# Towards engineer-to-order product configuration

Xie, H.; Lau, F.

# Towards engineer-to-order product configuration

**IMTI-XP-105**

Xie, H.; Lau, F.

A version of this document is published in / Une version de ce document se trouve dans:
15th International Conference on Computer Application in Industry and Engineering, San Diego, CA., Nov. 1, 2002, pp. 1-5

# Towards Engineer-to-order Product Configuration

**Helen Xie and Foster Lau**
**Integrated Manufacturing Technologies Institute**
**National Research Council Canada**
**800 Collip Circle, London, Ontario, Canada N6G 4X8**
**Helen.Xie@nrc.ca and Foster.Lau@nrc.ca**

## Abstract

Constraint satisfaction problem (CSP) paradigm has proved highly successful for solving product configuration problems, particularly for build-to-order configuration in determining combination of different pre-defined component types. However, many real world customized products deal with engineer-to-order configuration, where product configuration is generated not only from combinations of different types of components, but also from varying different design parameters to satisfy pre-defined design constraints. The challenge posted by engineer-to-order configuration stems from n-ary constraints with continuous variables. Consequently, incorporating n-ary constraints and continuous variables into constraint satisfaction paradigm typically involves expensive search. Therefore, effective search strategies along with a modeling approach supporting the search strategies are the main issues in solving engineer-to-order configuration. In this paper, we present an approach to engineer-to-order product configuration where configuration problems are represented as CSP with n-ary constraints and variables with both discrete and continuous domains. We extended CSP by introducing two types of variables, where controllable variables can be independently assigned with values and dependable variables are derived based on those controllable variables. By limiting it into a set of controllable variables, search space is considerably reduced, and therefore, search efficiency is significant improved. The search algorithm is based on repair strategy in which min-conflicts heuristic is applied along with backtracking search. A case study indicated that this approach could effectively solve engineer-to-order product configuration by reducing its search space.

Keywords: engineer-to-order, product configuration, constraint satisfaction, search algorithms, repair-based heuristic

## 1 INTRODUCTION

Market trends that affect today's competitive environment are changing dramatically. Customers demand products with lower prices, higher quality and faster delivery, but they also want products customized to match their unique needs [1]. To meet these demands, manufacturers have to adapt their business model to mass customization that allows customers to select, order and receive customized products, often choosing from among hundreds of product features and options, at a low cost. Successful implementation of mass customization, however, requires significant information technology capabilities across broad application supports. One of the key enabling technologies in the implementation is product configurators.

A product configurator is a software tool that captures customer's requirements as input and generates product configuration exactly matching a customer's specific needs, based on pre-defined design constraints. The application scope of the product configurator is regarded for routine design tasks, in which one keeps to known and established solution principles and adapts the concrete design to changed requirements [2]. Noticeably, there are two types of routine design tasks for accomplishing customized products: build-to-order (BTO) and engineer-to-order (ETO). The build-to-order configuration typically uses a set of pre-defined component types while taking into account a set of well-defined restrictions on how the component types can be combined [3]. The engineer-to-order configuration extends beyond build-to-order configuration, in which each component type is also associated with a pre-defined set of parameters, where each parameter has a predefined set of possible values [4]. The product configuration should satisfy the pre-defined design constraints among those design parameters as well.

Modeling configuration problems as Constraint Satisfaction Problems (CSP) [5] as well as its extension Dynamic Constraint Satisfaction Problems (DCSP) [6], Generative Constraint Satisfaction Problems [7][8], and Composite Constraint Satisfaction Problems (CCSP) [9], has proved highly successful for build-to-order product configuration problems. In a CSP, component types and their ports are represented as discrete variables with finite domains. Constraints among components restrict the ways various components can be combined to form a valid configuration. The configuration tasks are to assign values to all the variables without violating any constraints. The benefits using CSP to solve configuration problems is due to its declarative

representation structure, effective and flexible computability.

Most of the configuration research using CSP paradigm has been concentrated on build-to-order product configuration. However, engineer-to-order configuration problems have posted more challenges for CSP paradigm. In engineer-to-order configuration problems, constraints with n-ary variables are common, and constraint types among those variables are diverse and complex. Moreover, domains of the variables are usually continuous in a range. Most of the previous work on CSP algorithms has assumed binary constraints and variables with an enumerable discrete domain. Theoretically, a n-ary constraint can be easily approximated by a binary constraint by projecting the n-ary constraint onto the pairs of variables it contains. However, after continuous variables are converted into discrete variables with an interval, the domain of the variables usually becomes so large that the projection process itself is a very time consuming task. Furthermore, search space for such a large domain of variables has been considerably expanded. Therefore, conventional CSP search algorithms for binary constraints with discrete variables are not efficient enough for solving engineer-to-order product configuration problems. An effective search strategy should focus on reducing search space. In addition, many variables involved in engineer-to-order configuration are interdependent; changes on one variable would propagate changes on many other variables too. Therefore, an approach to effective update variables to accommodate constraint evaluation is essential to improve the efficiency of search algorithms.

To address the problems of solving engineering tasks represented as CSP, Gelle et al. [10] introduced a new type of local consistency for handling numeric and discrete variables to narrow down the search space effectively. In this paper, engineer-to-order configuration problems are represented as CSP with n-ary constraints and variables with both discrete and continuous domains. To reduce search space, we extended CSP by introducing two types of variables, where controllable variables can be assigned a value and dependable variables are derived based on controllable variables. Search space is only limited to controllable variables. The search algorithm is based on repair strategy in which mini-conflict heuristic is applied along with backtracking search [11]. This approach first generates a complete, but inconsistent assignment and then repairs constraint violations until a consistent assignment is achieved. In the next section, we introduce a framework as a CSP extension for engineer-to-order product configuration. We then introduce the search strategies for the framework in section 3. A case study for configuring an elevator system is presented in section 4. Section 5 is conclusions.

## 2 MODELING ENGINEER-TO-ORDER PRODUCT CONFIGURATION AS AN EXTENSION TO CSP

Before we define the engineer-to-order product configuration model, we would give a definition of typical constraint satisfaction problems.

**Definition 1**. Constraint Satisfaction Problem: A Constraint Satisfaction Problem (CSP) is defined as a triplet <X, D, C>, where

- $X = \{X_1, X_2, \ldots, X_n\}$ is a finite set of variables,
- Each $X_i$ can take its value from a finite domain $D_i$, where $D_i = \{D_{i1} \cup \ldots \cup D_{in}\}$, and
- A set of constraints C restricting the combination of values that variables can take.

A solution to a CSP is an assignment of a value from its domain to every variable from X, in such a way that every constraint from C is satisfied.

There are some limitations in representing engineering product configuration into CSP. First of all, variables in CSP are directly restricted by constraints. Hence, values can be directly selected for a variable to satisfy constraints. In engineering configuration problems, however, many design parameters, as variables cannot take an independent value, because they are also associated with other design parameters. Therefore, evaluating constraints involves the assignment of the associated design parameters. Secondly, the domain of variables in CSP is usually enumerable and discrete. However, most of the design parameters in engineering configuration problems are modeled as continuous domains. Continuous domains can be converted to discrete domains with an interval, but in terms of expanded search space, it would take a long time to search to find a satisfied solution. Hence, an effective model is essential for engineering configuration problems in order to achieve effective search performance.

By analyzing engineering configuration problems, we have recognized that many variables cannot take independent values because of their inherited constraints. If we let the search algorithm skip these variables, it will reduce huge search space. Based on this idea, we proposed a framework as an extension to CSP for representing engineering configuration problems. In the framework, the variables are first classified as controllable variables, dependable variables, and user-defined variables (Figure 2.1). The controllable variables can be independently assigned a value in their domains to satisfy constraints. The dependable variables are derived from one or more controllable variables, other dependable variables, and user-defined variables. The user-defined variables take input from user requirements as a value. Secondly, The constraints are classified as assignment

constraints, conditional constraints, range constraints, and user requirement constraints. The assignment constraints and conditional constraints are used to determine the value of dependable variables. The assignment constraints derive a dependable variable using formula. The conditional constraints are usually represented as "if – then" structure to derive a dependable variable. The range constraints specify the satisfaction criteria for variables, especially, dependable variables. If a dependable variable does not satisfy a range constraint, the corresponding controllable variables have to be modified, so that the resulting value of the dependable variable will satisfy the range constraint. The user requirement constraints are a set of unary constraints to restrict the user to assign a value from its domain to the user-defined variables. These variables are usually fixed once the user assigns one, but some of them could also be modified if no solution is found. Finally, the variables are clustered according to the component they belong to. By clustering variables into individual components, most variables can be solved independently within a component to avoid interaction and coupling among components.

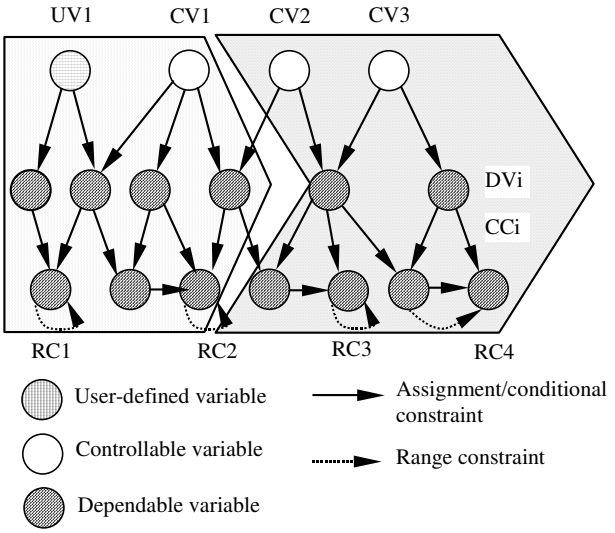


Figure 2.1 Variables and Constraints in Product Configuration Problems

In the following description, we will give definitions for engineer-to-order configuration as an extension of CSP. In terms of engineer-to-order products, a product configuration is defined by a set of components (CP) along with their associated design parameters and a set of constraints over those design parameters. The goal of the product configuration is to find a set of values for each design parameter that satisfy each constraint. The set of values for design parameters is referred to a valid configuration.

**Definition 2**. Engineer-to-order configuration EC is defined as a set <CP, DP, D, C>, where

- $CP = \{CP_1, \ldots, CP_n\}$ is a finite set of components,
- $DP_{CPi} = \{DP_{CPi1}, \ldots, DP_{CPin}\}$ is a finite set of design parameters belong to each component CPi,
- Each $DP_{CPij}$ can take its value from a finite domain $D_{CPi}$, where $D= \{D_{CPi1} \cup \ldots \cup D_{CPin}\}$,
- A set of constraints C restricting the combination of values the design parameters can take.

A solution to an engineer-to-order configuration EC is an assignment of a value from its domain to every design parameter from DP, in such a way that every constraint from C is satisfied.

**Definition 3**. Design parameter (DP): DP is a set of variables that fall in one of three categories: controllable variable (CV), and dependable variable (DV), and user-defined variable (UV). $DP = \{CV \cup DV \cup UV\}$. The domain of DP can be discrete or continuous. In case of continuous domain, it is usually converted to discrete domain with an interval. In either cases, D is a finite domain $D=\{V_1, \ldots, V_n\}$. In configuration process, CV can be assigned any value within its domain as $(CV_i = V)$, where $CV_i \in CV$, V is an element of the domain of $CV_i$. $DV_i$ may be derived from $CV= \{CV_i, \ldots, CV_j\}$, $UV= \{UV_i, \ldots, UV_j\}$, or other $DV = \{DV_1, \ldots, DV_{i-1}, DV_{i+1}, \ldots\}$ based on its assignment constraints and/or conditional constraints. UV is usually assigned a value from the user from its domain. $UV_i = V$, where $V \in D$.

The major difference between the framework for engineering configuration problems and typical constraint satisfaction problems is the ability to distinguish variables as controlled variables, dependable variables, and user-defined variables, so that the search space can be limited to the controlled variables and search efficiency can be improved.

## 3 REPAIR-BASED SEARCH STRATEGIES

Having defined engineer-to-order product configuration as an extension to CSP, we used repair based search strategy for solving the engineering product configuration. This approach first generates a complete, but inconsistence assignment for each variable, and then repairs violation constraints based on min-conflicts heuristic to achieve a valid solution. The approach is implemented through a product component module and a constraint-solving module.

The main function of the product component module is to update dependable variables for constraint evaluation. The design parameters, including controllable variables, dependable variables, and user-defined variables, are organized under their corresponding components. Given controllable variables and user-

defined variables, the module specifies how to derive dependable variables using assignment constraints and conditional constraints. Most of dependable variables are self-contained in their corresponding components, but a few of them also depend on variables from other components. In these cases, an observer pattern is applied for updating the dependable variables once new information becomes available. Assuming the sequence for deriving the dependable variables is from CP1, CP2, to CP3, the values for DV1 and DV2 are not correct without a current value for DV3 (Figure 3.1). The dependable variables DV1 and DV2 register themselves to DV3, and DV1 registers itself to DV2. Whenever a value for DV3 is modified, DV3 notifies DV1 and DV2 to update their values. DV1 and DV2 respond differently to the notification. Since DV2 depends only one external variable, it updates itself immediately. DV3 depends not only DV1 but also DV2. Therefore, it cannot update itself until it receives a notification from DV2.

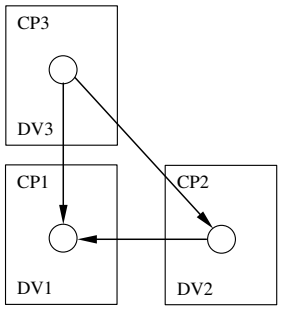


Figure 3.1 Process for updating dependable variables

The constraint-solving module constructs a configuration problem with a set of range constraints. Within each range constraint, the condition of constraint satisfaction is specified with a dependable variable. In addition, controllable variables required for fixing the constraint as well as their domains are also specified for each range constraint. A range constraint can usually be fixed by one or multiple controllable variables. The priority of the set of range constraints is assigned to the constraints with multiple controllable variables.

In solving a configuration problem, we start with an initial solution by deriving dependable variables based on user input and controllable variable assignment. Then each range constraint is evaluated and a list of violated constraints is found. The iterative process then starts by assigning a value for controllable variable in constraint-solving module, deriving related dependable variables in product component module, evaluating the dependable variable against a range constraint in constraint-solving module, until each constraint is satisfied at the same time. The search algorithm is described in details in Figure 3.2.

```
Procedure MIN-CONFLICT (CONFLICT SATISFIED)
  If CONFLICT is empty, then solution found, stop
  Let CON= a constraint in CONFLICT
  Remove CON from CONFLICT
  Put CON to SATISFIED
  Let VARS= list of controllable variables in CON
  For each VAR in VARS, until solution found
    Let VALUES= list of possible values in VAR
    For each VALUE in VALUES
      Update dependable variables
      If VALUE and the dependable variables do not
      conflict with any constraint in SATISFIED
      Then assign VALUE to VAR
      Call MIN-CONFLICT (CONFLICT SATISFIED)
      End if
    End for
  End for
End procedure

Begin program
  Let CONSTRAINTS= list of range constraints
  Assign user-defined variables from user input
  Initiate controllable variables with the first
  value in their domains
  Update dependable variables
  Evaluate the assignment of variables against
  CONSTRAINTS
  Let CONFLICT= list of conflict constraints
  Sort CONFLICT with priority on the constraints
  with multiple controllable variables
  Let SATISFIED= list of satisfied constraints
  Call MIN-CONFLICT (CONFLICT SATISFIED)
End program
```

Figure 3.2 Search algorithm using min-conflicts heuristic

The search algorithm uses min-conflicts heuristic and combines with backtracking to search the solution space systematically for the configuration problem. Therefore, a solution can be found if existing. On the other hand, the search space is limited to controllable variables related violated constraints up to all controllable variables, so that the search space is effective reduced comparing to searching for all of the design parameters. Therefore, the algorithm is appropriate for solving complex engineer-to-order product configuration problems.

## 4 A CASE STUDY AND EXPERIMENTAL RESULTS

To illustrate how above CSP model and search algorithm work with the complexity of engineer-to-order configuration problems, we introduce a benchmark problem—configuring elevator systems [12] for a case study. To configure an elevator system, one must assemble a collection of components that satisfies both customer requirements and design constraints. The configuration process begins with a list of customer requirements, such as elevator car capacity and speed, and building dimensions. The configurator then selects an appropriate set of elevator components and assigns design parameters to each component. In engineer-to-order product configuration, not all components are compatible, and certain combinations will not meet functional or

safety regulation. The configurator has to modify design parameters until achieving a satisfactory configuration.

In order for the product configurator to find a valid solution for the elevator design, it is necessary to generate product definitions in the product component module and the constraint-solving module. In a cable-operated elevator system, there are 18 major components, such as hoistway, car assembly, counterweight assembly, suspension system, safety mechanisms, and cables. Along with the major components, there are 241 associated design parameters. Of those design parameters, there are 25 user-defined variables (such as car capacity and car speed), 32 controllable variables (such as platform model, and counterweight buffer quantity), and 184 derived variables (such as counterweight quantity, and hoist cable quantity). In addition, associated with the derived variables, there are assignment constraints, conditional constraints and user-defined constraints to control how the derived variables gain their values from controllable variables or user-defined variables. This product definition information should be generated in the product component module. On the other hand, the constraint-solving module should contain information for 50 range constraints and the responding controllable variables. This information establishes criteria for functional and safety regulations and guide the configurator to find a valid solution.

We implemented a prototype system for the proposed approach using Java and web-based architecture. The system allows the customer to enter requirements and displays the final configuration back to the customer through the Web. Most of cases we have tested returned configuration results between 20 to 50 seconds. The system was deployed on IBM WebShpere Application Server with Windows 2000 operating system on an Intel Pentium 4 CPU, 1.4GHz, and 512M RAM.

## 5 CONCLUSION

Customized products offer great potential to manufacturers in global market competition and improved customer satisfaction. The complexity of engineer-to-order product created new demands for configuration technology to cope with search efficiency. However, today's configuration systems only support build-to-order product configuration and cannot meet increased complexity of engineer-to-order product configuration. In this paper, we present a framework for engineer-to-order product configuration as an extension of CSP paradigm. The extension supports n-ary constraints and variables with both discrete and continuous domains. We distinguish controllable variables and dependable variables to reduce search space. The search algorithm is based on repair strategy in which min-conflicts heuristic is applied along with backtracking search. A web-based prototype system was built to test the efficiency of the proposed model and algorithm. The approach could be applied to general engineer-to-order configuration problems.

## REFERENCES


[1] R. W. Bourke, "Product Configurators: Key Enablers for Mass Customization," MIDRANGE ERP, <http://www.midrangeerp.com>, August 2000

[2] G. Paul and W. Beitz, Engineering Design: A Systematic Approach, Springer-Verlag, London, 1999

[3] T. Soininen, "Configuration Workshop Notes," the 17th International Joint Conference on Artificial Intelligence, Seattle, WA, August 2001

[4] K. Orsvarn and T. Axling, "The Tacton View of Configuration Tasks and Engines," AAAI'99 Workshop on Configuration, the 16th National Conference on Artificial Intelligence, Orlando, FL, July 1999

[5] S. Mittal and F. Frayman, "Towards a Generic Model of Configuration Tasks," In Proc. Of the 11th IJCAI, Detroit, MI, pp. 1395-1401, 1989

[6] S. Mittal and B. Falkenhainer, "Dynamic Constraint Satisfaction Problems," In Proceedings AAAI Conference, pp. 25-32, 1990

[7] M. Stumptner and A. Haselbock, "A generative constraint formalism for configuration problems," 3rd Congress Italian Assoc. for AI, Torino, Italy, Lecture Notes in AI, Springer-Verlag. vol. 729, pp. 302-313, 1993

[8] G. Fleischanderl, G. Friedrich, A. Haselboeck, H. Schreiner and M. Stumptner, "Configuring Large Systems Using Generative Constraint Satisfaction," IEEE Intelligent Systems, vol. 13, pp. 59-68, July/August, 1998

[9] D. Sabin and E. C. Freuder, "Configurations as Composite Constraint Satisfaction," in Working Notes, AAAI Fall Symposium on Configuration, Boston, pp. 28-36, 1996

[10] E. Gelle, B. V. Faltings, D. E. Clement, and I. F. C. Smith, "Constraint Satisfaction Methods for Applications in Engineering," Engineering with Computers, Springer-Verlag, London Limited, vol. 16, pp. 81-85, 2000

[11] S. Minton, M. D. Johnston, and P. Laird, "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems," Artificial Intelligence, vol. 58, pp. 161-206, 1992

[12] G.R. Yost, and T. R. Rothenfluh, "Configuring Elevator Systems," Int. J. Human-Computer Studies, vol. 44, pp. 521-568, 1996