Case-Based Reasoning for Multi-Step Problems and its Integration with Heuristic Search

Christian Reiser
Alcatel Austria
Ruthnergasse 1–7
A–1210 Wien, Austria
C.Reiser@aaf.alcatel.at

Hermann Kaindl
Siemens AG Österreich
Geusaugasse 17
A–1030 Wien, Austria
kaih@siemens.co.at

Abstract

The usual case-based reasoning approach assumes that for each given problem instance it is necessary to retrieve from scratch a similar case from the case base. Therefore, an indexed memory structure or other means of facilitating fast access is typically needed. Moreover, a complete solution is usually stored together with each case, that can be adapted to the given problem. We developed a different approach for *multi-step* problems. It utilizes the information about the relevant case for the last step to quickly find the appropriate case for the current step from only few relevant cases that are connected. Therefore, no special indexing schemata are required. Instead, we store a value for each case and similarity links to other cases, but no solution. For situations outside the scope of the case base we integrated case-based reasoning in several ways with heuristic search. We performed experiments in a game domain, that showed the usefulness of our approach. In particular, we achieved a statistically significant improvement through combination of case-based reasoning with search over pure search or pure case-based reasoning. For multi-step problems, our approach appears to be more useful than the standard approach to case-based reasoning.

Introduction

In recent years, case-based reasoning (CBR) has become more and more popular. However, there appears to be a lack of theory of how to apply it best to *multi-step* problems. Multi-step problems are very common in industrial applications like, e.g., factory and power plants, railway and network control. Such problems require several steps in order to achieve a solution. A step has to be chosen before finding a *complete* solution is feasible in real time. Consequently, it is rarely possible to find optimal solutions. However, complete solutions to multi-step problems are often not found at all. Therefore, they are unavailable for storage in a case base.

From a planning perspective, this means that plan generation and execution are intertwined. While such problems are common also in single-agent problem solving, we primarily studied case-based reasoning in the context of two-player games with perfect information. In particular, we experimented with a strategic game named Abalone (see Appendix).

Usually, access to a case base is done during problem solving in two parts: through an index, and using a similarity metric. Apart from the effort involved in building such an index structure, its compatibility with the similarity metric is an important issue. In our context, the index part can be omitted, since our approach utilizes the information about the relevant case for the last step to quickly find the appropriate case for the current step from only few relevant cases (that are connected).

Moreover, in the classic CBR approaches, a *complete* solution is stored with each case, and this solution is adapted to the given problem. For dealing with multi-step problems in real time, we prefer a different memory organization. We store a value for each case and similarity links to other cases instead of a solution. These values give "hints" for the selection of the next step. For dealing with situations outside the scope of the case base we show several ways to integrate case-based reasoning with heuristic search.

Our approach described in this paper specially copes with multi-step problems and incomplete solutions. Finding the appropriate case is optimized and no complete solution is stored explicitly in the case base.

First, we describe our novel approach to case-based reasoning for multi-step problems in a real-time environment. Then we present our concepts for integrating it with heuristic search. Empirical results show the usefulness of our overall approach. Finally, we discuss its relation to other work.

Our Approach to Case-Based Reasoning for Multi-Step Problems

Memory Organization of the Case Base

The case base contains a number of states (positions of the game). Each such state is one case. For each case the state information (the board configuration) and the value of the state (from the evaluation function) is stored.

No solution is stored directly with the case (for large problems such a solution may not even be known). However, relations between similar cases are explicitly stored (see the bottom of Figure 1). Hence, information about other relevant cases is available.

In our approach, no further memory organization like indexing schemata is needed. Instead we use a *history pointer* which indicates the relevant case for the current state.

101

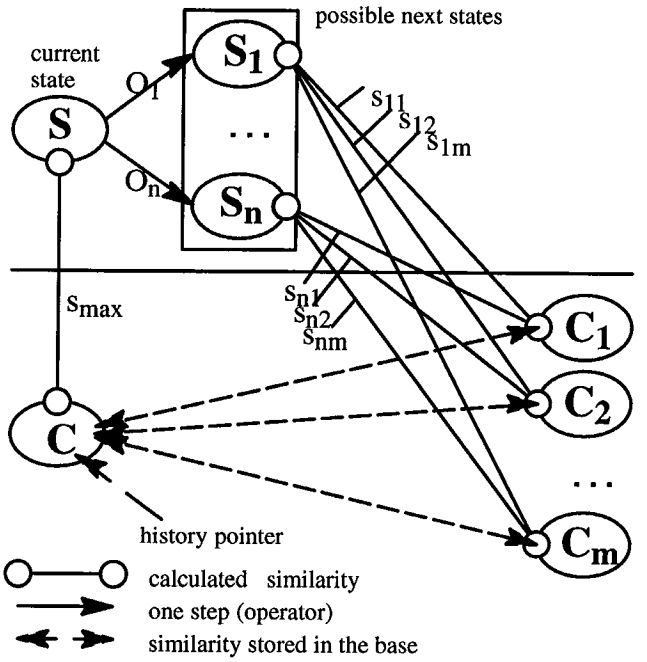

Figure 1: Interplay between the multi-step problem and the case base

The retrieval and adaptation algorithm described below uses these cases as a hint to the solution for the current case (cf. subgoals in planning). The current case is the case in the base which is one of those most similar to the current state.

The Retrieval and Adaptation Algorithm

Our retrieval and adaptation algorithm uses a case base of the type described above to determine the next step. A history pointer always points to the case used for the last retrieval. At the beginning, it is initialized with the start state (of the game).

When the next retrieval occurs, the relevant case for the current state is either the case to which the history pointer points or, one of the cases to which a similarity relation exists in the case base from this case. So the search for the relevant case is reduced to a limited number of cases.

The adaptation algorithm calculates all states reachable from the current state and checks which of them is the "best". For each of these states both the similarity to a case in the base and the value of this case (from an evaluation function) are taken into account, because a more similar case is easier to reach and a high value should be found.

Figure 1 shows the interplay between the multi-step problem (above the line) and the case base (below). The retrieval and adaptation algorithm can be sketched as follows:

1. Locate the case referred to by the history pointer and all cases directly related to this one in the case base, and determine among these the case C that is most similar to the current state S (s_{max}).
2. Generate all states S_i which are reachable from S in one step by applying the operators O_i defined in the domain (e.g., possible moves).
3. Calculate the similarities s_{ij} between the states S_i and the cases C_j for which similarity relations with C are stored in the case base.
4. Combine the similarities s_{ij} with the values of the cases C_j with a function, which weights the similarity versus the value, and select the step which leads to one of the states S_i with the best result. This step is selected and performed in the real world.
5. Check, if the new state is more similar to one of the cases C_j than to C. If so, change the history pointer to this case.

The similarity metric — which also exists in classical case bases — is more important in this approach, since together with the history pointer it also replaces the indexing mechanism normally used to find the relevant case. Moreover, the similarity relations stored in the case base can also be based on this metric (see below).

The adaptation algorithm introduced here uses multiple cases to find a solution, but it does not split the cases. It selects a step by comparing several ones.

How to Generate a Case Base

When there is a large number of possible states, it is hard to generate a case base by hand. All relevant states have to be found and additional states in between have to be generated, so that the similarity between the cases is high enough.

When case bases are generated by hand, parts of earlier step sequences are taken, some of the states are removed and the remaining ones are used as cases. Each case has a similarity relation to its predecessor and successor in a sequence and additional relations are introduced between parts of different step sequences. (The relations between the cases of a step sequence are a useful optimization, but the case base can also be based only on the similarity metric.) Such a generation requires much knowledge about the application domain.

Less sophisticated but more easily achievable case bases were generated automatically for the game used for our experiments as follows.

With the algorithm described below, 100 different realistic game positions (states) were generated. An iterative $\alpha\beta$ search played against itself, and the records of these games were used to generate case bases.

Depending on the target size of the case base every nth position was converted into a case with similarity relations to its predecessor and successor. These cases were added together, where each new case was linked with a similarity relation to the most similar case already in the base, using the similarity metric. The values of the cases are the static values of the positions assigned by the static evaluation function that is also used by the search heuristic (see below).

These case bases are independent of human skill and can be used for an objective comparison of different algorithms.

Integration of Case-Based Reasoning with Heuristic Search

When following the case base as described above, states may occur that are not sufficiently similar to any case in the

case base used. In such a situation, it is possible to switch to another problem-solving method: heuristic search.

For two-player games with complete information, there exists a well-developed theory about using heuristic search (for a review see, e.g., (Kaindl 1990)). The standard approach is to perform iterations of deeper and deeper minimax searches. These can be performed using several algorithms (for a comparison see (Kaindl, Shams & Horacek 1991)). The best-known of these is the $\alpha\beta$ algorithm.

Sometimes it may be known a priori that in certain situations the case base is inappropriate. In the game-playing context this applies to "tactical" situations, where large differences in the evaluation may occur (cf. the concept of *quiescence* in (Kaindl & Scheucher 1992)). Cases representing such situations can be labeled. Whenever a given state is most similar to such a labeled case, heuristic search is used instead of case-based reasoning. In some sense, these labels represent *meta-knowledge* about the case base itself, i.e., its (in-)competence.

Another useful possibility for switching to the search mode occurs in situations where a tactical move is possible. This means that the case base is considered incompetent in all such situations (or at least it is safer to use search here). In this variant the decision to switch is based on the actual problem, not on the case base.

These approaches represent an integration of case-based reasoning with heuristic search, but a rather loose one. Tighter integrations are possible when using the knowledge in the case base for *guiding* a search.

One possibility is to search branches deeper that contain favorable steps according to the case base. This leads to *variable-depth* search (see, e.g., (Kaindl 1983)).

When the branching degree of the given domain is high (as in the game we used for our experiments), an old approach called *forward pruning* becomes of interest. While it turned out to be not the first choice in highly tactical domains like chess, we investigated its integration with the case base in one of our variants. With the exception of steps that lead to material change (in order to cope with tactics), the search prunes away all the moves that are not considered useful according to the case base.

Empirical Results

The procedure to statically evaluate a position (the heuristic evaluation function) was used both in the $\alpha\beta$ algorithm and to assign a value to the cases in the base. In this way, the same domain knowledge about the quality of positions is available both for the search approach and for case-based reasoning.

This static evaluation function (for Abalone) takes into account the material balance, the distance of the balls from the border, and their compactness.

Similarity Metric

The similarity metric is an essential part of our case base and the appropriate retrieval and adaptation algorithm. We defined and used one that compromises between exactness and run-time cost.

In order to use the value 0 for identical positions, actually the difference rather than the similarity is expressed by this metric. Low values indicate similar states and high values dissimilar ones. A detailed description and a proof that it is a metric can be found in (Reiser 1994).

Experiment Design

Since all of the investigated algorithms are deterministic, some special means have to be taken to gather statistical data. The games in the tournaments between the algorithms were started with automatically generated start states. 200 different states were generated as described below.

1. The algorithm for iterative $\alpha\beta$ search was changed so that it does not use the best but randomly 1 of the 5 best steps.
2. Using a random number generator, 5 – 40 steps were played with this changed algorithm.
3. The resulting state was statically evaluated and rejected if the absolute value was larger than a certain limit, i.e., the advantage of one of the colors was decisive.
4. The colors of the states were exchanged to get another position.

Since the automatically generated case bases provide different information for different colors, it is not possible to compare our algorithm with another one by having it play once as black and once as white on the same start position. This is compensated by exchanging the colors of the position as described above.

In our experiments, the influence of different sizes of case bases as generated from the same "training" games as well as different combinations of the case base with search algorithms were investigated.

We compared all our algorithms using the case-base with an iterative $\alpha\beta$ search with variable search depth. Since all our algorithms were tested against the same opponent, comparing them is possible.

Results

The difference in size of case bases generated from the same set of games showed no statistically significant difference in their results (according to the sign test) when at least every 16th position is inserted in the case base. We used the case base containing every 4th position for the experiments of comparing several variants of case-based reasoning (integrated with search) against the pure iterative $\alpha\beta$ search.

A small selection of the most interesting results of our experiments is summarized in Table 1. (More details can be found in (Reiser 1994).) This table shows the results of three variants playing against pure (iterative) $\alpha\beta$ search. The first column gives the percentage of wins for each variant in these games. The data in the *Significance* column are based on the *sign test*. The null hypothesis is that the algorithms are equally good. These data signify the probability that the result is from chance fluctuation. The third column gives the time consumption relative to the pure search approach. "Forward Pruning" wins 83.33%, but this result is less significant than that of "Fastness" (77.78%) because more games were drawn.

Algorithm	wins n%	Signifi-cance	needs n% of time
Switch	58.33	0.06332	81.8
Forward Pruning	83.33	0.01046	100.0
Fastness	77.78	0.00921	40.82

Table 1: Selected Results of our Experiments

The "Switch" algorithm used here is an instantiation of the one described above that switches to the $\alpha\beta$ search whenever a tactical move is possible. More precisely, it does so whenever a pushing move is possible in the actual position. It wins in 58.33 of the cases versus the pure search approach, though using less time.

"Forward pruning" only searches the steps with immediate material change, and those recommended by the case base. The latter ones are all steps where the product of similarity and value of the case is higher than for the current position, i.e., improvements according to the knowledge in the case base. The highly significant win of this approach vs. pure search indicates the potential of such combinations of case-based reasoning with search.

The table entry "Fastness" shows the result of having the pure case-based algorithm play vs. the pure search algorithm, when the latter may only use 2 seconds instead of 10 seconds as in all the other experiments. Still, the case-based algorithm is faster in our environment. The highly significant result indicates the power of our case-based reasoning approach in situations with small and strict time limits.

Related Work

Well-known CBR Systems use flat memory (Hypo (Ashlez & Rissland 1988)), shared feature network (MicroMOPs (Kolodner & Riesbeck 1989)), priorized discrimination nets (CHEF (Hammond 1989)) or redundant discrimination nets (Julia (Hinrichs & Kolodner 1991)) as memory organization for the case library together with some sort of indexing for retrieval. In these kinds of case-base organization all cases can be equally reached.

Work done for multi-step problems in the current case-based reasoning research can be found, e.g., in (Zito–Wolf & Alterman 1993). This paper introduces multi-cases as memory (space) optimization of micro-cases (see (Goodman 1991)). Both micro and multi-cases are designed for "procedural knowledge" (as multi-step problems are called there). Every case is reachable in retrieval for every problem.

Our case-base memory organization is problem oriented. In multi-step problems it is not necessary to reach all cases, but using the similarity relations and the history pointer, relevant cases can be found easily with less effort.

In CBR Systems where multiple cases are used for one request (e.g., (Sycara & Navinchandra 1991)) the problem of splitting these cases for adaptation exists. We reason from multiple cases, but no splitting is done. The cases have weights and the step is selected which leads to the state most similar to the worthiest case.

CBR integrated with heuristic search can be found in (Lehnert 1987, Bradtke & Lehnert 1988). Case bases organized in discrimination nets and equivalence classes are used for forward pruning. In our work, where the cases are connected by similarity links, forward pruning is one of several possibilities to use the case base and similarity metric. We developed a pure case-based algorithm and combined it with other ways of heuristic search. Since we use a similarity metric instead of equivalence classes, the density of the case base is less important, and by combining the similarity value with the value of the case we have the possibility of explicitly representing the importance of certain cases. In (Bradtke & Lehnert 1988) neither a possibility of assigning "importance" to the cases can be found nor a possible use of the case base for other algorithms than heuristic search. Furthermore, it is not described, how a case base is designed. Our case-base algorithm can also work by itself, and the case base used was generated automatically.

Conclusion

In summary, we developed a novel approach for applying case-based reasoning to multi-step problems. In particular, we designed a new memory organization for this class of problems. Our approach avoids the compatibility issue of index structure and similarity metric in only using a similarity metric. Although it may be possible to view our approach as a form of *caching*, the use of a similarity metric and the combination of similarity value and case value makes it a form of reasoning based on cases.

Moreover, we integrated case-based reasoning in several ways with heuristic search. The results of our experiments show a statistically significant improvement through combination of case-based reasoning with search over pure search or pure case-based reasoning.

Our approach does not even require knowledge about complete solutions. In real-world problems, complete solutions are often unavailable, and just an appropriate next step is selected. Our algorithm selects such a step based on knowledge about preferable cases and their similarity to the achievable states in the given situation.

Similarity links and values organize the case base in a kind of subgoal structure, that is used for the selection of the next step. However, the cases are not subgoals in the usual sense of planning. They are more like prototypes of achievable states.

We primarily studied case-based reasoning in the context of two-player games. However, the approach appears to be general enough to be also applicable to single-agent problem solving. In particular, we propose it for real-time processing.

The applicability of our algorithm for real-time processing in environments of realistic size still has to be evaluated. Furthermore, it has to be focused on the generation of real-world case bases and the testability for applications in a safety critical real-time system.

Acknowledgments

We would like to thank Wilhelm Barth, Atilla Bezirgan, Helmut Horacek, Christian Koza and Stefan Kramer for comments on earlier versions of this paper.

Appendix: The Game Abalone

Abalone is a two-player game with perfect information like, e.g., chess or checkers, developed by the French computer scientists Michel Lalet and Laurent Levi. It focuses on the idea of synergy.

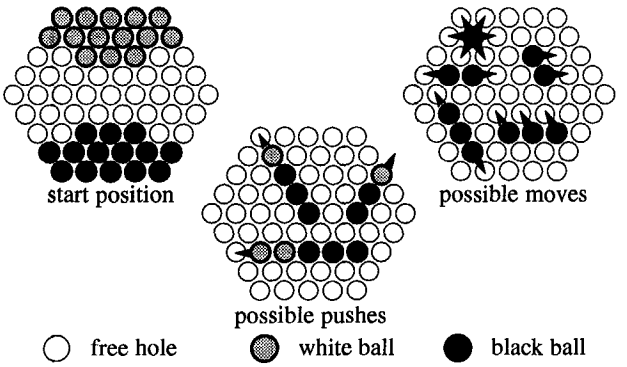

Figure 2: Abalone

Rules

1. The player who first pushed 6 balls of the opponent out of the board wins.
2. Each player performs one step in turn.
3. In one step 1, 2, or 3 balls of the player may be moved according to Figure 2 "possible moves".
4. In one step 1 or 2 opponent's balls may be pushed if they are in one line with the player's balls, the player has more balls in the line and the hole behind the opponent's ball(s) is empty, or the ball is pushed out of the board (see Figure 2 "possible pushes").
5. The opponent's balls do not necessarily have to be pushed or pushed out of the board.

For a detailed description of the rules of Abalone see (Abalone 90).

Special Characteristics

Two special characteristics of the game Abalone are important for our experiments. The "branching degree" of the game in a typical position is between 50 and 120 different moves, and "strategy" is more important than "tactics".

References

Abalone s. a. 1990. *The game Abalone: Rules.* Abalone s. a., Z. I. de la Bonde, 91300 Massy, France.

Ashlez, K. D.; Rissland, E. L. 1988. Compare and Contrast, A Test of Expertise. In *Proceedings of a Workshop on CBR (DARPA)*, Morgan Kaufmann Publishers Inc: 31–36.

Bradtke, S.; Lehnert, W. G. 1988. Some Experiments With Case-Based Search. In *Proceedings of a Workshop on CBR (DARPA)*, Morgan Kaufmann Publishers Inc: 80–92.

Goodman, M. 1991. A case-based, inductive architecture for natural language processing. In *AAAI Spring symposium on Machine Learning of Natural Languagev and Ontology.*

Hammond, K. J. 1989. Case-Based Planning, Viewing Planning as a Memory Task. In *Perspectives in Artificial Intelligence*, Academic Press, Inc.

Hinrichs, T. R.; Kolodner, J. L. 1991. The Roles of Adaptation in Case-Based Design. In *Proceedings of the 11th National Conference on AI (AAAI–91).*

Kaindl, H. 1983. Searching to Variable Depth in Computer Chess. In *Proceedings Eighth International Joint Conference on Artificial Intelligence (IJCAI–83)*, Karlsruhe, Los Altos, Calif.: Kaufmann: 760–762.

Kaindl, H. 1990. Tree Searching Algorithms. In *Computers, Chess, and Cognition* (T. A. Marsland and J. Schaeffer, Eds.), New York: Springer-Verlag: 133–158.

Kaindl, H.; Scheucher, A. 1992. Reasons for the Effects of Bounded Look-Ahead Search. *IEEE Transactions on Systems, Man, and Cybernetics* SMC–22(5): 992–1007.

Kaindl, H.; Shams, R.; Horacek, H. 1991. Minimax Search Algorithms with and without Aspiration Windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI–13(12):* 1225–1235.

Kolodner, J.; Riesbeck, C. 1989. Case-Based Reasoning. *Tutorial Notes at the 11th International Joint Conference on Artificial Intelligence (IJCAI–89).*

Lehnert, W. G. 1987. *Case-Based Reasoning as a Paradigm for Heuristic Search.* COINS 87–107, Department of Computer and Information Science, University of Massachusetts at Amherst, Amherst, MA.

Reiser, C. 1994. *Case-Based Reasoning für mehrstufige Probleme und die Integration mit heuristischer Suche.* Doctoral Dissertation, Technical University of Vienna, Vienna, Austria, April.

Stankovic, J. A. 1988. *Misconceptions about Real-Time Computing: A Serious Problem for Next-Generation Systems.* IEEE Computer Society Press, Washington, D.C., USA.

Sycara, K. P.; Navinchandra, D. 1991. Influences: A Thematic Abstraction for Creative Use of Multiple Cases. *Proceedings of a Workshop on CBR (DARPA)*, Morgan Kaufmann Publishers Inc: 133–145.

Zito-Wolf, R.; Alterman, R. 1993. A Framework and an Analysis of Current Proposals for the Case-Based Organization and Representation of Procedural Knowledge. *Proceedings of the 11th National Conference on AI (AAAI–93):* 73–78.

105