Generating Tradeoffs for Interactive Constraint-Based Configuration

Eugene C. Freuder[1]* and Barry O'Sullivan[2]

[1] Cork Constraint Computation Centre
University College Cork, Ireland
e.freuder@4c.ucc.ie
[2] Department of Computer Science
University College Cork, Ireland
b.osullivan@cs.ucc.ie

Abstract. In this paper we have modeled tradeoffs in constraint-based configuration as additional constraints, and begun to study the issues involved in generating and evaluating such tradeoffs. We describe our basic approach in the context of a "toy" configuration problem based on the classic N-Queens problem. Initial experiments compare "proposal strategies" for generating tradeoffs. We demonstrate that arc-consistency can be used as an effective trigger for generating tradeoff proposals in interactive configuration.

1 Introduction

Configuration is becoming a well studied design activity [5]. While there has been a growing interest in issues such as diagnosis of knowledge-bases for configuration [2], advice generation for design [1] and explanation generation [3], there is still a need for work on techniques which learn user's preferences and use these to assist users achieve satisfactory configurations. This paper presents initial steps towards the development of such techniques.

During an interactive configuration session we may reach a point where our desires cannot be met. At this point we can consider "tradeoffs". For example, in configuring a camera, we find that it is impossible to get one weighing less that 10 ounces with a zoom lens of 10X or more, so we consider a tradeoff: "I will increase my weight limit to 14 ounces if I can have a zoom lens of 10X or more." Ideally, we would like the configurator to suggest appropriate tradeoffs to us.

We have modeled tradeoffs in constraint-based configuration as additional constraints, and begun to study the issues involved in generating and evaluating such tradeoffs [4]. In Section 2, we describe our basic approach in the context of a "toy" configuration problem. We utilize the classic N-Queens problem, with the addition of user-generated "preference constraints" and system-generated

* This work was performed while this author was at the University of New Hampshire.

"tradeoff constraints". Initial experiments compare "proposal strategies" for generating tradeoffs. Our results can be summarized as follows. Firstly, users who have strong preferences can be more successfully assisted in finding acceptable solutions to a configuration problem. Secondly, when users have strong preferences, arc-consistency is a sufficient trigger for proposing tradeoffs to the user to successfully overcome inconsistency. This is a particularly interesting result since techniques based on arc-consistency are extremely useful in industry-based configurators. Finally, a number of concluding remarks are made in Section 3.

2 A Case Study

The configuration problem that will be studied here is based on the N-Queens problem. The user attempts to solve the configuration problem by interactively specifying a series of preference constraints to a constraint-based configurator. We assume that the user prefers higher column values. Thus, when the user proposes a preference constraint we will assume it to be of the form $row \geq column$, where row corresponds to a row number in the N-Queens problem and $column$ corresponds to the column value for that row. For example, the constraint $4 \geq 6$ means that the queen on row 4 should be placed in a column whose value is at least 6.

During an interactive session with the configurator, the user may specify a constraint which causes the set of preference constraints to become "overconstrained", identified using some measure of consistency (e.g the problem becomes arc-inconsistent). At this point our configurator attempts to recommend a set of appropriate "tradeoff" constraints to the user which she can accept before continuing to develop a solution for the configuration problem.

User-specified preference constraints are modeled as unary constraints. On the other hand, tradeoffs are modeled as binary constraints. During an interactive session with our configurator, when a tradeoff constraint, involving row x and y, is accepted, it replaces the unary preference constraints involving these variables.

2.1 Tradeoff Proposal Strategies

We have considered a number of tradeoff proposal strategies. These will be briefly outlined below:

- *Maximum Sum of Column Values* – this strategy proposes a set of tradeoff constraints, each of whose sum of column values is maximal and is *arc-consistent*.
- *Maximum Sum of Viable Column Values* – this strategy generates a set of tradeoff constraints, each of whose sum of column values is maximal and could yield a *solution* to the configuration problem.
- *Maximum Viability* – this strategy generates a set of tradeoff constraints which have the potential to yield the greatest number of solutions (maximally viable) to the configuration problem.

- *Minimum Viability* – this strategy generates a set of tradeoff constraints which could yield at least one solution (minimally viable) to the configuration problem.
- *Pareto optimality* – this strategy generates a set of tradeoff constraints which could yield a Pareto optimal solution to the configuration problem.

2.2 Evaluation of Proposal Strategies

Experiments were made on the 8-Queens problem. The experiments involved simulating the interaction between a human user and a configurator.

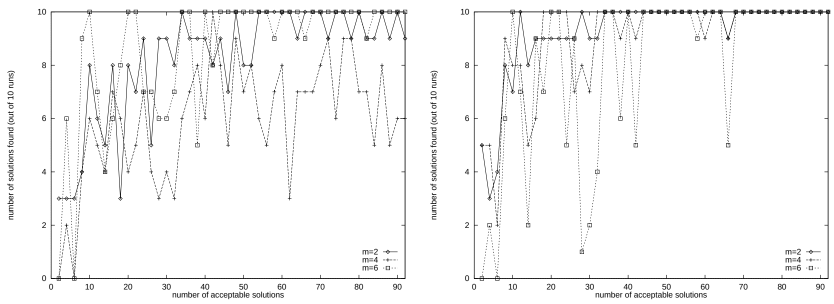
We simulated the human user on a combination of two axes. Firstly, we considered the set of solutions to the problem which the user would find acceptable as an experimental axis. Different points on this axis were chosen and solutions were generated at random for each point. Our simulated user would accept a tradeoff if it permitted a solution in the "set of acceptable solutions".

Secondly, we modeled the "strength" (greediness) of the user's preference constraints. Therefore, we considered the strength (m) of the simulated user's constraints as an experimental axis. Based on this axis our simulated user proposes new preference constraint bounds randomly chosen between m and N (for N-Queens) for different values of m. In our experiments we chose points along this axis from the set $\{2, 4, 6\}$. Thus, if we were simulating a user whose preference constraints had strength 4, a new constraint on row x would be of the form $x \geq y$, where y would be randomly chosen from the set of integers between 4 and 8 (for 8-Queens).

We simulated an interactive configurator. The configurator accepted user-specified preference constraints and incorporated these into the "built-in" model of the Queens problem. Preference constraints were accepted from the simulated user while they were consistent. If the simulated user managed to solve the problem without encountering an inconsistency, tradeoffs were never proposed. When the user proposed a preference constraint, $x \geq i$, which had the effect of introducing an inconsistency into the model, the configurator proposed a set of tradeoffs to the user. We used two different measures of consistency to trigger the proposal of tradeoffs: arc-consistency and full viability checking. If the user accepted one of the proposed tradeoff constraints, $C_{(x,y)}$, the unary constraints on x and y in the user's set of preference constraints were replaced with a single binary constraint representing the accepted tradeoff. If the set of tradeoff constraints proposed to the user was empty, or none was acceptable to the user, this was regarded as a failure. In the experiments presented here we assumed that the user could not "backup" by retracting a previous preference constraint or tradeoff. We are currently addressing the issue of revisiting previous decisions.

2.3 Most Significant Results

Of the five tradeoff proposal strategies evaluated, two are of particular interest: maximum sum of viable column values and minimum viability. The results for these will be discussed below.



(a) Generating tradeoffs when arc-inconsistent

(b) Generating tradeoffs when viability check fails

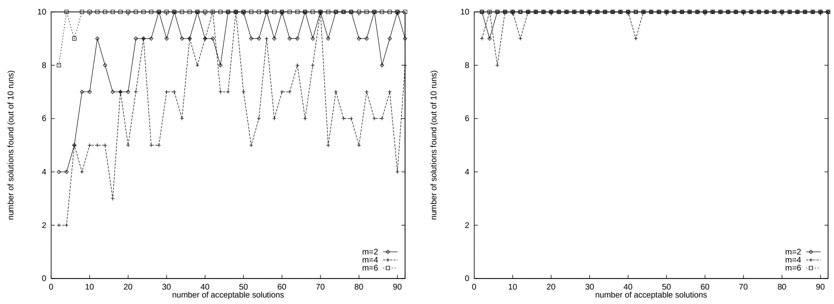
Fig. 1. The performance of the "maximum sum of column values with full viability checking" strategy.

The "maximum sum of viable column values" strategy only recommends tradeoffs which it knows could lead to a full solution to the problem. Thus, while greedy, it attempts to work with the distributions found in the solutions to the N-Queens problem. The performance of this strategy is presented in Figure 1. One of the most interesting aspects of the performance of this strategy is that its performance is quite volatile for smaller sets of acceptable solutions. This can be seen quite clearly in Figure 1(b), but is also evident in Figure 1(a) for user preference strengths of 2 (m=2) and 6 (m=6). The fact that either high strength (m=6) and low strength (m=2) preferences are more successful at finding solutions (Figure 1(a)) is a consequence of the tradeoffs working with both the distribution of solutions in the Queens problem and their symmetries. It should also be noted that for high strength preferences (m=6), solutions are found consistently regardless of the type of consistency check used to trigger tradeoff proposals. This is significant since it implies that arc-consistency can be used to generate good tradeoff proposals to the user. Arc-consistency is of much more practical use than full viability checking.

The "minimum viability" strategy found acceptable solutions for almost every point along the axis of number of acceptable solution when using a high preference strength Figure 2). However, since this strategy generates tradeoffs which are minimally viable there is a disadvantage associated with it, namely, that a large number of tradeoffs are proposed. In an interactive environment this could cause a certain amount of information overload from the perspective of the user. However, it may be possible to alleviate the problems associated with this by using an anytime approach to generating tradeoffs based on "minimum viability".

3 Conclusion

The ability of configurators to generate tradeoffs to users during interactive configuration is valuable. In this paper we have modeled tradeoffs in constraint-based



(a) Generating tradeoffs when arc-inconsistent

(b) Generating tradeoffs when viability check fails

Fig. 2. The performance of the "minimum viability" strategy.

configuration as additional constraints, and begun to study the issues involved in generating and evaluating such tradeoffs. We have found that users who have strong preferences can be more successfully assisted in finding acceptable solutions to a configuration problem. In addition, when users have strong preferences, arc-consistency is a sufficient trigger for proposing tradeoffs to the user to successfully overcome inconsistency. This is a particularly interesting result since techniques based on arc-consistency are extremely useful in industry-based configurators.

Acknowledgments

This work was supported in part by Trilogy. Professor Freuder is supported by a Principal Investigator Award from Science Foundation Ireland.

References

1. James Bowen. Using dependency records to generate design coordination advice in a constraint-based approach to Concurrent Engineering. *Computers in Industry*, 33(2–3):191–199, 1997.
2. Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumpter. Consistency-based diagnosis of configuration knowledge-bases. In *Proceedings of the 14h European Conference on Artificial Intelligence*, pages 146–150, 2000.
3. Eugene C. Freuder, Chavalit Likitvivatanavong, and Richard J. Wallace. A case-study in explanation and implication. In *In CP2000 Workshop on Analysis and Visualization of Constraint Programs and Solvers*, 2000.
4. Eugene C. Freuder and Barry O'Sullivan. Generating tradeoffs for constraint-based configuration. In *Working Notes of the CP-2001 Workshop on User-Interaction in Constraint Satisfaction*, December 2001.
5. Daniel Sabin and Rainer Weigel. Product configuration frameworks – a survey. *IEEE Intelligent Systems and their applications*, 13(4):42–49, July–August 1998. Special Issue on Configuration.