. Because of this, the graphs of Figure 2 are slightly misleading, since they could lead to the naive conclusion that the best performance can be achieved using a similarity threshold of 1.0, or, in other words, by demanding a perfect match. Figure 3 shows a better comparison of the nine selection criteria.

Figure 3 uses as an evaluation criterion the "contribution to the overall solution." This metric indicates how many correct solutions were found, versus (1) incorrect solutions, plus (2) cases where there were no competing solutions (i.e. all retrieved cases had the same billing number[2]), plus (3) problems where no solution was identified since no case matched at or above the matching threshold. So, this rate is the percentage of problems for which competing cases where retrieved times the percentage of them that were solved correctly. We believe that this is a better criterion of the overall performance of the selection metrics, since it indicates the contribution that the selection metrics make to the overall

[2] As mentioned, when cases are found and retrieved, in approximately 45% of the problems there are no competing solutions. So, the solution selection criteria are used in the remaining 55% or of the problems

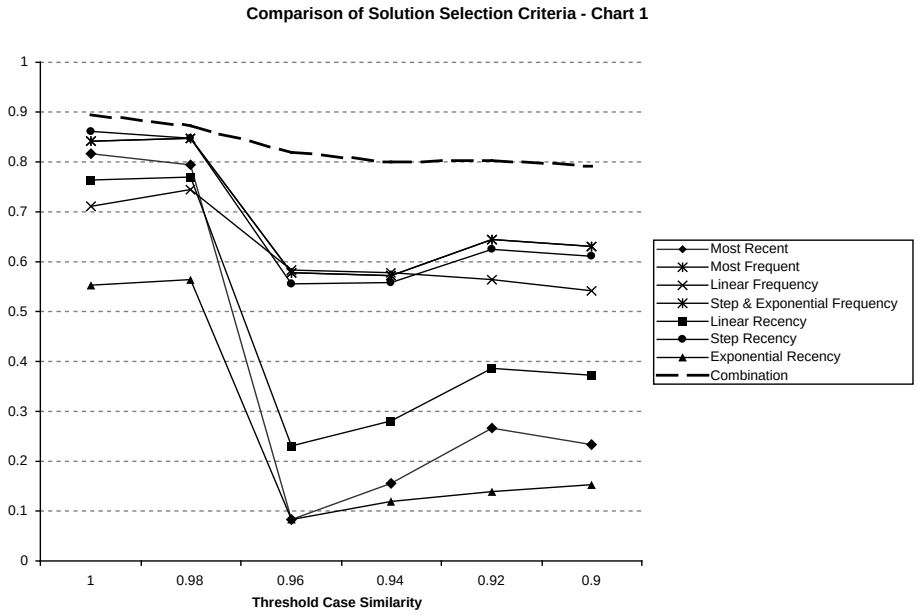

Fig. 2. Comparison of the nine solution selection criteria as a function of the percent of problems they solve correctly versus the similarity threshold used for case selection.

quality of the solution. Figure 2, on the other hand, shows the accuracy of the selected solution in cases where the system *can find a solution.*

For example, for a matching threshold of 1.0, CBR retrieval returned 24 cases without conflicting billing numbers, and 25 cases with conflicting billing numbers. Of the latter, we used the GA-generated combination formula to select the correct number and did so successfully in 22 instances (approximately 88%). So, at matching threshold of 1.0, the success of the simple CBR retrieval is 24 out of 500 (5%), while the combined system is 46 out of 500, or approximately 9%. The success of the GA-generated combination formula for the same threshold was 22 out of 476 (that is, 500 test cases minus the 24 that had no conflicting billing numbers), or approximately 4.6%. This 4.6% is the *contribution* of the GA-based selection criteria to the overall solution quality.

The combination method gave the best overall results, contributing 31% points to the overall solution quality. There seem to be no obvious trends for the other criteria. Interestingly, the "simple" solutions of selecting either the most frequent or the most recent solution do diametrically different, with frequency giving much better results than recency. On the other hand, the recency of a solution is important, as shown by the success of the "step recency" formula. Currently we have no insights as to the reasons the graphs look as they do, and, as discussed in Section 7, "Conclusions," we are performing more experiments to help gain a better understanding of the behavior of the GA-generated selection criteria. Since the current experiment dealt only with small variations in the

matching threshold, it is possible that the behaviors shown by the individual plots are small permutations that will not seem important as we extend our experiments to smaller matching threshold, say down to 0.6.

Despite the performance of the individual selection criteria, though, it is clear that the combination method performs substantially better than the others, and especially against the simple selection methods of "most recent" and "more frequent."

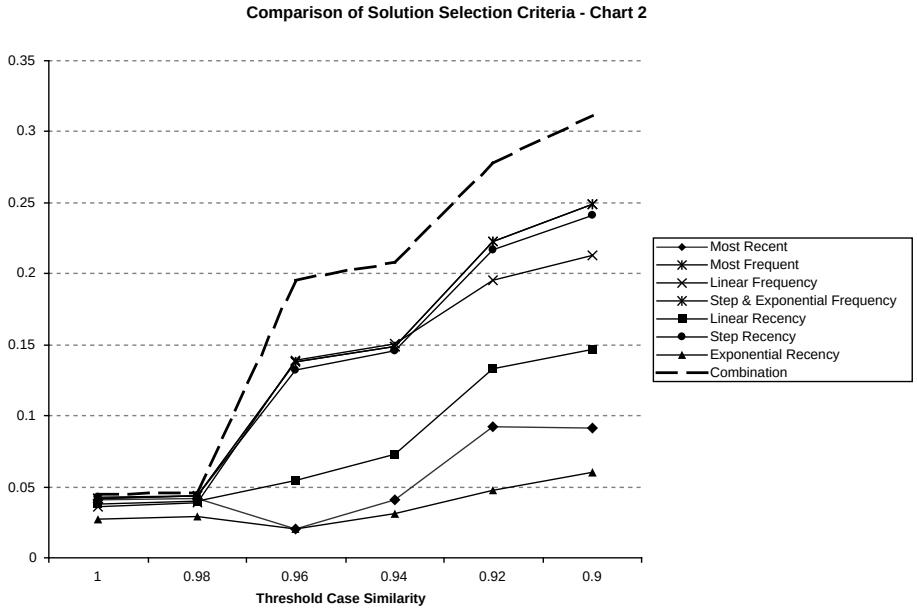

Fig. 3. Comparison of the nine solution selection criteria as a function of the contribution they make to the overall solution quality (expressed as percentage added to the combined CBR+GA system's accuracy) versus the similarity threshold used for case selection.

6 Previous Research

There has been a substantial volume of research that integrated CBR with genetic algorithms (GAs). Some of the previous work used CBR to seed GAs, so that they would not start from a random configuration of chromosomes, but, instead, start from a generation that —based on experience— was "close" to the required solution. For example, Ramsey and Grefenstette [14] presented a GA that was initialized by cases that contained descriptions of past GA runs (e.g. task environment and parameter values), and where CBR aimed to focus the GA to the current environment and situation. Liu [9] injected 5–15% of cases

into a GA system for the design of gate-level circuits. Similarly, Perez et al.[12] used a case base of designs to seed a GA that also evolved circuits at the gate level. The work by Job et al. [4] is also very similar, in that it uses cases to seed an FPGA design program that uses GAs.

Other research used GAs to adapt solutions retrieved by CBR. This work is very similar to the one where CBR is used to seed GAs (the retrieved solutions can be seen as the GA seeds), but the focus is more on CBR than GAs. For example, structural design cases were adapted using GAs in the work described in [3]: CBR was used to retrieve up to four cases from a case base, that were then used as the initial population for a GA that evolved them to generate the most appropriate solution. A GA was also used to adapt designs that conform to feng shui rules and that were retrieved by CBR [1]. As in previous such work, the cases were used as "seeds" for the GA that then adapted them using standard GA techniques. The authors concluded that the combined CBR and GA system solved the same percentage of problems as a GA system alone, but that it generated designs of better quality and in faster times. A slightly different approach was taken in the work by Purvis and Athalye [13], where the generation of a solution from multiple retrieved cases was seen as a CSP problem that was solved by a CA.

The work in the CIGAR system [10] integrates GA seeding by a case base with the adaptation by a GA of the solutions generated by CBR. CIGAR uses a case base to inject part (10–15%) of a population at every GA step when attempting to solve a new problem in an evolutionary manner; as the GA proceeds, the solutions offered by the original cases are evolved by crossover and mutation to adapt to the new problem; new solutions are then learned by getting stored in the case base.

A large body of research has focused on determining the weights of features used in case matching using a GA. Early work [5] used GAs to find feature weights for a k nearest neighbor algorithm. Skalak [16] demonstrated how GAs can be used to perform feature selection and showed how this can be of great help in reducing computational costs without sacrificing accuracy in pattern recognition problems. Oatley et al. [11] used a GA to optimize the values of the weights of features used to compute similarity. Similar work is described in Shin and Han [15], where GAs are used to decide on the best indexing features for a CBR system for stock market prediction. In Kool et al. [7], the weight, selection, and ordering of features is performed using a GA and a case-based approach, and the two techniques are compared in the domain of memory-based language processing.

A very preliminary study of using GAs for CBR maintenance was discussed in [2]. In this work a set of representative cases were selected, and then their feature weights were modified by a GA to allow the cases to "move" in the problem space. Communication between case-controlling agents would eventually guide the GA to position all cases in a way that would provide the minimal representative case base.

Soh et al. [18] and Soh and Tsatsoulis [19] have examined using GAs to create a case base. In their work they define a set of domain-specific evaluation criteria for the quality of cases, but no actual case base exists initially. GAs are then used to generate populations of cases that fit the evaluation criteria and that are then used by a standard CBR system for problem solving.

Our work is not really similar to most previous work that combined CBR and GAs. Closest to our approach is the research that learns feature weights, since it attempts to improve retrieval, which is what our system also tries to do. Our work, though, applies GAs to cases where traditional CBR matching and retrieval has failed, and assumes that information that is *not* part of the feature set of the case (and, consequently, not known or not knowable) is what would improve retrieval.

Finally, early work on the CBR system PARADYME [6] used the recency with which a particular source case was last used as preference heuristic between retrieved cases that had the same similarity value. PARADYME's main goal was to show how different goals should lead to weighting case features differently, and the psychologically inspired factor of recency was just a simple post-retrieval criterion. While the goals of our system are similar to PARADYME's use of recency (select between competing cases), as mentioned in our discussion of the domain, cyclic billing means that some recent billing records would be correct, but have much less frequency than older ones; on the other, a recent record may be simply incorrectly entered by personnel and should not necessarily override voluminous but older records. Because of these complexities, it was necessary to learn/evolve the formulas that combine similarity, frequency and recency for case selection. Note also that in our work "recency" does not reflect when a case was used last (i.e. by the CBR system), but when the action described by a case was last performed.

7 Conclusions

In certain domains a case base may contain contradictory but correct cases. The contradictory solutions are due to known domain and problem characteristics which are not part of the case description, and which cannot be formally or explicitly described. In such cases, it is important to develop methods that will use these criteria to select among the competing solutions of the matching cases. We showed how one can use genetic algorithms to discover methods to combine and operationalize vague selection criteria such as "recency" and "frequency." In our domain it was assumed that these criteria were important for selecting among competing solutions, but there was no explicit way for doing so. GAs helped us develop simple equations that combined the solution selection criteria, and significantly improve the correctness of the system. In the best case experiment, the GA-developed selection criteria added approximately 31% correct solutions to the CBR system as compared to using simple weighted matching and retrieval. For our application this translates to over 1500 bills per month that would be

send to the correct payee and would not be contested, resulting in substantial savings and in uninterrupted payments for shipments.

We observed that formulas combining all selection criteria performed better than ones considering a single criterion. We also discovered that the selection of the fitness function for genetic learning had a significant impact on the solution, and different fitness functions should be experimented with before committing to one. In our, albeit only three, experiments we noticed that the selection criteria generated by the GAs were very similar and led to almost identical results.

We intend to continue our experiments with decreasing similarity thresholds in an effort to see if contributions to the overall solution quality offered by the GA-generated selection criteria can be improved beyond the current 31%, and to find the point where —as we expect— performance will start declining.

More generally, CBR retrieval has —in most cases— resulted in the selection of the best matching case (or a small number of top matching cases), from which a solution is then adapted. Most work has concentrated on finding the best features, or the best weights, or has used the user to guide the ultimate selection, as done in Conversational CBR. The assumption has been in most cases that the similarity assessment process was sufficient to guide the problem solver to the best case from which to start adaptation. Our work in an, admittedly, uncommon domain of application, has led us to believe that in cases where there is no guarantee that the feature set describing a case adequately represents the problem instance, or that it would lead to correct case retrieval, the use of case selection criteria used after the similarity assessment phase can greatly improve performance.

Many case bases start as company data bases or textual documents, and their contents were never designed to be cases or be useful in establishing similarity between them. In such environments, the developers of the CBR system must try to elicit from experts further selection criteria and rules. Another possibility, as we have shown, is to have the experts identify which characteristics that are not part of the case description they use in selecting between cases, and then use these characteristics to learn the actual selection equations. In our work, the expert-defined characteristics were frequency and recency; in other domains they may be different. Our experiments showed that GAs are a good way of generating these selection criteria equations, and significantly improved the performance of the CBR system. While more experimentation will be necessary, it seems that the selection functions are stable and might be modeling inherent properties of the domain and of the expert case selection process. It would be interesting to attempt a similar experiment in a more "traditional" CBR application, and to study whether GA-generated selection criteria would do a better job selecting the best case from a subset of matching cases as compared to selecting the best matching case.

8 Acknowledgments

This work was supported in part by the Burlington Northern and Santa Fe Railway Company. The comments by the anonymous reviewers helped us improve the exposition of our work.

References

1. de Silva Garza, A.G. and M.L. Maher: "An Evolutionary Approach to Case Adaptation," in Proceedings of 3rd Int. Conf. on CBR (ICCBR-99),K.-D. Althoff, R. Bergmann and L.K. Branting (Eds.), Berling: Springer-Verlag, (1999) 162–172.
2. Huang, Y.: "An Evolutionary Agent Model of Case-Based Classification," in Proceedings of 3rd European CBR Workshop (EWCBR-96), I. Smith and B. Faltings (Eds.), Berling: Springer-Verlag, (1996) 193–203.
3. Hunt, J.: "Evolutionary Case Based Design," in Progress in Case-Based Reasoning: 1st UK Workshop, I.D. Watson (Ed.), Berlin: Springer-Verlag, (1995) 17–31.
4. Job, D., Miller, J. and Shankararaman, V.: "Combining CBR abd GA for Designing FPGAs," in Proceedings of 3rd Int. Cnf. on Computational Intelligence and Multimedia Applications, (1999) 133–137.
5. Kelly, J.D. and Davis, L.: "A Hybrid Genetic Algorithm for Classification," in Proceedings of 12th Int. Conf. on AI (IJCAI-91), (1991) 645–650.
6. Kolodner, J: "Selecting the Best Case for a Case-Based Reasoner," in Proceedings of 11th Ann. Conf. of the Cognitive Science Society, Hillsdale, NJ: Larence Erlbaum Associates, (1989) 155–162.
7. Kool, K., Daelemans, W. and Zavrel, J.: "Genetic Algorithms for Feature Relevance Assignment in Memory-Based Language Processing," in Proceedings of 4th Conf. on Computational Natural Language Learning and of the 2nd Learning Language in Logic Workshop, Lisbon, Somerset, NJ: Association for Computational Linguistics, (2000) 103–106.
8. Leake, D. B. and Wilson, D. C: "Case-Base Maintenance: Dimensions and Directions," in Proceedings of European Workshop on Case-Based Reasoning. Springer-Verlag, Berlin Heidelberg New York (1998) 196–207.
9. Liu, X.: "Combining Genetic Algorithms and Case-based Reasoning for Structure Design," M.S. Computer Science Thesis, University of Nevada at Reno (1996).
10. Louis, S.J. and Johnson, J.: "Robustness of Case-Initialized Genetic Algorithms," in Proceedings of 12th Int. Florida AI Research Society (FLAIRS-99), (1999) 129–133.
11. Oatley, G., Tait, J. and McIntyre, J.: "A Case-based Reasoning Tool for Vibration Analysis," in Applications and Innovations in Expert Systems VI: Proceedings of the BCS Expert Systems Conference, R. Milne, A. Macintosh and M. Bramer (eds.), Berlin: Springer-Verlag (1998).
12. Perez E.I., Coello, C.A. and Aguirre, A.H.: "Extracting and Re-Using Design Patterns from Genetic Algorithms using Case-Based Reasoning," in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002, Langdon, W.B. et al. (Eds.), San Francisco, CA: Morgan Kaufmann Publishers, (2002).
13. Purvis, L. and Athalye, S.: "Towards Improving Case Adaptability with a Genetic Algorithm," in 2nd Int. Conf. on CBR (ICCBR-97), Leake, D.B. and E. Plaza (Eds.), Berlin: Springer-Verlag, (1997) 403–412.

14. Ramsey, C.L. and Grefenstette, J.J.: "Case-Based Initialization of Genetic Algorithms," in Proceedings of 5th Int. Conf. on Genetic Algorithms, (1993) 84–91.
15. Shin, K. and Han, I: "Case-based Reasoning Supported by Genetic Algorithms for Corporate Bond Rating," J. of Expert Systems with Applications 16(2), (1999) 85–95.
16. Skalak, D.B: "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms," in Proceedings of 11th Int. Conf. on Machine Learning, (1994) 293–301.
17. Smyth, B. and McKenna, E.: "Modeling the Competence of Case-Bases," in Proceedings of 4th European Workshop on Case-Based Reasoning, EWCBR-98, Lecture Notes in Artificial Intelligence 1488, Smyth, B. and Cunningham, P. (Eds.), Berlin: Springer Verlag, (1998) 208–220.
18. Soh, L-K., Tsatsoulis, C., Jones, M. and Agah, A: "Evolving Cases for Case-Based Reasoning Multiagent Negotiations," in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Spector, L. et al (Eds.), San Francisco, CA: Morgan Kaufmann Publishers, 909 (2001).
19. Soh, L-K. and Tsatsoulis, C.: "Combining Genetic Algorithms and Case-Based Reasoning for Genetic Learning of a Casebase: A Conceptual Framework," in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Spector, L. et al (Eds.), San Francisco, CA: Morgan Kaufmann Publishers, (2001) 376–383.
20. Zhu, J and Yang, Q.: "Remembering to Add: Competence-Preserving Case-Addition Policies for Case Base Maintenance," in Proceedings of the Fifteenth Int. J. Conf. on Artificial Intelligence. Morgan Kaufman (1999) 234–239.