FastDiag: A Diagnosis Algorithm for Inconsistent Constraint Sets

Alexander Felfernig 1, Monika Schubert 1

1 Graz University of Technology, Inffeldgasse 16b, 8010 Graz, Austria
alexander.felfernig@ist.tugraz.at
monika.schubert@ist.tugraz.at

ABSTRACT

Constraint sets can become inconsistent in different contexts. For example, during a configuration session the set of customer requirements can become inconsistent with the configuration knowledge base. Another example is the engineering phase of a configuration knowledge base where the underlying constraints can become inconsistent with a set of test cases. In such situations we are in the need of techniques that support the identification of minimal sets of constraints that have to be adapted or deleted in order to restore consistency. In this paper we introduce a divide-and-conquer based diagnosis algorithm (FastDiag) which identifies minimal sets of faulty constraints in an over-constrained problem. This algorithm is specifically applicable in scenarios where the efficient identification of leading (preferred) diagnoses is crucial. We compare the performance of FastDiag with the conflict-directed calculation of hitting sets and present an in-depth performance analysis that shows the advantages of our approach.

1 INTRODUCTION

Constraint technologies (Tsang, 1993) are applied in different areas such as configuration (Fleischanderl et al., 1998; Mittal and Frayman, 1989; Sinz and Haag, 2007), recommendation (Felfernig et al., 2009), and scheduling (Castillo et al., 2005). There are many scenarios where the underlying constraint sets can become over-constrained. For example, when implementing a configuration knowledge base, constraints can become inconsistent with a set of test cases (Felfernig et al., 2004). Alternatively, when interacting with a configurator application (Felfernig et al., 2009; O’Sullivan et al., 2007), the given set of customer requirements (represented as constraints) can become inconsistent with the configuration knowledge base. In both situations there is a need of an intelligent assistance that actively supports users of a constraint-based application (end users or knowledge engineers).

A wide-spread approach to support users in the identification of minimal sets of faulty constraints is to combine conflict detection (see, e.g., (Junker, 2004)) with a corresponding hitting set algorithm (Reiter, 1987; DeKleer et al., 1992). In their original form these algorithms are applied for the calculation of minimal diagnoses which are typically determined with breadth-first search. Further diagnosis algorithms have been developed that follow a best-first search regime where the expansion of the hitting set search tree is guided by failure probabilities of components (DeKleer, 1990). Another example for such an approach is presented in (Felfernig et al., 2009) where similarity metrics are used to guide the (best-first) search for a preferred (plausible) minimal diagnosis (including repairs). Note that the complement of such a minimal diagnosis can also be denoted as maximally successful sub-query (McSherry, 2004).

Both, simple breadth-first search and best-first search diagnosis approaches are predominantly relying on the calculation of conflict sets (Junker, 2004). In this context, the determination of a minimal diagnosis of cardinality $n$ requires the identification of at least $n$ minimal conflict sets. In this paper we introduce a diagnosis algorithm (FastDiag) that allows to determine one minimal diagnosis at a time with a logarithmic number of consistency checks. The algorithm supports the identification of preferred diagnoses given predefined preferences regarding a set of decision alternatives. FastDiag is boosting the applicability of diagnosis methods in scenarios such as online configuration & reconfiguration (Felfernig et al., 2004), recommendation of products & services (Felfernig et al., 2009), and (more generally) in scenarios where the efficient calculation of preferred (leading) diagnoses is crucial (DeKleer, 1990). FastDiag is not restricted to constraint-based systems but it is also applicable, for example, in the context of SAT solving (Marques-Silva and Sakallah, 1996) and description logics reasoning (Friedrich and Shchekotykhin, 2005).

The remainder of this paper is organized as follows. In Section 2 we introduce a simple example configuration task from the automotive domain. In Section 3 we discuss the basic hitting set based approach to the cal-
culation of diagnoses. In Section 4 we introduce an algorithm (FastDiag) for calculating preferred diagnoses for a given over-constrained problem. In Section 5 we present a detailed evaluation of FastDiag which clearly outperforms standard hitting set based algorithms in the calculation of the topmost-n preferred diagnoses. With Section 6 we provide an overview of related work in the field. The paper is concluded with Section 7.

2 EXAMPLE DOMAIN: CAR CONFIGURATION

Car configuration will serve as a working example throughout this paper. Since we exploit configuration problems for the discussion of our diagnosis algorithm, we first introduce a formal definition of a configuration task. This definition is based on (Felfernig et al., 2004) but is given in the context of a constraint satisfaction problem (CSP) (Tsang, 1993).

Definition 1 (Configuration Task). A configuration task can be defined as a CSP (V, D, C). V = {v1, v2, . . . , vn} represents a set of finite domain variables. D = {dom(v1), dom(v2), . . . , dom(vn)} represents a set of variable domains dom(vk) where dom(vk) represents the domain of variable vk. C = CKB ∪ CR where CKB = {c1, c2, . . . , cn} is a set of domain specific constraints (the configuration knowledge base) that restrict the possible combinations of values assigned to the variables in V. CR = {c3, c4, . . . , cn} is a set of customer requirements also represented as constraints.

A simplified example of a configuration task in the automotive domain is the following. In this example, type represents the car type, pdc is the parc distance control functionality, fuel represents the fuel consumption per 100 kilometers, a skibag allows the ski stowage inside the car, and 4-wheel represents the corresponding actuation type. These variables describe the potential set of requirements that can be specified by the user (customer). The possible combinations of these requirements are defined by a set of constraints which are denoted as configuration knowledge base. CKB = {c1, c2, c3, c4}. Furthermore, we assume the set of customer requirements CR = {c5, c6, c7}.

- V = {type, pdc, fuel, skibag, 4-wheel}
- D = {dom(type)={city, limo, combi, xdrive}, dom(pdc)={yes, no}, dom(fuel)={4l, 6l, 10l}, dom(skibag)={yes, no}, dom(4-wheel)={yes, no}}
- CKB = {c1: 4-wheel = yes ⇒ type = xdrive, c2: skibag = yes ⇒ type ≠ city, c3: fuel = 4l ⇒ type = city, c4: fuel = 6l ⇒ type ≠ xdrive}
- CR = {c5: type = combi, c6: fuel = 4l, c7: 4-wheel = yes}

On the basis of this configuration task definition, we can now introduce the definition of a concrete configuration (solution for a configuration task).

Definition 2 (Configuration). A configuration for a given configuration task (V, D, C) is an instantiation I = {v1=ins1, v2=ins2, . . . , vn=insn} where insk ∈ dom(vk).

A configuration is consistent if the assignments in I are consistent with the ci ∈ C. Furthermore, a configuration is complete if all variables in V are instantiated.

Finally, a configuration is valid if it is consistent and complete.

3 DIAGNOSING OVER-CONSTRAINED PROBLEMS

For the configuration task introduced in Section 2 we are not able to find a solution, for example, a combi-type car does not support a fuel consumption of 4l per 100 kilometers. Consequently, we want to identify minimal sets of constraints (c1 ∈ CR) which have to be deleted (or adapted) in order to be able to identify a solution (restore the consistency). In the example of Section 2 the set of constraints CR = {c5, c6, c7} is inconsistent with the constraints CKB = {c1, c2, c3, c4}, i.e., no solution can be found for the underlying configuration task. A standard approach to determine a minimal set of constraints that have to be deleted from an over-constrained problem is to resolve all minimal conflicts contained in the constraint set. The determination of such constraints is based on a conflict detection algorithm (see, e.g., (Junker, 2004)), the derivation of the corresponding diagnoses is based on the calculation of hitting sets (DeKleer and Williams, 1987; Reiter, 1987). Since both, the notion of a (minimal) conflict and the notion of a (minimal) diagnosis will be used in the following sections, we provide the corresponding definitions here.

Definition 3 (Conflict Set). A conflict set is a set CS ⊆ CR s.t. CKB ∪ CS is inconsistent. A conflict set CS is a minimal if there does not exist a conflict set CS′ with CS′ ⊂ CS.

In our working example we can identify three minimal conflict sets which are CS1 = {c3, c6}, CS2 = {c5, c7}, and CS3 = {c5, c6}.

CS1, CS2, CS3 are conflict sets since CS1 ∩ CKB ∪ CS2 ∩ CKB ∩ CS3 ∩ CKB is inconsistent. The minimal property is fulfilled since there does not exist a conflict set CS4 with CS4 ⊂ CS1 or CS4 ⊂ CS2 or CS4 ⊂ CS3. The standard approach to resolve the given conflicts is the construction of a corresponding hitting set directed acyclic graph (HSDAG) (Reiter, 1987) where the resolution of all minimal conflict sets automatically corresponds to the identification of a minimal diagnosis. A minimal diagnosis in our application context is a minimal set of customer requirements contained in the set of car features (CR) that has to be deleted from CR (or adapted) in order to make the remaining constraints consistent with CKB. Since we are dealing with the diagnosis of customer requirements, we introduce the definition of a customer requirements diagnosis problem (Definition 4). This definition is based on (Felfernig et al., 2004).

Definition 4 (CR Diagnosis Problem). A customer requirements diagnosis (CR diagnosis) problem is defined as a triple (CKB, CR) where CR is the set of given customer requirements and CKB represents the constraints part of the configuration knowledge base.

The definition of a CR diagnosis that corresponds to a given CR Diagnosis Problem is the following (see Definition 5).

Definition 5 (CR Diagnosis). A CR diagnosis for a CR diagnosis problem (CKB, CR) is a set ∆ ⊆ CR, s.t., CKB ∪ (CR · ∆) is consistent. ∆ is minimal if there does not exist a diagnosis ∆′ ⊂ ∆ s.t. CKB ∪
include the calculation of another conflict set determined by QuickXplain. For the purposes of our working example, let us assume that the first minimal conflict set determined based on an efficient divide-and-conquer search strategy. For the sake of our working example, let us assume that the first minimal conflict set determined by QuickXplain is the set $C_S^1 = \{c_5, c_6\}$. Due to the minimality property, we are able to resolve each conflict by simply deleting one element from the set, for example, in the case of $C_S^1$ we have to either delete $c_5$ or $c_6$. Each variant to resolve a conflict set is represented by a specific path in the corresponding HSDAG.

The original HSDAG algorithm (Reiter, 1987) follows a strict breadth-first search regime. Following this strategy, the next node to be expanded in our working example is the minimal conflict set $C_S^3$ which has been returned by QuickXplain for $C_R - \{c_5\} \cup C_KB$. In this context, the first option to resolve $C_S^3$ is to delete $c_6$. This option is a valid one and $\Delta_1 = \{c_5, c_6\}$ is the resulting minimal diagnosis. The second option for resolving $C_S^3$ is to delete $c_5$. In this case, we have identified the next minimal diagnosis $\Delta_2 = \{c_5, c_7\}$ since $C_R - \{c_5\} \cup C_KB$ is inconsistent. This way we are able to identify all minimal sets of constraints $\Delta_1$ that – if deleted from $C_R$ – help to restore the consistency with $C_KB$. If we want to calculate the complete set of diagnoses for our working example, we still have to resolve the conflict set $C_S^2$. The first option to resolve $C_S^2$ is to delete $c_5$ – since $\{c_5, c_6\}$ has already been identified as a minimal diagnosis, we can close this node in the HSDAG. The second option to resolve $C_S^2$ is to delete $c_7$. In this case we have determined the third minimal diagnosis which is $\Delta_3 = \{c_5, c_7\}$.

In our working example we are able to enumerate all possible diagnoses that help to restore consistency. However, the calculation of all minimal diagnoses is expensive and thus in many cases not practicable for interactive settings. Since users are often interested in a reduced subset of all the potential diagnoses, alternative algorithms are needed that are capable of identifying preferred diagnoses (De Kleer, 1990; Felfernig et al., 2009). Such approaches have already been developed (De Kleer, 1990; Felfernig et al., 2009), however, they are still based on the resolution of conflict sets which is computationally expensive (see Section 5). Our idea presented in this paper is a diagnosis algorithm that helps to determine preferred diagnoses without the need of calculating the corresponding conflict sets. The basic properties of FastDiag will be discussed in Section 4.

4 Calculating Preferred Diagnoses

With FASTDIAG

Users typically prefer to keep the important requirements and to change or delete (if needed) the less important ones (Junker, 2004). The major goal of (model-based) diagnosis tasks is to identify the preferred (leading) diagnoses which are not necessarily minimal cardinality ones (De Kleer, 1990). For the characterization of a preferred diagnosis we will rely on the definition of a total ordering of the given set of constraints in $C$ (respectively $C_KB$). Such a total ordering can be achieved, for example, by directly asking the customer regarding the preferences, by applying multi-attribute utility theory (Ardissono et al., 2003; Winterfeldt and Edwards, 1986) where the determined interest dimensions correspond with the attributes of $C_KB$ or by applying the rankings determined by conjoint analysis (Belanger, 2005). The following definition of a lexical ordering (Definition 6) is based on total orderings for constraints (Junker, 2004) for the determination of preferred conflict sets.

Definition 6 (Total Lexicographical Ordering). Given a total order $\prec$ on $C$, we enumerate the constraints in $C$ in increasing $<_{\text{lex}}$ order $c_1 \ldots c_n$, starting with the least important constraints (i.e., $c_j <_{\text{lex}} c_i \Leftrightarrow i < j$). We compare two subsets $X$ and $Y$ of $C$ lexicographically as follows:

$$X >_{\text{lex}} Y \iff \exists k: c_k \in Y - X \land$$

$$X \cap \{c_{k+1}, \ldots, c_l\} = Y \cap \{c_{k+1}, \ldots, c_l\}.$$
(c₇ < c₆ < c₅), FastDiag would then determine \{c₆, c₇\} as the preferred minimal diagnosis.

4.1 FastDiag Approach

For the following discussions we introduce the set AC = CKB ∪ CR which represents the union of customer requirements (Cₚ) and the configuration knowledge base (CKB). The basic idea of the FastDiag algorithm (Algorithm 1) is the following.¹ In our working example, the set of customer requirements CR = \{c₅, c₆, c₇\} includes at least one minimal diagnosis since CKB is consistent and CKB ∪ CR is inconsistent. If CR itself represents the minimal diagnosis this means that all constraints in Cₚ are part of the diagnosis. In our case Cₚ does not obviously represent a minimal diagnosis — the set of diagnoses in our working example is \{Δ₁ = \{c₅, c₆\}, Δ₂ = \{c₅, c₇\}, Δ₃ = \{c₆, c₇\}\} (see Section 3).

The next step in Algorithm 1 is to divide the set of customer requirements CR = \{c₅, c₆, c₇\} into the two sets C₁ = \{c₅\} and C₂ = \{c₆, c₇\} and to check whether AC - C₁ is already consistent. If this is the case, we can omit the set C₂ since at least one minimal diagnosis can already be identified in C₁. In our case, AC - \{c₅\} is inconsistent, which means that we have to consider further elements from C₂. Therefore, C₂ = \{c₆, c₇\} is divided into the sets \{c₆\} and \{c₇\}. In the next step we can check whether AC - (C₁ ∪ \{c₆\}) is consistent — this is the case which means that we do not have to further take into account \{c₇\} for determining the diagnosis. Since \{c₅\} does not include a diagnosis but \{c₆\} includes a diagnosis, we can deduce that \{c₅\} must be part of the diagnosis. The final step is to check whether AC - \{c₅\} leads to a diagnosis without including \{c₅\}. We see that AC - \{c₅\} is inconsistent, i.e., Δ = \{c₅, c₆\} is a minimal diagnosis for the CR diagnosis problem (CR = \{c₅, c₆, c₇\}, CKB = \{c₁, ..., c₄\}). An execution trace of the FastDiag algorithm in the context of our example is shown in Figure 2.

![Figure 2: FastDiag execution trace for the CR diagnosis problem (Cₚ=\{c₅, c₆, c₇\}, CKB=\{c₁, c₂, c₃, c₄\}).](image)

Algorithm 1 – FastDiag

```plaintext
1 func FastDiag(C ⊆ AC, AC = \{c₁,..,cₙ\}) : \Delta
2 if isEmpty(C) or inconsistent(AC - C) return ∅
3 else return FD(∅, C, AC);
4 func FD(D, C = \{c₁,..,cₖ\}, AC) : diagnosis \Delta
5 if D ≠ ∅ and consistent(AC) return ∅;
6 if singleton(C) return C;
7 k = \#C
8 C₁ = \{c₁,...,cₖ\}; C₂ = \{cₖ₊₁,..,cₙ\};
9 D₁ = FD(C₁, C₂, AC - C₁);
10 D₂ = FD(D₁, C₁, AC - D₁);
11 return (D₁ ∪ D₂);
```

¹In Algorithm 1 we use the set C instead of CR since the application of the algorithm is not restricted to inconsistent sets of customer requirements.

4.2 Calculating n>1 Diagnoses

In order to be able to calculate n>1 diagnoses² with FastDiag we have to adopt the HSDAG construction introduced in (Reiter, 1987) by substituting the resolution of conflicts (see Figure 1) with the deletion of elements cₙ from CKB (C) (see Figure 3). In this case, a path in the HSDAG is closed if no further diagnoses can be identified for this path or the elements of the current path are a superset of an already closed path (containment check). Conform to the HSDAG presented in (Reiter, 1987), we expand the search tree in a breadth-first manner. In our working example, we can delete \{c₅\} (one element of the first diagnosis Δ₁ = \{c₅, c₆\}) from the set CR of diagnosable elements and restart the algorithm for finding another minimal diagnosis for the CR diagnosis problem (\{c₅, c₇\}, CKB). Since AC - \{c₅\} is inconsistent, we can conclude that CKB = \{c₆, c₇\} includes another minimal diagnosis (Δ₂ = \{c₆, c₇\}) which is determined by FastDiag for the CR diagnosis problem (CR = \{c₅, c₆, c₇\}, CKB). Finally, we have to check whether the CR diagnosis problem (\{c₅, c₇\}, CKB) leads to another minimal diagnosis. This is the case, i.e., we have identified the last minimal diagnosis which is Δ₃ = \{c₆, c₇\}. The calculation of all diagnoses in our working example on the basis of FastDiag is depicted in Figure 3.

Note that for a given set of constraints (C) FastDiag always calculates the preferred diagnosis in terms of Definition 7. If Δ₁ is the diagnosis returned by FastDiag and we delete one element from Δ₁ (e.g., c₅), then FastDiag returns the preferred diagnosis for the CR diagnosis problem (\{c₅, c₆, c₇\}, \{c₃, ..., c₇\}) which is Δ₂ in our example case, i.e., Δ₁ > Δ₂. Consequently, diagnoses part of one path in the search tree (such as Δ₁ and Δ₂ in Figure 3) are in a strict preference ordering. However, there is only a partial order between individual diagnoses in the search tree in the sense that a diagnosis at level k is not necessarily preferable to a diagnosis at level k+1.

4.3 FastDiag Properties

A detailed listing of the basic operations of FastDiag is shown in Algorithm 1. First, the algorithm checks whether the constraints in C contain a diagnosis, i.e., whether AC - C is consistent — the function assumes

²Typically a CR diagnosis problem has more than one related diagnosis.
The only situation in which elements of \( C \) are added

\[ 5.1 \quad \text{Performance of FastDiag} \]

\[ 5 \]

Evaluation

QuickXplain conflict detection algorithm introduced by FastDiag, this would mean that there exists at least

\[ \Delta \]

the existence of a non-minimal diagnosis in the sense of Definition 5. If we assume

part of the diagnosis. FastDiag computes only

it is not possible to delete elements from \( C \) which are

elements of \( C \)

\[ \emptyset \]

be deleted from \( C \) (i.e., not included in a diagnosis) is

constraint \( c \) FastDiag, this would mean that there exists at least one

diagnosis can be identified. If we assume the existence

AC-C is consistent and \( C \) consists of at least one con-

one of them (the preferred one – see Definition 7)

FD will find it. If there are multiple minimal diagnoses

that if \( C \) contains exactly one minimal diagnosis then

a minimal diagnosis. FastDiag is

AC is still inconsistent, then the element in \( C \) is part of

in \( C \) which is part of the diagnosis but not

returned by FD. The only way in which elements can be deleted from \( C \) (i.e., not included in a diagnosis) is

by the return \( \emptyset \) statement in FD and \( \emptyset \) is only returned in the case that AC is consistent which means that the elements of \( C \) (\( C_1 \)) from the previous FD incarnation are not part of the preferred diagnosis. Consequently, it is not possible to delete elements from \( C \) which are part of the diagnosis. FastDiag computes only minimal
diagnoses in the sense of Definition 5. If we assume

the existence of a non-minimal diagnosis \( \Delta \) calculated by FastDiag, this would mean that there exists at least one

constraint \( c_i \) with \( \Delta - \{ c_i \} \) is still a diagnosis. The only situation in which elements of \( C \) are added to a diagnosis \( \Delta \) is if \( C \) itself contains exactly one element. If \( C \) contains only one element (let us assume \( c_i \)) and AC is inconsistent (in the function FD) then \( c_i \)

is the only element that can be deleted from AC, i.e., \( c_i \) must be part of the diagnosis.

5 Evaluation

5.1 Performance of FastDiag

In this subsection we compare the performance of

FastDiag with the performance of the hitting set algorithm (Reiter, 1987) in combination with the

QuickXplain conflict detection algorithm introduced in (Junker, 2004).

The worst case complexity of FastDiag in terms of

the number of consistency checks needed for calculating

one minimal diagnosis is \( 2d\log_2(\frac{n}{d})+2d \), where \( d \)

is the minimal diagnoses set size and \( n \) is the number

of constraints (in \( C \)). The best case complexity is \( \log_2(\frac{n}{d})+2d \). In the worst case each element of the di-

agnosis is contained in a different path of the search

tree: \( \log_2(\frac{n}{d}) \) is the depth of the path, \( 2d \) represents

the branching factor and the number of leaf-node consistency checks. In the best case all elements of the diagnosis are contained in one path of the search tree.

The worst case complexity of QuickXplain in terms of

consistency checks needed for calculating one minimal
diagnosis then \( n \) is the minimal conflicts set size and \( n \) is again the number of

constraints (in \( C \)) (Junker, 2004). The best case com-

plexity of QuickXplain in terms of the number of con-

sistency checks needed is \( \log_2(\frac{n}{d})+2k \) (Junker, 2004).

Consequently, the number of consistency checks per

conflict set (QuickXplain) and the number of consistency checks per diagnosis (FastDiag) fall into a loga-

rithmic complexity class.

Let \( n_{c} \) be the number of minimal conflict sets in a

constraint set and \( n_{diag} \) be the number of minimal diag-

noses, then we need \( n_{diag} \) FD calls (see Algorithm

1) plus \( n_{c} \) additional consistency checks and \( n_{c} \) ac-

tivations of QuickXplain with \( n_{diag} \) additional consistency

checks for determining all diagnoses. The re-

sults of a performance evaluation of FastDiag are de-

picted in the Figures 4–7. The basis for these evalu-

ations were generated constraint sets (\( t = 100 \) con-

straints with a randomized lexicographical ordering and \( n = 100 \) variables) with a varying number of con-

flict sets (of cardinality 1–4) and corresponding diag-

noses (#diagnoses between 3 – 22). The constraint

solver used for consistency checking was CHOCO

(choco.emn.fr) and the tests have been executed on a

standard desktop computer (Intel(R) Core(TM)2 Quad

CPU QD9400 2.66Ghz and 2GB RAM).

Each experiment (data point in the graph) has been conducted 10 times with a differing constraint order-

ing. Figure 4 shows a comparison between the hitting

set based diagnosis approach (denoted as HSDAG) and

the FastDiag algorithm (denoted as FastDiag) in the

case that only one diagnosis is calculated. Fast-

Diag clearly outperforms the HSDAG approach inde-

pendent of the way in which diagnoses are calculated

(breadth-first or best-first). Figure 5 shows the perfor-

mance evaluation for calculating the topmost-5 mini-

mal diagnoses. The result is similar to the one for

calculating the first diagnosis, i.e., FastDiag outper-

forms the two HSDAG versions. Our evaluations show

that FastDiag is very efficient in calculating preferred

minimal diagnoses. In contrast to the HSDAG-based

best-first search mode FastDiag has a performance that

makes it an excellent choice for interactive settings.

5.2 Prediction Quality of FastDiag

On the basis of interaction data collected with a com-

puter configuration environment we also analyzed the

prediction quality of FastDiag. We measured the qual-

ity of the algorithm in terms of precision (see Formula

1): how often a diagnosis that leads to a selected con-

figuration (selected by the user) is among the topmost-

Figure 3: FastDiag: calculating the complete set of minimal diagnoses.
Figure 4: Calculating the first min. diagnosis with FastDiag vs. hitting set based diagnosis with QuickXplain.

Figure 5: Calculating topmost-5 min. diagnoses with FastDiag vs. hitting set based diagnosis with QuickXplain.

$$\text{precision} = \frac{|\text{correctly predicted diagnoses}|}{|\text{predicted diagnoses}|}$$

The major result of this study was that we could not detect any significant difference in prediction quality between FastDiag and best-first search based HSDAG. Due to space limitations a detailed characterization of the study results will be provided in an extended version of this paper.

5.3 A Simple Extension: FlexDiag

We now introduce a simple modification of FastDiag (FlexDiag) which allows us to further reduce the number of consistency checks in the case we want to calculate exactly one (preferred) diagnosis. This can be achieved with the tradeoff of loosing the minimality property (see Definition 5). Note that further evaluations and extensions of FlexDiag are within the scope of future work.

A reduction of the number of consistency checks can be achieved by changing the $\text{singleton}(C)$ statement in line 6 of Algorithm 1 to $\text{size}(C) \leq m$. If we check the current set of constraints $C$ for the containment of $\leq m$ elements (for $m>1$), the modified algorithm (FlexDiag) determines a constraint set which includes the minimal diagnosis (but itself is not necessarily minimal). Consequently, there is a tradeoff: if we increase $m$, we decrease the number of consistency checks and vice-versa.

FlexDiag calculates a set of constraints $C_D$ where $C_D = \Delta \cup C_{\text{offset}}$. $\Delta$ is the minimal (and preferred) diagnosis calculated by Algorithm 1 and $C_{\text{offset}}$ is a set of constraints which are not part of $\Delta$. We evaluated this approach w.r.t. the relevance (see Formula 2) of $C_D$. We define relevance as the share of relevant constraints (all constraints in $\Delta$) w.r.t. the constraints in $C_D$ (see Formula 2):

$$\text{relevance}(C_D) = \frac{|\Delta|}{|C_D|}$$

We calculated the average relevance of diagnoses determined by FlexDiag for our dataset (see Section 5.1). The relevance for $m=2$ is around 60% which means that 40% of the constraints are not part of the minimal (and preferred) diagnosis, on an average (calculated from 100 runs with randomized constraint order). Further increasing the value of $m$ (for $m=3, 4, 5$) significantly decreases the relevance of the corresponding diagnoses $C_D$ (see Figure 8). Lower relevance values correlate with decreasing runtimes (see Table 1).

<table>
<thead>
<tr>
<th>$m$</th>
<th>16.6%</th>
<th>33.8%</th>
<th>35.5%</th>
<th>41.1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Runtime improvements achieved with FlexDiag, for example, with $m=2$ we can reduce the average runtime for calculating one diagnosis by 16.6%.

6 RELATED WORK

The authors of (Felfernig et al., 2004) introduce an algorithm for the automated debugging of configuration knowledge bases. The idea is to combine a conflict detection algorithm such as QuickXplain (Junker, 2004) with the hitting set algorithm used in model-based diagnosis (MBD) (Reiter, 1987; DeKleer et al., 1992) for the calculation of minimal diagnoses. In this context, conflicts are induced by test cases (examples) that, for example, are stemming from previous
configuration sessions, have been automatically generated, or have been explicitly defined by domain experts. Further applications of MBD in constraint set debugging are introduced in (Felfernig et al., 2007) where diagnosis concepts are used to identify minimal sets of faulty transition conditions in state charts and in (Felfernig et al., 2008) where MBD is applied for the identification of faulty utility constraint sets in the context of knowledge-based recommendation. In contrast to (Felfernig et al., 2007; 2004; 2008), our work provides an algorithm that allows to directly determine diagnoses without the need to determine corresponding conflict sets. FastDiag can be applied in knowledge engineering scenarios for calculating preferred diagnoses for faulty knowledge bases given that we are able to determine reasonable ordering for the given set of constraints – this could be achieved, for example, by the application of corresponding complexity metrics (Chen and Suen, 2003).

In contrast to the algorithm presented in this paper, calculating diagnoses for inconsistent requirements typically relies on the existence of (minimal) conflict sets. A well-known algorithm with a logarithmic number of consistency checks depending on the number of constraints in the knowledge base and the cardinality of the minimal conflicts QuickXplain (Junker, 2004) has made a major contribution to more efficient interactive constraint-based applications. QuickXplain is based on a divide-and-conquer strategy. FastDiag relies on the same principle of divide-and-conquer but with a different focus, namely the determination of minimal diagnoses. QuickXplain calculates minimal conflict sets based on the assumption of a linear preference ordering among the constraints. Similarly – if we assume a linear preference ordering of the constraints in C – FastDiag calculates preferred diagnoses.

The authors of (O’Sullivan et al., 2007) focus on interactive settings where users of constraint-based applications are confronted with situations where no solution can be found. In this context, (O’Sullivan et al., 2007) introduce the concept of minimal exclusion sets which correspond to the concept of minimal diagnoses as defined in (Reiter, 1987; DeKleer et al., 1992). As mentioned, the major focus of (O’Sullivan et al., 2007) are interactive settings where the proposed algorithm supports users in the identification of acceptable exclusion sets. The authors propose an algorithm (representative explanations) that helps to improve the quality of the presented exclusion set and thus increases the probability of finding an acceptable exclusion set for the user. Our diagnosis approach calculates preferred diagnoses in terms of a predefined ordering of the constraint set. Thus – compared to the work of (O’Sullivan et al., 2007) – we follow a different approach in terms of focusing more on preferences than on the degree of...
representativeness.

Many of the existing diagnosis approaches do not take into account the need for personalizing the set of diagnoses to be presented to a user. Identifying diagnoses of interest in an efficient manner is a clear surplus regarding the acceptance of the underlying application. A first step towards the application of personalization concepts in the context of knowledge-based recommendation is presented in (Felfernig et al., 2009). The authors introduce an approach that calculates leading diagnoses on the basis of similarity measures used for determining n-nearest neighbors. A general approach to the identification of preferred diagnoses is introduced in (DeKleer, 1990) where probability estimates are used to determine the leading diagnoses with the overall goal to minimize the number of measurements needed for identifying a malfunctioning device. We see our work as a major contribution in this context since FastDiag helps to identify leading diagnoses more efficiently – further empirical studies in different application contexts are within the major focus of our future work.

7 CONCLUSION

In this paper we have introduced a new diagnosis algorithm (FastDiag) which allows the efficient calculation of one diagnosis at a time with logarithmic complexity in terms of the number of consistency checks. Thus, the computational complexity for the calculation of one minimal diagnosis is equal to the calculation of one minimal conflict set in hitting set based diagnosis approaches. The algorithm is especially applicable in settings where the number of conflict sets is equal to or larger than the number of diagnoses, or in settings where preferred (leading) diagnoses are needed.

REFERENCES


