2009 International Conference on Adaptive and Intelligent Systems

Solving Over-Constrained Problems using Network Analysis

Monika Schubert
Applied Software Engineering, IST
Graz University of Technology
Graz, Austria
monika.schubert@ist.tugraz.at

Alexander Felfernig
Applied Software Engineering, IST
Graz University of Technology
Graz, Austria
alexander.felfernig@ist.tugraz.at

Monika Mandl
Applied Software Engineering, IST
Graz University of Technology
Graz, Austria
monika.mandl@ist.tugraz.at

***Abstract*—Requirements for which no recommendation can be calculated are unsatisfactory for the user. The detection and resolution of conflicts between those requirements and the product assortment is an important functionality to successfully guide the user to a solution. In this paper we introduce a new approach how to identify minimal conflict sets in over constrained problems through network analysis. Conflict sets offer the information which constraints (requirements) need to be changed to retrieve a solution. Random constrained problems are used to evaluate our approach and compare it to existing conflict detection algorithms. A major result of this evaluation is that our approach is superior in settings typical for knowledge-based recommendation problems.**

***Keywords*-Recommender Systems; Explanations; CSP**

I. Introduction

Knowledge-based recommender systems are a special class of recommender systems, that support the identification of interesting items based on user requirements and knowledge about the items [1], [2]. In order to receive a recommendation from a knowledge-based recommender system, the user has to enter a set of requirements (constraints). The system then determines those items that satisfy the given set of user requirements. If these constraints can not be fulfilled, the system needs to calculate explanations [2], [4], [9] that indicate minimal sets of changes such that a recommendation can be found. Existing approaches to handle these changes focus on low-cardinality diagnoses [5], [6] or on high-cardinality fulfilled subqueries [7].

In this paper we present a new method to identify a minimal set of constraints that need to be changed in order to correct a set of unsatisfyable requirements. This approach focuses on an efficient calculation of n minimal conflict sets - where n is an integer number between 1 and the maximum amount of minimal conflict sets - using methods from network analysis [10]. These sets can be further used to identify the best suited repair action for the user [4].

This paper is organised as follows. First, we introduce a motivating example, which will be used throughout the paper to explain the algorithm described in Section III. After the description of the algorithm based on concepts of the network analysis, we evaluate our approach including a comparison to QuickXplain [6] the standard algorithm for the calculation of minimal conflict sets. In section V we discuss related work and conclude the paper.

Table I
WORKING EXAMPLE

	P1	P2	P3	P4	P5	P6	P7	P8
Price	600	700	1200	1000	1100	1150	800	1200
Weight	2.2	2.6	3.2	2.3	2.5	3.0	2.6	3.3
Size	8.9	15.4	17	10	15.4	12	15.4	13
RAM	1	2	3	4	2	4	3	4
Hard Drive	80	120	150	200	180	300	120	400
OS included	yes	yes	yes	no	yes	no	no	yes

II. Working Example

For demonstration purposes we introduce an example that will be used for further explanations. This example comes from the field of recommender systems and deals with retrieving laptops for specific requirements. Table I shows the products of our working example (P1, ..., P8) including a specification of their properties (e.g., price). We want to retrieve one laptop with the following requirements: r_1*: price* < 750, r_2*: weight* ≤ 2.5, r_3*: size* ≥ 15.4, r_4*: RAM* > 2 and r_5*: OS included = yes*. The products are normally stored in one or more database tables (Table I is an example for a simple database table). In order to receive a recommendation we create a conjuntive query that is sent as a request to the database. The conjunctive query for the user requirements r_1 to r_5 would be: *SELECT * FROM WorkingExapmple WHERE price* < 750 *AND weight* ≤ 2.5 *AND size* ≥ 15.4 *AND RAM* > 2 *AND OS included = yes*. When requesting this query we can easily see that the returning result set is empty - there is no product that satisfies all these requirements (constraints).

III. Approach

Our approach is based on the analysis of a network consisting of constraints (requirements) and items (products). The basis for our algorithm is a specific type of constraint satisfaction problem (CSP). Such a CSP consists of a set of constraints $C = \{r_1, r_2, ..., r_m\}$ and a set of items $I = \{p_1, p_2, ..., p_n\}$. Each item $p_i \in I$ has a set of possible values D_i from its domain. Such a CSP is satisfiable (it has a solution) if there exists at least one item from I that fulfills

978-0-7695-3827-3/09 $25.00 © 2009 IEEE
DOI 10.1109/ICAIS.2009.12

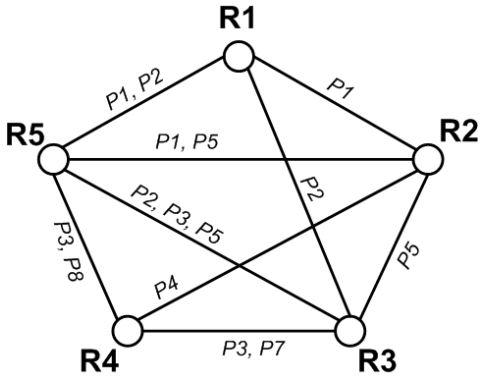

Figure 1. Graph connecting the requirements (constraints)

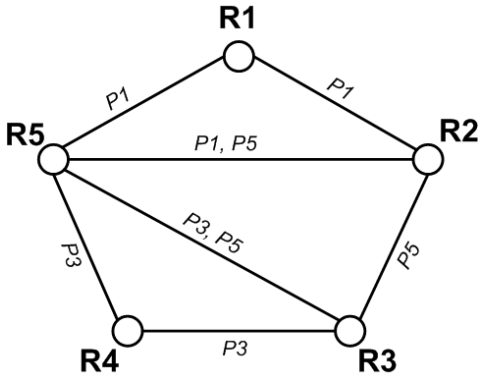

Figure 2. Pruned graph connecting the requirements (constraints) after removing all minimal conflict sets with cardinality ≤ 3

all constraints in C. If a CSP is over-constrained it has no solution i.e., there does not exist an item that fulfills all the constraints in C.

A. Constraints as a Network of Items

The main core of a CSP are the constraints. Studying the structure of these constraints can bring more insight to the solution of the CSP. Every CSP in the above sense can be represented as an adjacency matrix represented by the dimensions constraints (requirements) and items (products) (see, e.g., Table II).

Table II
ADJACENCY MATRIX OF CONSTRAINTS AND ITEMS

	P1	P2	P3	P4	P5	P6	P7	P8
Price	1	1	0	0	0	0	0	0
Weight	1	0	0	1	1	0	0	0
Size	0	1	1	0	1	0	1	0
RAM	0	0	1	1	0	1	1	1
OS included	1	1	1	0	1	0	0	1

Every time an item fulfils a constraint the value in the matrix is set to 1 (if not the value is 0). This adjacency matrix can also be seen as a two-mode network (also referred as affiliation network) of items I which are associated with the constraints C. One way to analyse such a two-mode network is to project (or fold) [10] it into a one-mode network of one of the two possible amplitudes. For analysing the structure of the constraints we project the adjacency matrix M to a constraint network $C^* = M^T M$. This network C* contains information about the structure of the constraints and allows to analyse their relationships. Figure 1 shows our constraint network of the working example. The constraints are stored in nodes. The information which product satisfies the constraints is stored in the edges. Whenever a product satisfies two constraints an edge is connecting these two constraints. How this can be used for the identification of conflicts is explained in the next section.

B. Identification of Conflicts

A graph consisting of nodes that represent constraints can be used to identify minimal conflict sets. If there is a constraint not satisfied by any item, then this is a minimal conflict set as well (*conflict set with cardinality 1*). Such a constraint can be identified by calculating the sum of each row of the adjacency matrix (meaning each constraint). If the sum of a row is zero than this constraint is not satisfied by any item and thus a minimal conflict set. Note that no such constraint exists in our working example.

In order to identify the conflicts in constraint networks with *cardinality 2*, we are looking for missing edges in the constraint graph. If there is no edge between two constraints, it means that there is no product that can satisfy these two constraints. Therefore at least one of these constraints needs a relaxation in order to find a solution.

In order to be able to apply this step of the algorithm to our working example, we have to check whether there is any edge missing (compared to a fully connected graph) in the graph. We see, for example, that an edge connecting R1-R4 is missing. Consequently we have found a minimal conflict set $\{r_1, r_4\}$.

After the identification and extraction of all minimal conflict sets caused by the missing edges (*conflict sets of cardinality 2*), we continue with the next step. We analyse all fully connected subgraphs. If in a subgraph not all edges have a common label for at least one item (product) p_i, this subgraph represents a conflict set. This property is based on the simple fact that there is no item that satisfies all connected constraints.

Let us assume the first fully connected subgraph retrieved from our working example is *R1-R2-R3*. The intersection of the labels of the edges connecting this subgraph is empty and thus we found another minimal conflict set. Note that *R1-R2-R4* is not a fully connected subgraph and therefore we don't have to deal with it ($\{r_1, r_4\}$ is already a minimal conflict set). Applying this method to all subgraphes with 3 nodes

we identify the minimal conflict sets $CS_i \in MCS$: $CS_1 = \{r_1, r_2, r_3\}$, $CS_2 = \{r_2, r_3, r_4\}$ and $CS_3 = \{r_2, r_4, r_5\}$. Note that the edges of triangle *R1-R2-R5* have the item *P1* in common, the edges *R2-R3-R5* have the item *P5* in common and finally, the edges of *R3-R4-R5* have item *P3* in common.

All minimal conflict sets can be removed, which leaves us with a pruned graph shown in Figure 2. In this graph we only take into account the labels of the common items because the other labels do not contain any useful information. The pruned graph of our working example does not have any further fully connected subgraphs with a cardinality > 3. Consequently we have completed the calculation of all minimal conflict sets that exist in our working example.

The algorithm for calculating the minimal conflict sets using a graph based approach is the following (Algorithm 1 - NA). The input value is an adjacency matrix M which holds the satisfaction of each item for every constraint (see, e.g., Table II). From this matrix we retrieve all constraints that are not satisfied by any item (*retrieveSingleNodes*). All these constraints are minimal conflict sets and thus stored in the list of minimal conflict sets *MCS*. In the next step we project the two-mode adjacency matrix into a constraint-mode graph *G* (see Section III-A). All possible edges which are not in the graph are calculated by *retrieveMissingEdges* and are stored in the list of minimal conflict sets *MCS*. Thus all minimal conflict sets of cardinality 2 have been identified.

For all subgraphs with increasing cardinality (*starting with cardinality 3*) we take the first one (with *subgraphs.next*) and check whether this subgraph *SG* has a common label on all edges. If it does not then we found a minimal conflict set which is added to *MCS*. If this subgraph *SG* is a minimal conflict set then it can be excluded from further investigations. At this point there are different implementation possibilities: either all subgraphs are hold in a list and the super sets of the minimal conflict graphs are removed or all subgraphs for one cardinality are calculated and after checking each individual, the graph is rebuild excluding the current minimal conflict sets (see, e.g., Figure 2). In our implementation we chose alternative one with removing all supersets.

Algorithm 1 NA (M)

{Input: M - adjacency matrix of constraints and items}
$MCS \leftarrow retrieveSingleNodes(M)$
$G \leftarrow M^T M$
$MCS \leftarrow MCS \cup retrieveMissingEdges(G)$
for all $subgraphs$ **do**
  $SG \leftarrow subgraphs.next$
  if $notHasCommonEdge(SG)$ **then**
    $MCS \leftarrow MCS \cup \{edges(SG)\}$
    $subgraphs.adjust(SG)$
    {to ensure to only retrieve minimal conflict sets}
  end if
end for
return MCS
{Output: return MCS including all minimal conflict sets}

IV. Evaluation

We are now going to discuss in the runtime of our algorithm for different applications with different characteristics. In this context we compare our network analysis approach with the QuickXplain [6] from Junker. We implemented both algorithms in Java 1.6, all experiments were performed on a normal desktop PC where all products (items) are stored in a SQL database. The consistency check is done through sending conjunctive queries to the database. The QuickXplain [6] algorithm is based on a recursive divide-and-conquer strategy, which calculates one conflict set at a time. To make this approach comparable to our NA algorithm (Algorithm 1), which calculates all minimal conflict sets for a CSP, we used the HSDAG (Hitting Set Directed Acylic Graph) by Reiter [8] to build up a tree with all minimal conflict sets.

A. Influence of the Number of Items

First we studied the influence of the number of items on the runtime. Therefore we generated applications with 10 constraints and an increasing number of items. The number of items is distributed on a logarithmic scale in order to study the influence on a wide spectrum. Both algorithms, our network analysis based approach and QuickXplain [6] started with 100 items and went up to 1600.

In the first test we set the satisfaction of the items to 50% which means that the probability that a constraint is fulfilled by an item is 50%. This provides us with a high variant of different cardinalities of minimal conflict sets. This test was performed 30 times and the mean values of the runtime are plotted in Figure 3. The plot shows that up to 200 items the NA algorithm (Algorithm 1) performs better than the QuickXplain. The continuous increase of the runtime of the NA algorithm can be explained by the folding procedure (see Section III-A).

In comparison to this observation we performed another test with a satisfaction rate of 40%. From the plot of both algorithms (Figure 4) we can see that up to more than 800 items the network analysis approach outperforms the QuickXplain [6]. An interesting observation is the fact, that the QuickXplain [6] performs even better with a higher number of items. The runtime of the QuickXplain algorithm is linear dependent on the number of minimal conflict sets due to a higher amount of calls of the theorem prover for more minimal conflict sets. This amount of minimal conflict sets decreases with a higher number of items. This holds especially for a higher satisfaction rate as this indicates the probability that an item satisfies one constraint which could eliminate the constraint from the minimal conflict set.

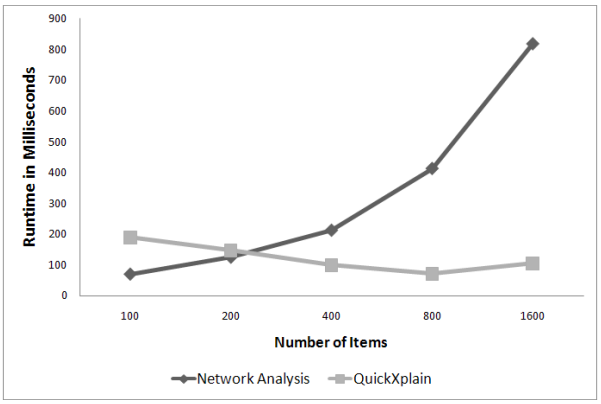

Figure 3. Runtime comparison of the algorithms QuickXplain and Network-Analysis for increasing amount of items and a fixed amount of 10 constraints and a satisfaction rate of 50%

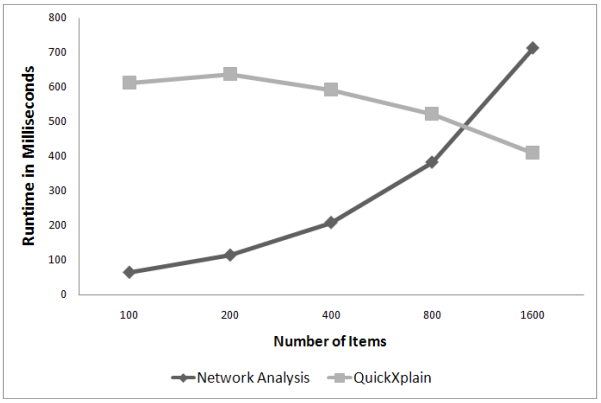

Figure 4. Runtime comparison of the algorithms QuickXplain and Network-Analysis for increasing amount of items and a fixed amount of 10 constraints and a satisfaction rate of 40%

B. Influence of the Number of Constraints

To study not only the influence of the number of items to our algorithm (Algorithm 1) we evaluated our approach with different number of constraints. We started with 6 constraints and went up to 14. For all these settings we used 500 items to be comparable. Figure 5 shows that for up to 8 constraints the runtime is so low that there is nearly no difference between the algorithms. This fact was already discussed in the Section IV-A. With more than 10 constraints the NA algorithm (Algorithm 1) outperforms the QuickXplain [6]. For 14 constraints the QuickXplain [6] needs around 75 seconds, whereas the network based approach needs less than 5 seconds.

Summarizing, for typical knowledge-based recommendation settings our proposed network analysis algorithm (Algorithm 1) is the better choice since the cardinality of a set of customer requirements could include 10-20 items and a typical product assortment (e.g., in the financial

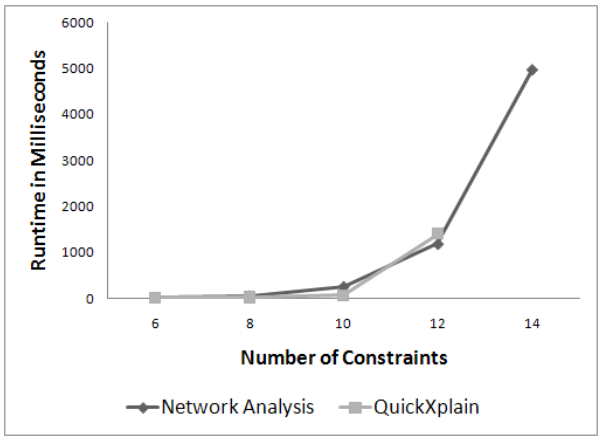

Figure 5. Runtime comparison of the algorithms QuickXplain and Network-Analysis for an increasing amount of constraints and a fixed amount of 500 items (satisfaction rate: 50%)

services domain) includes 50-250 items.[1] Especially for a high number of requirements - up to 20 - that occure in more complex settings the performance of the QuickXplain [6] algorithm is weak compared to our approach (Algorithm 1).

C. Dependency on Satisfaction Rate

As a further exploration of the feasibility of calculating minimal conflict sets with the network based approach (Algorithm 1) we extracted the runtime for different satisfaction rates. The satisfaction rate represents the probability of a constraint of being satisfied by an item. This satisfaction rate has a high impact on the cardinality and the number of minimal conflict sets. If an algorithm performs well on a wide spectrum of satisfaction rates, than it can be used without knowing much about the actual problem.

To have representative and comparable settings we used a set of 500 items and 10 constraints with a satisfaction rate r between 10% and 70%. We stopped at $r = 70\%$ because it is hard to find a setting with a satisfaction rate greater than 70% and still having an over-constraint problem. The satisfaction of each item-constraint relation is set randomly.

Each algorithm calculated the minimal conflict sets of 30 different applications. Figure 6 shows the mean value of the runtimes (30 applications) for $r = 10\%, 20\%, ..., 70\%$. From this figure we can identify, that the NA algorithm (Algorithm 1) has a low average runtime over the different satisfaction rates. The increase of the runtime depending on a larger r is based on the larger search space (less minimal conflict sets can be excluded in an early phase of the algorithm). In contrast to this the QuickXplain [6] has a weak performance on a low satisfaction rate, meaning a

[1]We assumed a satisfaction rate per query of about 40-50% which occurs in the financial services domain [3] – clearly further (product domain-specific) evaluations have to be conducted in this context.

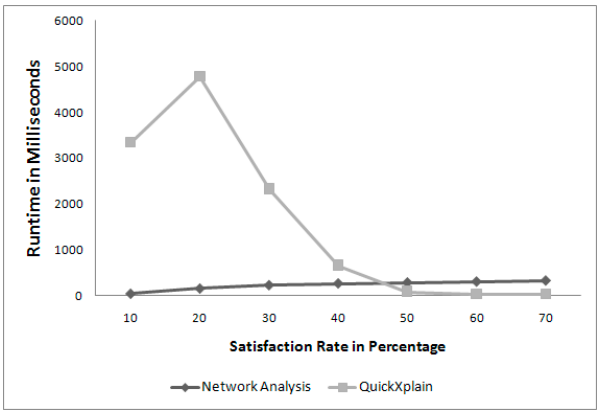

Figure 6. Runtime comparison of the algorithms QuickXplain an Network-Analysis for different satisfaction rates, 10 constraints and 50 items

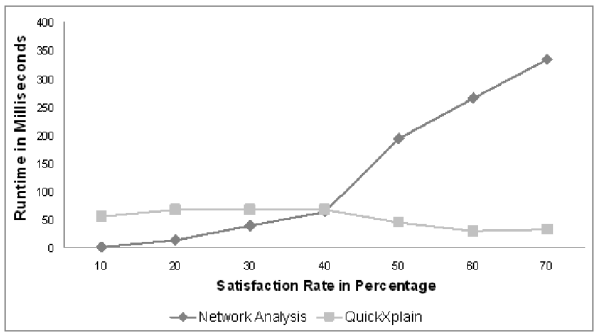

Figure 7. Runtime comparison of the algorithms QuickXplain and Network-Analysis for different satisfaction rates calculating one minimal conflict set

high number of minimal conflict sets with a low cardinality. But on the other hand the QuickXplain [6] performs well for a high satisfaction rate.

D. Calculation of one and more MCS

In applications like [4] not all minimal conflict sets are needed. Thus we compared the runtime for the calculation of one minimal conflict set of our Network Analysis algorithm (Algorithm 1) and QuickXplain [6]. We compared these results for a set of different satisfaction rates representing characteristics of different applications. We performed a test for each satisfaction rate 30 times and calculated the corresponding average value. A lower satisfaction rate results in a smaller cardinality of the minimal conflict set. Up to a satisfaction rate of 40% the Network Analysis algorithm performs better than the QuickXplain. For a more dense set of data the Network Analysis algorithm needs a lot of calculations to check for minimal conflict sets with a low cardinality. The results of the performance test for calculating one minimal conflict set is plotted in Figure 7.

In some applications it is neither necessary to calculate one, nor all minimal conflict sets. For an increasing number

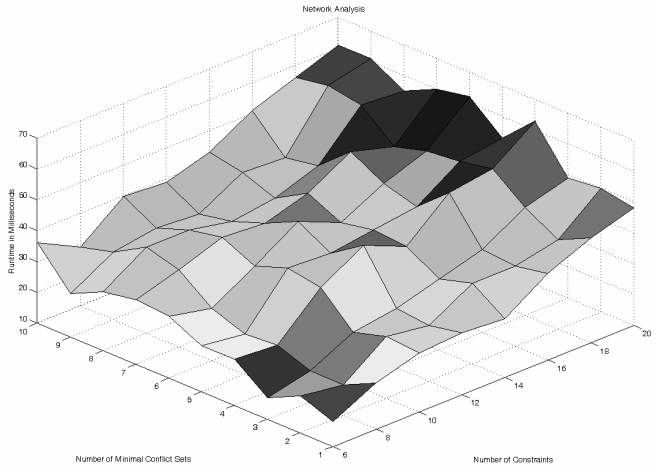

Figure 8. Plot of the runtime of the Network-Analysis for a different amount of minimal conflict set and an increasing number of constraints

of constraints we compared the runtime of the calculation from one up to ten minimal conflicts sets. These tests were performed with 30 runs with 1000 items and an increasing number of constraints (from 6 up to 20). The results of the Network Analysis algorithm (Algorithm 1) can be seen in Figure 8 and the results of the QuickXplain [6] are plotted in Figure 9. Calculating one minimal conflict set using the QuickXplain [6] algorithm is quite fast, but calculating two minimal conflicts increases the runtime with a factor 4.75 when using 20 constraints. In general we identfy from Figure 9 that the runtime of the QuickXplain algorithm is more dependent on the amount of constraints than on the number of minimal conflict sets that are calculated (if more than one minimal conflict set is calculated). In contrast to this the Network Analysis algorithm (Algorithm 1) is dependent on the size of the minimal conflict set. That is the reason why Figure 8 looks a bit clumsy. Anyway we can identify an increasing runtime dependent on the number of constraints as well as on the number of minimal conflicts sets that are calculated. Comparing the two algorithms between each other, the Network Analysis algorithm (Algorithm 1) performs faster on an average. Especially for settings with 12 constraints and above and calculating more than one minimal conflict set the Network Analysis algorithm (Algorithm 1) outperforms the QuickXplain [6] algorithm.

V. Related Work

The work of [2] includes concepts for the automated detection of minimal sets of inconsistent requirements on the basis of Model-Based Diagnosis (MBD) [8]. Those diagnoses are calculated by deriving hitting sets on the basis of conflict resolution. The conflicts exploited in [2] are not necessarily minimal which enlarges the underlying

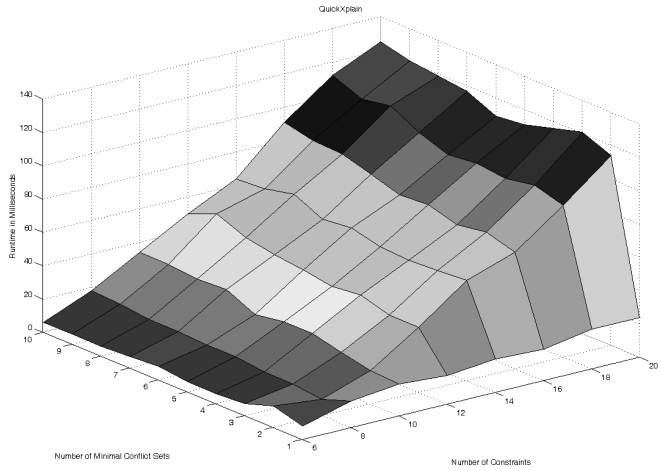

Figure 9. Plot of the runtime of the QuickXplain for a different amount of minimal conflict set and an increasing number of constraints

hitting set acyclic directed graph (HSDAG). In [9] such minimal sets (diagnoses) are denoted as *exclusion sets*, in the work of McSherry (see, e.g., [7]) the complement of such a minimal set is denoted as *maximally successful sub-query*. An efficient algorithm for the determination of minimal conflict sets has been introduced by [6] (QuickXPlain) – in this paper we have compared the performance of QuickXPlain with our network-based algorithm. The major contribution of our paper is a novel approach that is faster in typical recommender system settings for calculating minimal conflict sets, which can be used as a basis for calculating diagnoses and corresponding repair actions.

VI. Conclusion

In this paper we introduced a method how to identify all minimal conflict sets for an over-constrained satisfaction problem. The identification of these minimal conflict sets is extremly important in a knowledge-based recommender system to provide a customer with repair actions for unrealisable requirements (constraints). Thus we came up with an algorithm inspired by common techniques of the network analysis, which performs well and even faster than the state-of-the-art algorithms. The results of the evaluation clearly demonstrate the improvements induced by our algorithm.

Acknowledgment

The work presented in this paper has been developed within the scope of the research projects WECARE (funded by the Austrian Research Agency FFG) and Softnet Austria that is funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsfoerderungsgesellschaft mbH (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

References

[1] R. Burke, *Knowledge-based recommender systems*, in 'Library and Information Systems', vol. 69 (32) New York: Marcel Dekker, 2000, pp. 180-200.

[2] A. Felfernig, G. Friedrich, D. Jannach and M. Stumptner, *Consistency-based Diagnosis of Configuration Knowledge Bases*, in Artificial Intelligence, vol. 152 (2), Essex, UK, 2004, pp. 213-234.

[3] A. Felfernig, K. Isak, K. Szabo, P. Zachar, *The VITA Financial Services Sales Support Environment*, AAAI/IAAI, Vancouver, Canada, 2007, pp. 1692-1699.

[4] A. Felfernig, M. Schubert, G. Friedrich, M. Mandl, M. Mairitsch and E. Teppan, *Plausible Repairs for Inconsistent Requirements*, in 'Proceedings of the 21st International Joint Conference on Artificial Intelligence', Pasadena, California, 2009, pp. 791-796

[5] D. Jannach, *Finding Preferred Query Relaxations in Content-based Recommenders*, in 'Intelligent Techniques and Tools for Novel System Architectures', vol. 109, Springer Berlin / Heidelberg, 2008, pp. 81-97.

[6] U. Junker, *QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems*, in 'Proceedings of the 19th National Conference on Artificial Intelligence', The AAAI Press, California, 2004, pp. 167-172.

[7] D. McSherry, *Retrieval Failure and Recovery in Recommender Systems*, in Artificial Intelligence Review, vol. 24, Norwell, USA, 2005, pp. 319-338.

[8] R. Reiter, *A theory of diagnosis from first principles*, in Artificial Intelligence, vol. 32(1), Essex, UK, 1987, pp. 57-95.

[9] B. OSullivan, A. Papadopoulos, B. Faltings and P. Pu, *Representative Explanations for Over-Constrained problems*, in Proceedings of the National Conference on Artificial Intelligence, 2007, pp. 323-328.

[10] S. Wassermann and K. Faust, *Social Network Analysis*, Cambridge University Press, Cambridge, 1994.