Short paper

Learning Games for Configuration and Diagnosis Tasks

Alexander Felfernig[1] and **Michael Jeran**[1] and **Thorsten Ruprechter**[1] and **Alexander Ziller**[1] and **Stefan Reiterer**[1] and **Martin Stettinger** [1]

Abstract. A goal of many Artificial Intelligence (AI) courses is to teach properties of synthesis and analysis tasks such as *configuration* and *diagnosis*. Configuration is a special case of design activity where the major goal is to identify configurations that satisfy the user requirements and are consistent with the configuration knowledge base. If the requirements are inconsistent with the knowledge base, changes (repairs) for the current requirements have to be identified. In this paper we present games that can, for example, be used within the scope of Artificial Intelligence courses to easier understand configuration and diagnosis concepts. We first present the CONFIGURATIONGAME and then continue with two further games (COLORSHOOTER and EATIT) that support the learning of model-based diagnosis concepts.

1 Introduction

Theoretical problems of combinatorial games have long been studied [8]. For example, Bodlaender [4] analyzes the properties of coloring games were players have to color vertices of a graphs in such a way that never two adjacent vertices have the same color. The player who was last able to color a vertex in a consistent fashion wins the game. Börner et al. [5] analyze complexity properties of different variants of two-person constraint satisfaction [12] games were, for example, two players alternately make moves and the first player tries to find a solution whereas the second player tries to make the constraint satisfaction problem (CSP) inconsistent. Different complexity classes of such games are analyzed which primarily depend on the allowed quantifiers – quantified constraint satisfaction problems (QCSPs) are constraint satisfaction problems were some of the variables are universally quantified [9].

Bayer et al. [2] present an application that models Minesweeper puzzles as a CSP [12]; the game supports players in finding a solution and is primarily used as means to support students in understanding the mechanisms of constraint-based reasoning. In a similar fashion, Simonis [14] shows how to solve Sudoku puzzles on the basis of constraint technologies. In addition to problem solving approaches, the authors also focus on mechanisms for puzzle generation and propose measures for evaluating puzzle complexity. Finally, we want to mention the application of constraint technologies in the context of the generation of crossword puzzles. In crossword puzzle generation [3], a crossword puzzle grid has to be filled with words from a dictionary in such a way that none of the words in the dictionary is included more than once in the grid.

In the line of previous work, we present the CONFIGURATION GAME which is based on conventional CSP representations [12] and was implemented with the goal to support the learning of basic concepts of knowledge-based configuration [6, 15]. Furthermore, we introduce two games which focus on analysis in terms of model-based diagnosis [13]. COLORSHOOTER and EATIT are based on the ideas of model-based diagnosis and were developed to support students in the understanding of the principles of hitting set determination [13]. To the best of our knowledge these are new types of games based on conflict detection [10] and model-based diagnosis [7, 13]. All the presented games are *serious games* [11] with the purpose of teaching AI knowledge and also domain knowledge (EATIT).

The remainder of this paper is organized as follows. In Section 2 we introduce definitions of a configuration and a corresponding diagnosis task. The subsequently presented games are discussed on the basis of these definitions. In Section 3 we introduce the CONFIGURATIONGAME Android app and present the results of a corresponding user study. In Section 4 we introduce the COLORSHOOTER diagnosis game and also present results of a user study. In Section 5 we introduce a new diagnosis game embedded in the domain of healthy eating. In Section 6 we discuss issues for future work. With Section 7 we conclude the paper.

2 Configuration and Diagnosis Task

Knowledge-based Configuration is one of the most successful technologies of Artificial Intelligence [6, 15]. Configurators determine configurations for a given set of user requirements, for example, on the basis of constraint technologies. In terms of a CSP, a configuration task and a corresponding solution can be defined as follows.

Definition 1 (Configuration Task and Solution). A configuration task can be defined as a constraint satisfaction problem (V, D, C) where $V = \{v_1, v_2, ..., v_n\}$ is a set of variables, $D = \cup dom(v_i)$ represents the corresponding domain definitions, and $C = \{c_1, c_2, ..., c_m\}$ is a set of constraints. Additionally, user requirements are represented by a set of constraints $CREQ = \{r_1, r_2, ..., r_k\}$. A solution for a configuration task is an assignment $S = \{inst(v_1), inst(v_2), ..., inst(v_n)\}$ where $inst(v_i) \in dom(v_i)$ which is consistent with the constraints in $C \cup CREQ$.

Example (Configuration Task and Solution). An example of a very simple configuration task (and a corresponding solution S) represented as a constraint satisfaction problem is the following. Such configuration tasks have to be solved by players of the CONFIGURATION GAME (see Section 3). This example represents a simple Map Coloring Problem were variables ($V = \{v_1, v_2, v_3\}$) represent, for example, countries on a map and the constraints ($C = \{c_1, c_2\}$) restrict solutions to colorings were neighborhood countries must be represented by different colors. In our example we assume that the neighborhood countries are $\{v_1, v_2\}$ and $\{v_2, v_3\}$ and the user requirements are $CREQ = \{r_1 : v_1 = 1\}$. A player of the CONFIG-

[1] Graz University of Technology, Institute for Software Technology, Austria, email: {felfernig, jeran, reiterer, stettinger}@ist.tugraz.at {thorsten.ruprechter, alexander.ziller}@student.tugraz.at

URATIONGAME has successfully solved a configuration task (found a solution $\underline{S}$) if consistent($S \cup CREQ \cup C$).

- $V = \{v_1, v_2, v_3\}$
- $dom(v_1) = dom(v_2) = dom(v_3) = \{1, 2\}$
- $C = \{c_1 : v_1 \neq v_2,\ c_2 : v_2 \neq v_3\}$
- $CREQ = \{r_1 : v_1 = 1\}$
- $\underline{S} = \{v_1 = 1,\ v_2 = 2,\ v_3 = 1\}$

In configuration scenarios it is often the case that no solution can be found for a given set of user requirements ($CREQ \cup C$ is inconsistent). In this context, users are in the need of additional support in order to be able to identify reasonable changes to the current set of requirements more efficiently. Model-based diagnosis [13] can help to automatically identify minimal sets of requirements that have to be deleted (or adapted) such that a solution can be identified. A diagnosis task related to the identification of faulty requirements can be defined as follows (see Definition 2).

Definition 2 (Diagnosis Task and Diagnosis). A diagnosis task can be defined by a tuple $(C, CREQ)$ where C represents a set of constraints and $CREQ$ represents a set of customer requirements. If the requirements in $CREQ$ are inconsistent with the constraints in C, a diagnosis Δ ($\Delta \subseteq CREQ$) represents a set of requirements such that $CREQ - \Delta \cup C$ is consistent (in this context we assume that the constraints in C are consistent). Δ is minimal if $\neg\exists\Delta' : \Delta' \subset \Delta$.

Example (Diagnosis Task and Diagnosis). An example of a simple diagnosis task and a corresponding diagnosis Δ is the following. Similar diagnosis tasks have to be solved by players of the the COLORSHOOTER and the EATIT game (see Sections 4 and 5). The following example represents a diagnosis task were the set of customer requirements ($CREQ$) is inconsistent with the constraints in C. A player of COLORSHOOTER and EATIT has successfully solved a diagnosis task (found a diagnosis $\underline{\Delta}$) if $CREQ - \Delta \cup C$ is consistent.

- $V = \{v_1, v_2, v_3\}$
- $dom(v_1) = dom(v_2) = dom(v_3) = \{0, 1\}$
- $C = \{c_1 : \neg(v_1 = 1) \vee \neg(v_2 = 1), c_2 : \neg(v_1 = 1) \vee \neg(v_3 = 1), c_3 : \neg(v_2 = 1) \vee \neg(v_3 = 1)\}$
- $CREQ = \{r_1 : v_1 = 1, r_2 : v_2 = 1, r_3 : v_3 = 1\}$
- $\underline{\Delta} = \{r_1, r_2\}$

A wide-spread approach to determine diagnoses for a given diagnosis task is to identify minimal conflict sets [10] in CREQ and to resolve these conflicts on the basis of a hitting set directed acyclic graph (HSDAG) approach [13]. A (minimal) conflict set can be defined as follows (see Definition 3).

Definition 3 (Conflict Set). A set $CS \subseteq CREQ$ is a conflict set if $CS \cup C$ is inconsistent (C is assumed to be consistent). CS is minimal if $\nexists CS'$ with $CS' \subset CS$.

On the basis of a set of identified minimal conflict sets [10] we are able to automatically determine the corresponding minimal diagnoses (see Figure 1). In our example, the minimal conflict sets are $CS_1 : \{r_1 : v_1 = 1, r_2 : v_2 = 1\}, CS_2 : \{r_2 : v_2 = 1, r_3 : v_3 = 1\}$, and $CS_3 : \{r_1 : v_1 = 1, r_3 : v_3 = 1\}$. The corresponding minimal diagnoses are $\Delta_1 : \{r_1, r_2\}, \Delta_2 : \{r_1, r_3\}$, and $\Delta_3 : \{r_2, r_3\}$. Exactly this example pattern is implemented in the diagnosis games presented in Section 4 and Section 5.

3 CONFIGURATIONGAME

User Interface. With this game (see Figure 2), one should be able to gain first insights into the basic concepts of knowledge-based configuration. Constraints of the underlying CSP are depicted in the upper left corner. Constraints represent incompatible combinations of values, i.e., combinations that must not be positioned on adjacent vertices of the grid depicted in Figure 2. This way, tasks such as the map coloring problem [4] can be defined as a simple configuration problem (similar examples can also be found in [6]).

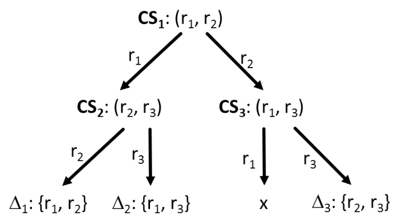

Figure 1. Example hitting set directed acyclic graph derived from the conflict sets CS_1, CS_2, and CS_3.

Each individual task to be solved by a player can be interpreted as a configuration task (V,D,C) (see Definition 1). In the setting shown in Figure 2, $V = \{v_1, v_2, ..., v_{14}\}$ represents a set of *14 interconnected hexagons* (in the center of the user interface). Furthermore, it is assumed that each variable has the same domain (in Figure 3, dom(v_i)={1,2,3,4}) and possible variable instantiations are represented by the values (hexagons) in the lower right corner. Constraints $C = \{c_1, c_2, ..., c_{16}\}$ represent incompatible colorings of adjacent vertices, for example, $c_1 : \neg(v_1 = 1 \wedge v_2 = 1) \wedge \neg(v_1 = 2 \wedge v_2 = 2) \wedge \neg(v_1 = 3 \wedge v_2 = 3) \wedge \neg(v_1 = 4 \wedge v_2 = 4) \wedge \neg(v_1 = 4 \wedge v_2 = 3) \wedge \neg(v_1 = 3 \wedge v_2 = 4) \wedge \neg(v_1 = 1 \wedge v_2 = 2) \wedge \neg(v_1 = 2 \wedge v_2 = 1)$. Incompatibilities are defined by red lines between individual values (left upper corner of Figure 2). A self-referring red line expresses an incompatibility on the same value, i.e., two adjacent vertices must not have to the same value. A proprietary constraint solver is used to generate individual tasks with increasing complexity in terms of the grid size (vertices and arcs) and the number of possible values and also to check proposed solutions for consistency.

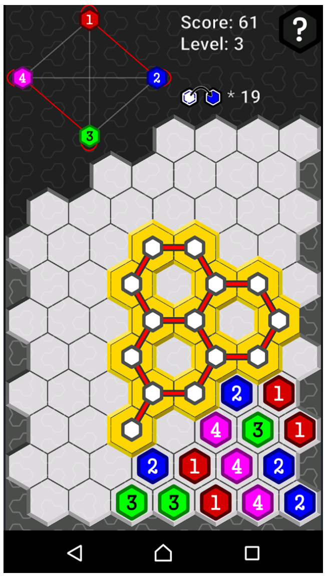

Figure 2. CONFIGURATIONGAME user interface.

The task of a player is to move values (hexagons) from the bottom to corresponding (empty) hexagons depicted in the middle of Figure 2. A player has found a solution if the grid instantiation S is

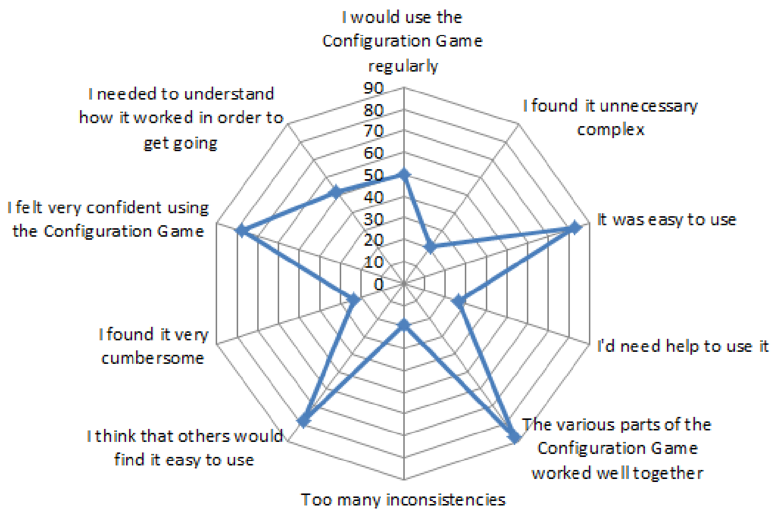

Figure 3. Results of a usability analysis of the CONFIGURATIONGAME on the basis of the system usability scale (SUS) [1].

consistent with the constraints in C.[2] A screenshot of an intermediate state of the CONFIGURATIONGAME is shown in Figure 4. The CONFIGURATIONGAME allows to define constraints that go beyond typical patterns of map coloring problems [4] since different types of incompatible adjacent vertices can be defined (in contrast to the map coloring problem where only incompatibilities regarding the same value (color) are defined). Instances of the configuration game are generated automatically.

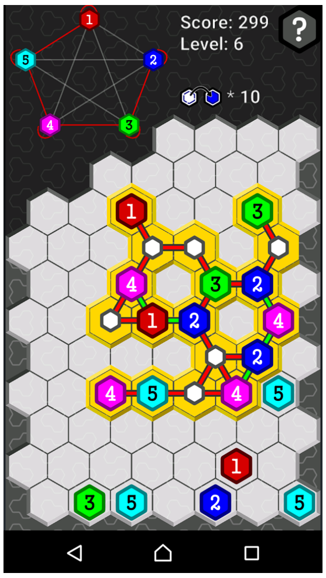

Figure 4. CONFIGURATIONGAME user interface (after the completion of some configuration steps).

Empirical Study. N=28 subjects of a usability study evaluated the CONFIGURATIONGAME. A first prototype of the game was made available to the subjects in the *Google Play Store*. The questionnaire was based on the system usability scale (SUS) [1] and thus focused on analyzing usability aspects of the system under investigation. The system was considered as easy to understand and well integrated. Results of the study are summarized in Figure 3.

4 COLORSHOOTER

User Interface. The COLORSHOOTER game (see Figure 5) focuses on providing first insights into the concepts of model-based diagnosis [13]. The game is available online in the *Apple App Store* (as an iOS application). The columns of the game represent minimal conflict sets (see Definition 3) – related diagnoses (see Definition 2) are represented by minimal color[3] sets such that at least one color from each row is included. The game consists of twenty different levels and inside each level of 30 different individual COLORSHOOTER tasks. Individual tasks are pre-generated in an automated fashion where the #colums, #rows, # of different colors, and diagnosis cardinality are major impact factors for determining the complexity of one COLORSHOOTER instance. Correct solutions (diagnoses) are pre-generated on the basis of a HSDAG [13].

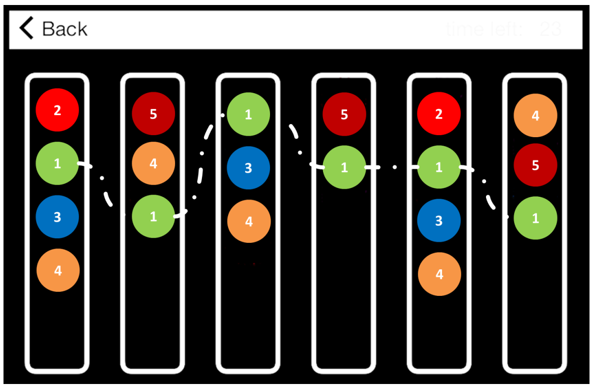

Figure 5. Example of a COLORSHOOTER diagnosis task.

Empirical Study. After two lecture units on model-based diagnosis [13], we investigated in which way COLORSHOOTER type games can actively support a better understanding of the principles

[2] In the CONFIGURATIONGAME we assume that $CREQ = \{\}$.

[3] For readability purposes we annotated the colored circles with numbers.

of model-based diagnosis. N=60 subjects (students) participated in a user study where each participant was assigned to one of three different settings (see Table 1).

Participants of the first setting used the COLORSHOOTER game directly before solving two diagnosis tasks. Participants of the second setting interacted with COLORSHOOTER one day before completing the two diagnosis tasks. Finally, participants of the third setting never interacted with COLORSHOOTER but only solved the two diagnosis tasks. The two diagnosis tasks where designed in such a way that a participant had to figure out all minimal diagnoses for each of two predefined inconsistent constraint sets (C_1 and C_2). C_1 included 4 constraints, 4 variables of domain size [1..3], and 3 related diagnoses. C_2 included 5 constraints, 4 variables of domain size [1..3] and 6 related diagnoses. Preliminary results in terms of the number of successfully completed diagnosis tasks are depicted in Table 1. In the case of the more complex constraint set C_2 we can observe a performance difference between users who applied COLORSHOOTER and those who did not.

setting	all minimal diagnoses found (C_1)	all minimal diagnoses found (C_2)
played directly before	33%	20%
played one day before	20%	20%
did not play before	23%	10%

Table 1. User study with three different settings: subjects played directly before solving two diagnosis tasks, subjects played one day before, and subjects did not use COLORSHOOTER.

5 EATIT

EATIT (see Figure 6) is an application currently under development, i.e., no related user studies have been conducted up to now. The major ideas of the game are the following: (1) similar to COLORSHOOTER, students should be able to more easily understand the concepts of model-based diagnosis. (2) there is a serious game [11] line of learning which is directly related to the underlying application domain: users of the system should learn about, for example, which vitamins are contained in which food. In EATIT, "conflicts" are represented by food items assigned to the same shelf (each food item contains the vitamin represented by the shelf) and diagnoses represent minimal sets of food items that are needed to cover all vitamins.

6 Future Work

In the CONFIGURATIONGAME our major goal for future work is to extend the expressiveness of constraints that can be defined for configuration tasks. A higher degree of expressiveness will allow the inclusion of further tasks such as scheduling and resource balancing. Furthermore, EATIT will be extended with functionalities that help to include user preferences and menu quality. In the current version of EATIT such aspects are not taken into account. In our future research we will also analyze in more detail which specific game types better help to increase understandability. Furthermore, we will analyze to which extent the games can be exploited to develop better configurator user interfaces and interaction schemes.

7 Conclusions

The overall goal of the (serious) games presented in this paper is to help to better understand the concepts of configuration and model-based diagnosis. Results of empirical studies are promising in the sense that the apps are applicable and can have a positive impact on the learning performance. Two of the presented games are already available: COLORSHOOTER in the Apple App Store and the CONFIGURATIONGAME in the Google Play Store.

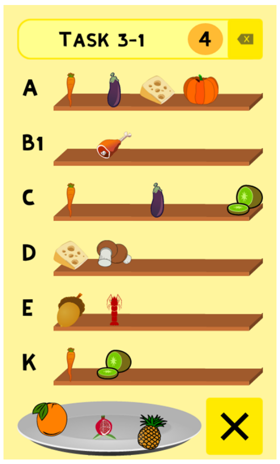

Figure 6. Screenshot of EATIT. Each shelf represents food that contains a specific vitamin (e.g., A, B1, ...). A solution (diagnosis) is found if the selected food on the plate covers all vitamins.

REFERENCES

[1] A. Bangor, P. Kortum, and J. Miller, 'An Empirical Evaluation of the System Usability Scale (SUS)', *International Journal of Human-Computer Interaction*, **24**(6), 574–594, (2008).
[2] K. Bayer, J. Snyder, and B. Choueiry, 'An Interactive Constraint-Based Approach to Minesweeper', in *AAAI 2006*, pp. 1933–1934, (2006).
[3] A. Beacham, X. Chen, J. Sillito, and P. vanBeek, 'Constraint Programming Lessons Learned from Crossword Puzzles', in *AI 2001*, LNAI, pp. 78–87. Springer, (2001).
[4] H. Bodlaender, 'On the Complexity of Some Coloring Games', *LNCS*, **484**, 30–40, (1991).
[5] F. Börner, A. Bulatow, H. Chen, P. Jeavons, and A. Krokhin, 'The Complexity of Constraint Satisfaction Games and QCSP', *Information and Computation*, **207**(9), 923–944, (2009).
[6] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Elsevier/Morgan Kaufmann Publishers, 1st edn., 2014.
[7] A. Felfernig, M. Schubert, and C. Zehentner, 'An efficient diagnosis algorithm for inconsistent constraint sets', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, **26**(1), 53–62, (2012).
[8] A. Fraenkel, 'Complexity, Appeal and Challenges of Combinatorial Games', *Theoretical Computer Science*, **313**(3), 393–415, (2004).
[9] I. Gent, P. Nightingale, and K. Stergiou, 'A Solver for Quantified Constraint Satisfaction Problems', in *IJCAI 2005*, pp. 138–142, (2005).
[10] Ulrich Junker, 'QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems', in *19th Intl. Conference on Artifical Intelligence (AAAI'04)*, eds., Deborah L. McGuinness and George Ferguson, pp. 167–172. AAAI Press, (2004).
[11] H. Kelly, K. Howell, E. Glinert, L. Holding, C. Swain, A. Burrowbridge, and M. Roper, 'How to build Serious Games', *Communications of the ACM*, **50**(7), 44–49, (2007).
[12] A. Mackworth, 'Consistency in Networks of Relations', *Artificial Intelligence*, **8**(1), 99–118, (1977).
[13] R. Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **32**(1), 57–95, (1987).
[14] H. Simonis, ' Sudoku as a constraint problem', in *CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems*, pp. 13–27, (2005).
[15] M. Stumptner, 'An Overview of Knowledge-based Configuration', *AI Communications*, **10**(2), 111–126, (1997).