Evaluation of two Strategies for Case-Based Diagnosis handling Multiple Faults

Martin Atzmueller, Joachim Baumeister, Frank Puppe

Department of Computer Science
University of Würzburg, 97074 Würzburg, Germany
email: {atzmueller, baumeister, puppe}@informatik.uni-wuerzburg.de

Abstract: Case-based diagnosis handling multiple faults is still a challenging task. In this paper we present methods for handling multiple faults, embedded in the standard CBR cycle. The approaches extend it by case decomposition and combination strategies which generate candidate cases for case reuse. The context of our work is to supplement a medical documentation and consultation system by CBR techniques which facilitate the extended retrieval of experiences for experience management. We present an evaluation of our approaches with a real-world case base.

1 Introduction

Case-based diagnosis reduces knowledge acquisition and maintenance costs considerably compared to model-based approaches and has therefore become quite popular in experience-rich domains. However, handling multiple faults is a major problem: in our domain of sonography the examination considers several partially disjunctive subdomains, e.g. liver or kidney, which results in multiple faults, i.e. most cases contain multiple diagnoses. Our goal is to extend a medical documentation and consultation system with a component for experience management, which enables the retrieval of experiences such as explanations for a query case based on the presented similarity to former cases and additional information contained in these, e.g. therapy, complications, prognosis or the treating physician as contact person for special questions. To obtain this information we use case-based reasoning. Due to the combinatorial rules the chance to reuse a case with even 3 independent diagnoses from say 100 alternatives is roughly just one to one million. However, that many cases are rarely available. Therefore, since an average of about 6.8 diagnoses per case were found, naive case-based diagnosis performed very poor.
A closer look at other diagnostic problem solving methods like heuristic or set-covering approaches being able to handle multiple faults in medical domains [BEF+02] shows, that they exploit independence assumptions about the domain to reduce the combinatorial search space. The major ideas are:

1. Decomposing complex cases into several simpler cases
2. Finding solutions for the simple cases
3. Combining the cases with their solutions

The difficult task is decomposition. We propose two approaches: The first one uses static decomposition, which can be largely precomputed at compile time. The second one employs dynamic decomposition: it constructs partitions at run time. The static decomposition method takes advantage of the fact that many domains can be divided in rather independent subdomains, e.g. in the medical domain according to the different organ systems (i.e. liver, kidney etc. in our domain of sonography). Since a complete separation is usually not possible, overlapping observations and diagnoses are assigned to all subdomains.

If a static decomposition is impossible, then a dynamic approach is necessary: We adopt a set-covering strategy for dealing with multiple faults: Each diagnosis (fault) covers some observations and all diagnoses should be able to cover all observations. Since pure case-based reasoning has just the notion of similarity but not of set-covering, a combination of both approaches is necessary. Therefore, we learn diagnostic profiles from cases and use these within the set-covering approach to diagnose multiple faults and check the proposed solution by composing cases covering all diagnoses. We have implemented both extensions to case-based diagnosis and evaluated them with 744 cases from the sonography domain. First results show, that both approaches are able to handle the multiple fault problem.

The rest of the paper is organized as follows: In Section 2 we give essential basic definitions for case-based diagnosis. Furthermore, we introduce a conceptual process model of case-based diagnosis handling multiple faults. In Section 3 we describe the enhanced CBR retrieve and reuse step of the process model in more detail. An evaluation with a real-world case base is given in Section 4. We will conclude the paper in Section 5 with a discussion of the presented work and we show promising directions for future work.

2 Case-Based Diagnosis with Multiple Faults

In the following, we will first define the knowledge representation schema and discuss the similarity metric for comparing cases. After that, we will outline the process model which describes the case-based diagnosis process with multiple faults.

Basic Definitions Let Ω_D be the set of all diagnoses and Ω_A the set of all attributes. To each attribute $a \in \Omega_A$ a range $dom(a)$ of values is assigned. Further we assume Ω_F to be the (universal) set of findings $(a = v)$, where $a \in \Omega_A$ is an attribute and $v \in dom(a)$ is an assignable value. Let CB be the case base containing all available cases that have been solved previously. A case $c \in CB$ is defined as a tuple

$$c = (F_c, D_c, I_c) \tag{1}$$

where $F_c \subseteq \Omega_F$ is the set of findings observed in the case c. In CBR-problems these findings are commonly called *problem description*. The set $D_c \subseteq \Omega_D$ is the set of diagnoses describing the *solution* for this case. I_c contains additional information like therapy advices or prognostic hints.

Similarity Metric To compare the similarity of a query case c with another case c', we apply the commonly used weighted similarity measure given in Equation 2. This measure is an adaptation of the *Hamming distance* with weights and partial similarities between values of attribute a (e.g. see [BKI00], page 183f.):

$$sim(c, c') = \frac{\sum\limits_{a \in \Omega'_A} w(a) \cdot sim_a\big(v_c(a), v_{c'}(a)\big)}{\sum\limits_{a \in \Omega'_A} w(a)} \tag{2}$$

where $v_c(a)$ is a function, which returns the value of the attribute a in case c and $w(a)$ is the weight of attribute a. If weights for attributes are not available, then we set $w(a) = 1$ for all $a \in \Omega_A$. $sim_a\big(v_c(a), v_{c'}(a)\big)$ is a function which returns the similarity between two values $v_c(a)$ and $v_{c'}(a)$ of attribute a.
We consider the attributes contained in the the union of the defined attributes of both cases:

$$\Omega'_A = \big\{a \in \Omega_A \,|\, \exists f \in F_c \cup F_{c'} : f = (a = v), v \in dom(a)\big\}$$

If an attribute is only defined in one case then we treat this as an 'unknown' attribute value and give a default similarity of $sim_a(.,.) = 0.1$.

A Process Model for Case-Based Diagnosis handling Multiple Faults In [AP94], Aamodt and Plaza defined the case-based reasoning process as a cycle containing the following four sub-processes: *Retrieve*, *Reuse*, *Revise*, *Retain*. Typically starting with a (partial) problem description of observed findings the retrieve process ends with a previously solved case which matches the problem description of the query case best.

We say, that a case is *sufficiently similar* to another case, if the similarity between these two cases exceeds a given (and usually high) threshold $\mathcal{T}_{CBR}$. In [BAP02] we have described appropriate methods to learn similarities and weights from cases which we can apply for standard CBR. If we cannot retrieve a sufficiently similar case by standard CBR, then we rely on an adapted *Retrieve* and *Reuse* process, which is illustrated in Figure 1.

1. Apply standard CBR (using Equation 2) utilizing the learned knowledge [BAP02].
2. If *sufficiently similar* cases have been found, then reuse the solutions of the k best cases.
3. Otherwise, if no case is sufficiently similar:
   (a) Generate a set Ca of *candidate cases* according to a *candidate case generation strategy* (cf. Section 3)
   (b) Using Equation 2 select the k most similar cases to the query case c, which are also sufficiently similar: $Ca^k \subseteq \{ca \in Ca \,|\, sim(ca, c) \geq \mathcal{T}_{CBR}\}$. Return these as k possible solutions for the given problem description

A *candidate case* is created from a set of subcases which are merged to one case. We will discuss candidate case generation strategies in Section 3 in more detail.

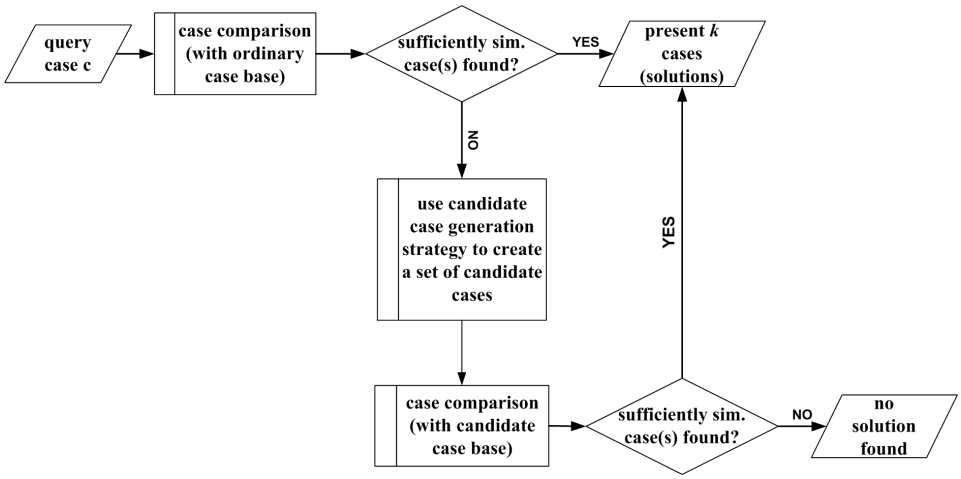

Figure 1: Process model for CBR handling multiple faults using candidate case creation

Definition 1 (Candidate Case). *A candidate case*

$$ca = (F_{ca}, D_{ca}, C_{ca})$$

consists of a set of subcases $C_{ca} \subseteq CB_{ca}$ of a given case base CB_{ca} and a set of diagnoses $D_{ca} \subseteq \Omega_D$. The set of findings $F_{ca} \subseteq \Omega_F$ of the candidate case is created by joining the findings of the subcases as described below. The set D_{ca} is defined as the union of the subcases' diagnoses, i.e. $D_{ca} = \bigcup_{c \in C_{ca}} D_c$.

To create candidate cases we have to combine the problem descriptions and the solutions included in the subcases of the candidate case. However, when combining problem descriptions, i.e. sets of findings, conflicts can arise if two cases contain different values for the same attribute. For the creation of candidate cases we apply the following algorithm:

1. Joining sets of the solutions contained in the subcases.
2. Joining problem descriptions of the subcases: $F_{ca} = \bigcup_{c \in C_{ca}} F_c$
3. Conflict resolution: If F_{ca} contains more than one finding for an attribute:
   (a) If background knowledge is available, then we use it (see below).
   (b) Otherwise we try to select an attribute value based on the finding in the query case:
      i. We choose the value contained in the query case if included in one subcase.
      ii. Alternatively, we select the value which is most similar to the value included in the query case.
      iii. Otherwise, if the value is not included in the problem description of the query case, then we randomly pick a value from the set of the attribute's values of the subcases. For ordinal or numerical findings we could use the mean of the values, alternatively.

If no additional knowledge is available the conflict resolution strategy above is motivated by the fact, that we want to obtain a candidate case explaining the findings in the query case. So, when choosing between different values for an attribute, we base this decision on the value which is contained in the query case. If the query case does not contain the conflicting attribute, we do not know, which value is relevant for the candidate case, so we can pick one randomly.
However, the conflict resolution step above can be enhanced using additional knowledge, i.e. *abnormalities*. Abnormality knowledge specifies which findings represent a normal or an abnormal state of their corresponding attribute (e.g. *pain=none* is normal, whereas *pain=high* is abnormal). If abnormalities are defined, then we take the value with the highest abnormality. We motivate this approach by a heuristic explained with the following example: If a patient has two (independent) diagnoses, then it seems to be reasonable that the more severe finding will be observed, e.g. *pain=high* from one diagnosis rather than *pain=none* from another diagnosis.

3 Strategies for Candidate Case Generation

In Section 2 we have discussed how to solve cases containing multiple faults which cannot be solved by standard CBR techniques. For these, we propose a special retrieve and reuse method based on candidate case creation strategies. There are two approaches for generating candidate cases. The first approach uses partitioning knowledge provided by the expert to split cases into several parts. Decomposed cases are retrieved and combined to candidate cases. The second approach does not need additional background knowledge, since it uses learned diagnostic profiles to build set-covering models. So, if the static decomposition method is not applicable, i.e. if necessary partitioning knowledge is not available, then we can always rely on the dynamic approach utilizing learned set-covering knowledge.

3.1 Candidate Case Generation using Partition Class Knowledge

For the first candidate case generation strategy, the expert can provide *partition class* knowledge describing how to divide the set of diagnoses and attributes into partially disjunctive subsets, i.e. partitions. These subsets correspond to certain problem areas of the application domain. For example, in the medical domain of sonography, we have subsets corresponding to problem areas like *liver*, *pancreas*, *kidney*, *stomach* and *intestine*. The idea of using partition class knowledge is to split the original case base into several case bases containing partial, decomposed cases corresponding to different partition classes. This decomposition is static and therefore can be precomputed at compile-time. According to the given subsets of attributes and diagnoses of each partition class we can split cases into subcases, where a case is divided by forming sets of attributes and diagnoses for each partition class.

Definition 2 (Partition Class). *A partition class pc is defined as a tuple*

$$pc = (D_{pc}, F_{pc})$$

where $D_{pc} \subseteq \Omega_D$ and $F_{pc} \subseteq \Omega_F$. For one partition class pc the sets D_{pc} and F_{pc} refer to the same problem area of the application domain. All diagnoses are covered by the different partition classes pc_i, i.e. $\Omega_D = \bigcup_i D_{pc_i}$.

Candidate Case Generation By recombining partial cases to candidate cases, we find possible solutions, i.e. most similar cases for the query case. We obtain these partial cases by first decomposing the query case into several subcases. Then we apply CBR for each query subcase, retrieve a most similar partial case, and finally recombine the retrieved solutions. We will outline the process in the following:

1. Precomputed: Divide the original case base into several 'partition class' case bases.
2. Decompose the query case into a set of partial cases according to the given partition classes.
3. For each partial case, apply standard CBR using the respective partitioned case base saving the set of most similar cases for each partial case in result sets.
4. Generate candidate cases by combining the result sets: Construct a set of subcases, for which one case from each result set is drawn. Create a candidate case using these subcases.

We basically follow a divide-and-conquer strategy. We decompose cases into subcases, which may even contain only a single diagnosis. Then we recombine the partial solutions into the general solution, which is represented by the candidate case. However, we have to make sure that the decomposed cases are still meaningful. Firstly, they must contain a minimum number of attributes to guarantee a certain support for the diagnoses contained in the case. Secondly, cases should still contain diagnostic information, i.e. they should contain at least one diagnosis. If a generated subcase does not fulfill these requirements, then it is not considered and removed.

Related Work In the literature the combination of standard case-based reasoning retrieval with special reuse, e.g. adaptation techniques has already been investigated in several approaches. Watson and Perera [WP98] presented a system, which retrieves decomposed cases from a case base made up of hierarchically structured cases, and adapts multiple sub-cases into a solution semi-automatically. The *CADSYN* [MZ91] system recombines sub-problems formed of decomposed cases into a solution taking the context of such a partial case into account. Smyth and Cunningham [SKC01] presented case-based reasoning on hierarchical case-bases, which allows complex problems to be solved by reusing multiple cases at various levels of abstraction. These approaches apply mainly techniques from case-based planning, i.e. utilizing hierarchical relations either in the CBR retrieve or reuse step, or both. In contrast, our candidate case generation strategy using partition class knowledge is a knowledge intensive approach, which uses the explicit partitioning information as background knowledge to decompose cases.

3.2 Candidate Case Generation using Set-Covering Models

If the partition class approach is not feasible for the given domain, we propose a more general dynamic approach for candidate case creation utilizing learning methods. In [BAP02] we showed how to learn diagnostic profiles which are transformed to set-covering models. This set-covering knowledge is used to generate hypotheses, i.e. a set of diagnoses, which can explain the problem description of the query case. These hypotheses are used in a combined *Retrieve* and *Reuse* step to generate candidate cases.

Set-Covering Models A *set-covering model* contains set-covering relations, which describe relations like: *A diagnosis d predicts that the finding f is observed in $freq_{f,d}$ percent of all cases."* We denote a set-covering relation by $r = d \rightarrow f\ [freq_{f,d}]$. Originally, plain set-covering models were introduced by [RNW83]. Due to the limited space we refer to [BSP01, BS02] for a more detailled introduction into set-covering models using additional knowledge, like weights and similarities. As additional knowledge we can use the inductively learned knowledge which we use for standard CBR as well. In [BAP02] we showed how diagnostic profiles are transformed to set-covering models. For short we add a set-covering relation $r = d \rightarrow f\ [freq_{f,d}]$ for all findings f that occur in a given diagnostic profile for a diagnosis d, to the set-covering knowledge base.

Candidate Case Generation The strategy for creating candidate cases with set-covering models uses these models to generate hypotheses, i.e. sets of diagnoses, that represent an explanation for the query case. Given a hypothesis we combine cases so that the union of their solution parts has a high coverage of the hypothesis. When generating candidate cases, we first construct a candidate case hypothesis, which restricts the number of subcases included.

Definition 3 (Candidate Case Hypothesis). *A candidate case hypothesis*

$$ca^n = (F_{ca}, D_{ca}, C_{ca})$$

of size n is a candidate case with not more than n subcases included, i.e. $|C_{ca}| = n$.

For defining the quality of a candidate case, we applied a metric introduced by Thompson and Mooney [TM94]. We call this metric *intersection coverage*. Intuitively, it measures the size of the intersection between the hypothesis' diagnoses and the candidate case's diagnoses compared to the size of the diagnoses. The metric is defined as follows:

Definition 4 (Intersection Coverage). *The intersection coverage of a candidate case reflects, the degree of coverage between its set of diagnoses $D_{ca} \subseteq \Omega_D$ and the hypothesis' set of diagnoses $D_h \subseteq \Omega_D$. The intersection coverage ic is defined by:*

$$ic(D_{ca}, D_h) = \frac{1}{2} \cdot \left(\frac{|D_{ca} \cap D_h|}{|D_h|} + \frac{|D_{ca} \cap D_h|}{|D_{ca}|} \right) \tag{3}$$

where D_{ca} is the set of diagnoses of the candidate case, and D_h is the set of diagnoses included in the hypothesis.

We say, that a candidate case $ca = (F_{ca}, D_{ca}, C_{ca})$ *completely covers* a hypothesis D_h iff $D_{ca} = D_h$. It is obvious that $ic(D_{ca}, D_h) = 1$ iff ca completely covers D_h. We use this quality measure when combining partial cases to construct candidate cases, and when ranking the best candidate cases which we return in the candidate case generation strategy using set-covering knowledge. It is easy to see that we are interested in candidate cases with a coverage of a given hypothesis as high as possible, because such candidate cases can explain the hypothesis best.
The strategy for candidate case creation using set-covering models is outlined below:

1. We use the transformed set-covering models to compute the l best hypotheses. A hypothesis is a set of diagnoses that can explain the problem description of the query case, i.e. the observed findings.
2. Given the hypotheses, we generate a set of *candidate cases*: We construct candidate case hypotheses first, restricting their size to a maximum number of subcases. The candidate case generation process is guided by the idea, that a candidate case should contain subcases whose combined solutions should cover the diagnoses of a given hypothesis according to the intersection coverage ic.
3. We rank the candidate cases given their corresponding hypotheses using the intersection coverage (ic) metric.
4. We return the m best candidate cases, i.e. the cases that fit the generated hypotheses best as the candidate case set.

It is obvious that this strategy relies mainly on automatic learning methods, in contrast to the alternative strategy for candidate case creation which we discussed in the previous section.

Related Work The combination of case-based reasoning with abductive or set-covering techniques has already been investigated in several approaches. The systems CASEY [Kot88] and ADAPtER [PT95] are prominent examples that use case-based reasoning for case retrieval. Abductive knowledge is applied to adapt old cases and give a verbose explanation for the adaptation. In our work we use the reverse approach, when using abductive reasoning for guiding the search of how to combine cases. Schmidt et al. [SPG99] considered a simple generation process of prototypes from cases with the ICONS project. Their system uses these prototypes for retrieval and adaptation of query cases. Prototypes are made up of cases, where new prototypes are generated, if new cases do not fit in existing ones. Work on handling multiple faults applying set-covering models is another aspect of the work presented here. The *LAB* algorithm by Thompson and Mooney [TM94] is an inductive learning algorithm, that generates set-covering relations from cases containing multiple faults, but no additional knowledge is applied later.

So, in contrast to other systems, which use abductive reasoning mainly for explanation, we apply a combined retrieve and reuse step guided by set-covering knowledge. The candidate case generation strategy using set-covering models applies a tight coupling of CBR and set-covering knowledge. The set-covering knowledge is learned automatically, and we are also able to include additional knowledge like partial similarities and feature weights in this process.

4 Evaluation

In this section we will first describe the case base we applied for the evaluation. Then we will discuss evaluation methods in multiple fault problems. After that, we will present our results and discuss these in detail. For a more detailed evaluation, which also includes learned background knowledge, we refer to [Atz02].

Properties of the Case Base For our evaluation we applied a real-world case base which is based on the knowledge-based documentation and consultation system for sonography SONOCONSULT, an advanced and isolated part of HEPATOCONSULT [BEF+02]. For the development of HEPATOCONSULT the shell-kit *D3* [Pup98] was applied directly by domain experts. The quality of the derived diagnoses usually is very good, i.e. the solutions are correct in nearly all cases.

Currently, we are using a part of the SONOCONSULT case base containing 744 cases. The case base contains an overall number of 221 diagnoses and 556 symptoms, with a mean $M_d = 6.71 \pm 04.4$ of diagnoses per case and a mean $M_f = 48.93 \pm 17.9$ of relevant findings per case.

Evaluating Accuracy in Multiple Fault Problems When evaluating the accuracy of the learned knowledge, we want to obtain a quantitative measure of the accuracy of the system's diagnoses. For single category problems a case is correctly classified, if its diagnosis matches the correct diagnosis. However, for cases with multiple-diagnoses each case is a positive or negative example for several diagnoses. Thompson and Mooney [TM94] propose the *Intersection Accuracy* as a measure for multiple fault problems:

Definition 5 (Intersection Accuracy). *The intersection accuracy $\mathcal{IA}(c, c')$ is defined as*

$$\mathcal{IA}(c, c') = \frac{1}{2} \cdot \left(\frac{|D_c \cap D_{c'}|}{|D_c|} + \frac{|D_c \cap D_{c'}|}{|D_{c'}|} \right) \tag{4}$$

where c and c' are two cases, $D_c \subseteq \Omega_D$ is the set of diagnoses of case c, and $D_{c'} \subseteq \Omega_D$ is the set of diagnoses contained in case c' likewise.

Intersection accuracy is derived by the two standard measures: *sensitivity* and *precision*, where it is just the average of sensitivity and precision. Essentially the intersection accuracy metric is equal to the intersection coverage defined in equation 3. However, since the semantics are slightly different, we define intersection accuracy as a metric for comparing solutions of cases while intersection coverage relates a solution of a case and a hypothesis.

The SONOCONSULT knowledge base contains hierarchical relationships between diagnoses, so it is possible to exploit these in accuracy assessment, e.g. differentiate between coarse diagnoses and finer-grained diagnoses. However, since the hierarchies in the SONOCONSULT knowledge base are not constructed uniformly, we essentially only use direct parent – child relationships, e.g. when a parent diagnosis p is present in one case and a child c of p in the other, we count this as a reduced match $match(p, c) = 0.5$. However, this situation occurred only in a minority of cases.

Experimental Results and Discussion Initially, we performed an analysis to find out, how many cases can be solved by a perfect case-based reasoning method. So, we compared each query case in the case base with its most similar case that exists in the case base considering only the solution part. We 'found' the perfect matching case by searching for a compare case with the highest intersection coverage of its diagnoses with the diagnoses of the query case. Performing this experiment, it turned out, that in principle all 744 cases are solvable with an intersection accuracy of 90%. However, the analysis showed that due to the multiple fault characteristic of our case base, intersection accuracy did not necessarily correlate with similarity. E.g., there are a lot of cases with an intersection accuracy of the matching case around 85%, while the similarity between the two cases is below 20%, i.e. cases with similar diagnoses may have quite different findings.

In the following table we present results of the experiments E0, E1 and E2. Before evaluating our methods, we performed an experiment E0 which applied a standard CBR approach using ordinary case-comparison. E1 and E2 illustrate the performance of the two hybrid methods introduced in Section 3. E1 uses the set-covering method, with the learned set-covering models, whereas E2 uses partition class knowledge provided by the expert. For these two methods we use all the available (learned) knowledge e.g. learned similarities and weights as discussed in [BAP02].

For the evaluation of our experiments we adopted the *intersection accuracy* measure, described in Section 4. We used leave-one-out cross-validation [Mit97] which is a variation of k-fold cross validation, where each fold consists only of exactly one case. We say that a compare case c' *solves* a query case c, iff $sim(c, c') \geq \mathcal{T}_{CBR}$, i.e. if the cases are sufficiently similar. Cases below this threshold were withdrawn and marked as *not solvable*. The more advanced strategies for candidate case generation enable us to decrease the case similarity threshold $\mathcal{T}_{CBR}$ without receiving a dramatically decreased intersection accuracy of the solved cases. In addition to the mean accuracy (*mean acc*) of all cases, we have also included the mean accuracy (*mean (40) acc*) of the 40 'best' cases, i.e. the cases with the highest intersection accuracy.

		CB (744 cases)			
	used knowledge / method	*threshold* $\mathcal{T}_{CBR}$	*solved cases*	*mean (40) acc*	*mean acc*
E0	no knowledge	0.78	33 (4%)	0.72	0.73
E1	set-covering strategy	0.54	443 (60%)	0.83	0.70
E2	partition class strategy	0.40	635 (85%)	0.88	0.78

Table 1: Results of Experiments E0, E1, E2 (Analysis using 744 cases)

E0 shows that the standard CBR method is performing poor for cases with multiple faults. Standard CBR utilizing no additional background knowledge can only solve 4% of the cases in the case base which is clearly insufficient. E1 performs acceptable since it can return 443 cases as solved, i.e. it can solve about 60% of the cases in the case base with a mean accuracy of 70%. This means a dramatic improvement compared to standard CBR-methods performed on cases with multiple faults. This conclusion strongly motivates the

usage of set-covering techniques in the case-based process. Experiment E2 using partition class knowledge is even better since it can solve about 85% of the cases in the case base with a mean accuracy of 78%. It is obvious that the partition class based strategy can deal with the multiple fault problem quite well.

The results presented above are quite promising. Nevertheless, we see enhancements for the number of solved cases and intersection accuracy when applying refined partition class knowledge. Also, refining the used set-covering models using quality measures will certainly improve the results for E1.

5 Conclusion and Outlook

Our context was to supplement a medical documentation and consultation system by CBR techniques for experience management to enable the extended retrieval of experiences. In this context, standard CBR showed not to be appropriate for handling multiple faults. We presented two approaches which significantly improved the handling of multiple faults. We arrived at a dynamic retrieve and reuse process that does not leave the paradigm of case-based reasoning, since it always explains its presented solutions in parts of cases. For this process, we combined dynamic case reuse strategies with common case-based techniques to form a new hybrid method for CBR. This method was especially well suited for the problem of handling cases with multiple faults. Additionally, the system uses various learning methods to acquire and integrate helpful background knowledge for improving its performance [BAP02]. One approach for the reuse process utilizing candidate cases uses set-covering knowledge to create hypotheses that guide the retrieve and reuse process. The alternative strategy uses background knowledge to decompose cases, exploiting independent assumptions of the domain and produced very promising results. The decomposition of cases can be precomputed in advance at compile-time. So, this strategy is more efficient concerning run-time than the strategy based on set-covering knowledge.

For future work we see room for improvements in several directions: Considering dependencies between features or diagnoses, when building the diagnostic profiles and set-covering models can enhance the quality of the models. If dependencies are provided by the expert, then this can further enhance the quality of the models if taken into account properly. Furthermore, fine-tuning partition class knowledge, learning partition classes automatically, and looking at the combine step of the candidate cases more closely, provide promising opportunities here. For example, techniques that evaluate the quality of the candidate cases may prove essential to ensure a certain quality of the generated candidate cases. So, methods which can evaluate combinations of cases, e.g. using Bayesian networks [HBR02], are a promising direction. An integration of both candidate case generation strategies could be a worthwhile approach, too. Cases could be decomposed into smaller problems using partition knowledge in a first step. The candidate case creation strategy using set-covering knowledge could be applied in a second step to produce solutions for the subproblems which can then be combined using the partition class strategy.

References

[AP94] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.

[Atz02] Martin Atzmueller. Semi-Automatic Data Mining Methods for Improving Case-Based Reasoning. Master's thesis, University of Wuerzburg, Department of Computer Science VI, 2002.

[BAP02] Joachim Baumeister, Martin Atzmueller, and Frank Puppe. Inductive Learning for Case-Based Diagnosis with Multiple Faults. In *Advances in Case-Based Reasoning*, volume 2416 of *LNAI*, pages 28–42. Springer-Verlag, Berlin, 2002. Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR-2002).

[BEF+02] Hans Peter Buscher, Ch. Engler, A. Fuhrer, S. Kirschke, and Frank Puppe. Hepato-Consult: A Knowledge-Based Second Opinion and Documentation System. *Artificial Intelligence in Medicine*, 24:205–216, 2002.

[BKI00] Christoph Beierle and Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme. Grundlagen, Algorithmen, Anwendungen*. Vieweg, 2000.

[BS02] Joachim Baumeister and Dietmar Seipel. Diagnostic Reasoning with Multilevel Set–Covering Models. In *Proceedings of the 13th International Workshop on Principles of Diagnosis (DX-02)*, Semmering, Austria, 2002.

[BSP01] Joachim Baumeister, Dietmar Seipel, and Frank Puppe. Incremental Development of Diagnostic Set–Covering Models with Therapy Effects. In *Proceedings of the KI-2001 Workshop on Uncertainty in Artificial Intelligence*, Vienna, Austria, 2001.

[HBR02] Daniel N. Hennessy, Bruce G. Buchanan, and John M. Rosenberg. Bayesian Case Reconstruction. In *Advances in Case-Based Reasoning*, volume 2416 of *LNAI*, pages 148–158. Springer-Verlag, Berlin, 2002. Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR-2002).

[Kot88] Phyllis Koton. Reasoning about Evidence in Causal Explanations. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 256–261, 1988.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill Comp., 1997.

[MZ91] M.L. Maher and D.M. Zhang. CADSYN: Using Case and Decomposition Knowledge for Design Synthesis. *Artificial intelligence in Design*, 1991.

[PT95] Luigi Portinale and Pietro Torasso. ADAPtER: An Integrated Diagnostic System Combining Case-Based and Abductive Reasoning. In *Proceedings of the ICCBR 1995*, pages 277–288, 1995.

[Pup98] Frank Puppe. Knowledge Reuse Among Diagnostic Problem-Solving Methods In The Shell-Kit D3. *Int. J. Human-Computer Studies*, 49:627–649, 1998.

[RNW83] James A. Reggia, Dana S. Nau, and Pearl Y. Wang. Diagnostic Expert Systems based on a Set Covering Model. *Journal of Man-Machine Studies*, 19:437–460, 1983.

[SKC01] Barray Smyth, Mark T. Keane, and Padraig Cunningham. Hierarchical Case-Based Reasoning Integrating Case-Based and Decompositional Problem-Solving Techniques for Plant-Control Software Design . *Transactions on Knowledge and Data Engineering*, 13(5):793–812, 2001.

[SPG99] Rainer Schmidt, Bernhard Pollwein, and Lothar Gierl. Case-Based Reasoning for Antibiotics Therapy Advice. In *Proceedings of the ICCBR 1999*, pages 550–559, 1999.

[TM94] Cynthia A. Thompson and Raymond J. Mooney. Inductive Learning for Abductive Diagnosis. In *AAAI, Vol. 1*, pages 664–669, 1994.

[WP98] Ian Watson and Srinath Perera. A Hierarchical Case Representation using Context Guided Retrieval. *The Knowledge Based Systems Journal Vol. 11*, pages 285–292, 1998.