Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2000), *14*, 271–287. Printed in the USA.
Copyright © 2000 Cambridge University Press 0890-0604/00 $12.50

Interval constraint networks for tolerance analysis and synthesis

CHRISTOPHER C. YANG,[1] MICHAEL M. MAREFAT,[1] AND FRANK W. CIARALLO[2]

[1]Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721, USA
[2]Department of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ 85721, USA

(RECEIVED July 29, 1999; ACCEPTED October 23, 1999)

Abstract

This paper proposes interval constraint networks and interval propagation techniques for automatic tolerance design. To develop an intelligent automatic tolerance design mechanism, the following interrelated problems are addressed: (1) representing the relationships between the tolerances of the entities, attributes, and the functional requirements of a design component by a constraint network, (2) developing tolerance propagation techniques in the network that will satisfy the constraints between the entities, attributes and functional requirement. For the purpose of tolerance design, we propose new definitions for consistency of constraints and consistency of a constraint network. New forward and backward propagation techniques achieve worst-case tolerance analysis and synthesis, respectively. A tank design problem illustrates the use of the procedures for tolerance analysis and synthesis. The discussion and examples compare the new constraint network and propagation techniques with previous works.

Keywords: Computer Integrated Manufacturing; Interval Constraint Networks; Constraint Propagation; Tolerance Design

1. INTRODUCTION

Tolerance design plays an important role in the relationship between performance and the manufacture of a product. Decreasing the tolerance range may improve performance, but will also make the manufacture more difficult and/or increase manufacturing cost. It is desirable to maintain valid tolerances when product design is constrained by the relationship between the dimensions of entities of a component and the functional requirement of the design. In this paper, we develop a constraint-based reasoning mechanism to analyze design tolerances and synthesize a new set of tolerances to satisfy the functional requirements of a product.

For a given design of a mechanical part, a relationship can be derived for the functional requirement in terms of the entities. This relationship can be expressed as: $\mathbf{Y} = f(\mathbf{X_1},\mathbf{X_2},\ldots,\mathbf{X_n})$ where **Y** is the functional requirement and $\mathbf{X_i}$ is the ith entity. n is the number of entities that are related by the equation to the corresponding functional requirement. For example, $\mathbf{V} = f(\mathbf{L},\mathbf{W},\mathbf{H}) = \mathbf{L} \times \mathbf{W} \times \mathbf{H}$, where **V** is the volume, **L** is the length, **W** is the width, and **H** is the height.

In *tolerance analysis*, the entity tolerances, $X_1,X_2,\ldots,X_n$, are given. The goal is to ensure that the functional requirement tolerance, Y, is met. The tolerances, X_i and Y, are the range of acceptable values for, $\mathbf{X_i}$ and **Y**, respectively. Tolerance analysis is essential for product design to determine whether an item can be assembled, or whether it will work correctly. Costly manufacturing problems can be the result of product design without careful tolerance analysis. If the assigned functional requirement tolerances are not met, the tolerances for the entities need to be reassigned by *tolerance synthesis* to achieve the functional requirements.

In tolerance synthesis, the functional requirement tolerance, Y, is given. The goal is to determine a set of feasible entity tolerances, $X_1,X_2,\ldots,X_n$, that fulfill the functional requirement. The task of tolerance synthesis is more difficult because n entity tolerances are determined based on one functional requirement tolerance. In contrast, in tolerance analysis, one functional requirement tolerance is determined based on n entity tolerances. Figure 1 gives the concept and relationship of tolerance analysis and synthesis. We have approached the problem of tolerance design by com-

Reprint requests to: Michael M. Marefat, Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721. E-mail: marefat@ece.arizona.edu

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

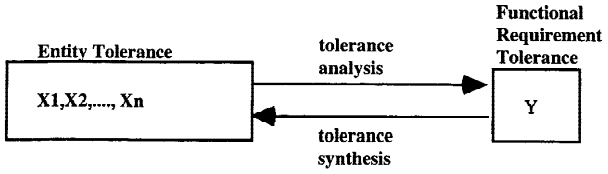


Fig. 1. Relationship of tolerance synthesis and tolerance analysis.

bining interval constraint network and interval propagation techniques. The contributions of this work include:

- New definitions for consistency of constraints and satisfaction of a constraint network for the purpose of tolerance design. The traditional definition of consistency used in interval constraint satisfaction problems (ICSP) cannot be applied for tolerance design.
- Forward and backward propagation algorithms for propagation between adjacent levels in the interval constraint network. Application of these algorithms to tolerance analysis and synthesis are shown.
- Examples and experiments to demonstrate the developed techniques in worst case tolerance synthesis and analysis. Comparison with previous related work is also provided.

This work focuses on the interval constraint networks for representation of tolerances and constraint propagation methods for tolerance maintenance and validation. We do not address the issue of tolerance allocation in this work. Tolerance allocation and cost are issues to be addressed later.

2. BACKGROUND AND RELATED WORK

2.1. Related work in tolerance design

Tolerance design has been the focus of a number of techniques. These techniques include tolerance calculation, worst-case analysis, statistical analysis, design optimization, and constraint-based reasoning. Many of these are restricted to either analysis or synthesis; only a few are applicable to both analysis and synthesis. Most of them approximate a nonlinear relationship between tolerances as a linear relationship for simpler computation and optimization. With this approximation, some of the essential characteristics of the tolerance relationships are often lost.

Tolerance calculations have been explored in several common design situations. For example, Foster (1983) and Spotts (1983) have developed formulas for calculating size position tolerances to achieve a desired class and grade of cylindrical fit between mating parts. However, optimization is not considered in the tolerance calculations. In addition, new formulas are required for each design.

Guilford et al. (1992) and Turner (1987) presented a worst-case tolerance-analysis on an industrial assembly using GEOS (an automated tolerance analysis package developed at the Rensselaer Design Research Center). A worst-case tolerance analysis determines the extreme values of the design function under any possible variation allowed by the tolerances. However, the technique does not synthesize new tolerances. Fortini (1967) presented worst-case tolerance synthesis with a linear design equation in which the limits for the design variables and the tolerance variables are assumed and used to obtain the overall limits on the remaining tolerance variables. Linear design equations are assumed.

Fortini (1967) also presented a statistical tolerance analysis, deriving a probability distribution for the design variable using the probability distributions assumed for the tolerance variables and a linear design equation. Parkinson (1982, 1984, 1985) utilized probabilistic methods to optimize the dimensional tolerance such that there is an acceptably low risk of the assembly failing to meet the specification, given the manufacturing costs. Limit-state equations are used to represent the relationships between the dimensions for assembly. These limit state equations are again replaced by equivalent linear equations.

Hoffmann (1982) reduced the analysis of tolerance and process inaccuracies to the analysis of systems of linear inequalities. The calculation of the resulting tolerances between components of parts is then solved by linear programming. Michael and Siddall (1981, 1982) developed an optimal allocation of manufacturing tolerances by nonlinear optimization, using the minimization of material cost as the optimization criterion. These authors have also developed optimization techniques for a production process with full acceptability (Michael & Siddall, 1981) and with allowance for a system scrap percentage (less than full acceptability) (Michael & Siddall, 1982). Ostwald and Huang (1977) employed linear programming with discrete and 0-1 variables to optimize the design tolerance for each independent basic dimension from several manufacturing processes. Spotts (1973) derived a formulation to calculate the tolerances that will give the assembly of minimum cost. The formulation is derived based on a cost function which varies inversely with the square of the tolerance. Cagan and Kurfess (1992) utilized a simulated annealing approach to allocate tolerances and the manufacturing processes in order to obtain the minimum cost. The cost function in the algorithm is based on the hourly costs of the machines and the production time for the processes. Speckhart (1972) developed an analytical method to allo-

List of Symbols

V	variable (corresponding to entity, attribute, and functional requirement)
V	interval assigned for **V**
v_{up}	upper limit of V
v_{low}	lower limit of V
v_{nom}	nominal value of the variable
v	any value in V

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

cate the optimum set of tolerances that guarantee that the device will meet the statistical fit basis or sure-fit basis and minimize the manufacturing costs. The cost function is an exponential function determined by a nonlinear least-squares curve fit procedure using the cost-tolerance data. Krishnaswami and Mayne (1994) utilized a quality-loss function in addition to the manufacturing-cost function proposed by Speckhart (1972) to optimize the tolerance allocation. The quality-loss function is inversely proportional to the square of the dimension's tolerance. Soderberg (1994) proposed tolerance allocation by optimizing the quality and manufacturing cost. The total loss of customers is computed as the sum of the functionality loss and the component price. These studies have contributed to the implementation of different optimization techniques, such as linear and nonlinear programming, or simulated annealing, and so forth, or to the ability to derive a new objective function based on the manufacturing cost, quality loss, and quality cost. However, the constraints between the tolerances of entities, attributes and functional requirements have not been considered and the satisfaction of these constraints is not guaranteed.

Lu and Wilhelm (1991) and Wilhelm and Lu (1992) proposed a tolerance synthesis approach, CASCADE-T, that used a representation of the conditional tolerance relations that exist between features of a part. Conditional tolerances are automatically determined from functional requirements and shape information. A constraint propagation network is employed for tolerance computation. However, the tolerances are propagated in a random order. This technique may find one solution that satisfies the constraints but is not guaranteed to find a feasible solution. A detailed comparison is provided in Section 5.

2.2. Related work in interval constraints

Constraint satisfaction problems (CSPs) are often formulated in AI tasks. In CSPs, values are assigned to variables subject to a set of constraints. Constraint specification represents the relationships among the variables. A constraint network is a declarative structure that consists of nodes and arcs. The nodes represent the variables or the constraints. The arcs represent the relationship between the variables and the constraints. The variables are labeled by intervals, or sets of possible values. The constraints include any type of mathematical operation or binary relation. The mathematical operations can be multiple inputs single output (MISO) or single input single output (SISO). Constraint propagation is utilized to perform inferences about quantities. For different types of variables and definitions of satisfaction in constraint satisfaction problems, different propagation techniques can be formulated. For tolerance design, the variables are labeled by intervals and the constraints are n-ary mathematical operations.

Dechter and Pearl (1988, 1989) developed a method of generating heuristic advice to guide the order of value assignments based on sparseness in the constraint network and the simplicity of tree-structured CSPs. A backtrack search algorithm is utilized to search for one or all solutions that assign a value to each variable which satisfies all the constraints. Mackworth and Freuder (1985) analyzed the time complexity of several node, arc, and path consistency algorithms in CSPs. However, the domains of the variables considered by Dechter and Pearl (1988, 1989), and Mackworth and Freuder (1985) are discrete, finite sets instead of real intervals.

Ladkin and Reinefeld (1992) developed a technique to solve qualitative interval constraint problems. However, the constraints are binary relations on intervals instead of n-ary mathematical operations on intervals.

Davis's (1987) and Hyvonen's (1992) work is most closely related to ours. The constraints in their interval constraint satisfaction problems (ICSPs) are n-ary mathematical operations and the intervals are real valued intervals. Davis (1987) adapted the Waltz filtering algorithm for screening impossible values from the variable domain to solve the ICSPs. However, the Waltz filtering algorithm cannot determine global solutions in general. Hyvonen (1992) used the tolerance propagation approach, which combines the consistency techniques based on the topology of the constraint net with techniques of interval arithmetic, to solve the ICSPs. Although this method is able to determine global solutions for ICSPs, its definition of consistency and satisfaction of the constraint network are not appropriate in the problem of tolerance synthesis and analysis for mechanical parts. Therefore, these techniques do not solve our problem. These differences in the definitions and applications will be discussed in the later sections.

3. INTERVAL CONSTRAINT NETWORK FOR TOLERANCE DESIGN

3.1. Interval constraint networks and constraint functions

A constraint network is a double, CN($\mathbf{V}$,C) where $\mathbf{V}$ represents the set of variables, $\{\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_n}\}$ and C represents the set of constraints, $\{C_1, C_2, \ldots, C_m\}$. The nodes in the constraint network represent variables or constraints. The arcs in the constraint network represent the relationship between the variables and the constraints.

For each component (we interchangeably refer to a component as a part), its mechanical design can be used to specify the relationships between three categories of items: the component's functional requirements, the component's attributes, and the component's physical entities (for brevity referred to as entities here on). The functional requirement describes the functions of the design and the requirement to satisfy these functions. Each functional requirement can be described as a function (constraint) in terms of attributes. For example, the functional requirement volume of a sphere can be described as a function of the

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

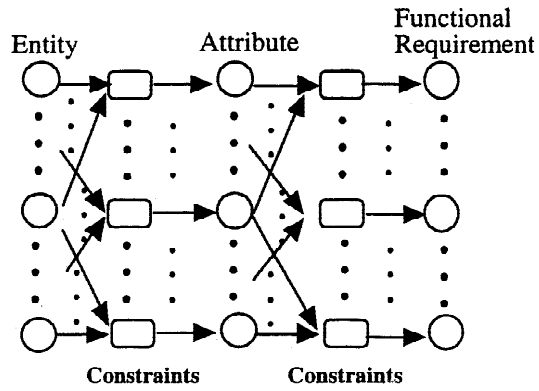


Fig. 2. An interval constraint network representing the constraints between the entities, attributes, and functional requirements of the mechanical design.

attribute, its inner radius. An attribute is also described as a function in terms of the mechanical part's physical entities. Although attributes may not be directly measurable, the physical entities are taken such that they can be measured. In the sphere example, we may not be able to directly measure the attribute inner radius, but the inner radius can be computed as a function of the sphere diameter and the thickness of the raw material. These relationships can be represented as a directed interval constraint network in our approach as shown in Figure 2.

In the directed interval constraint networks for tolerance design, there are three levels of variables, and there are constraint functions between each adjacent pair of levels of variables. The lowest level of variables corresponds to a set of physical entities, the middle level corresponds to a set of attributes, and the highest level corresponds to a set of functional requirements. The constraint functions correspond to the relationships between the corresponding variables. Each function is a MISO or a SISO function.

Each constraint function in the constraint network, $\mathbf{O} = f(\mathbf{I_1}, \mathbf{I_2}, \ldots, \mathbf{I_n})$, describes the relationship between one or more input variables, $\mathbf{I_1}, \mathbf{I_2}, \ldots, \mathbf{I_n}$, and the single output variable, $\mathbf{O}$. This constraint function is used to propagate the exact values of the input variables to the output variables. Based on the constraint functions between the variables and the properties of the interval arithmetic, the interval constraint functions between the corresponding intervals, $O = F(I_1, I_2, \ldots, I_n)$, can be derived.

An interval constraint function is derived based on interval arithmetic. The interval arithmetic is an extension of the real arithmetic, and it deals with closed intervals, X, $[x_{low}, x_{up}]$, which represents $\{x | x_{low} \leq x \leq x_{up}\}$. An interval constraint function, $Z = X \oplus Y$, where $\oplus$ is an arithmetic operation, is defined as:

$$Z = [z_{low}, z_{up}] = X \oplus Y = [x_{low}, x_{up}] \oplus [y_{low}, y_{up}]$$
$$= \{x \oplus y | x_{low} \leq x \leq x_{up}, y_{low} \leq y \leq y_{up}\}$$

For operations, such as addition, subtraction, multiplication, and division, the interval constraint functions can be derived as follows:

Addition:

$$[x_{low}, x_{up}] + [y_{low}, y_{up}] = [x_{low} + y_{low}, x_{up} + y_{up}]$$

Subtraction:

$$[x_{low}, x_{up}] - [y_{low}, y_{up}] = [x_{low} - y_{up}, x_{up} - y_{low}]$$

Multiplication:

$$[x_{low}, x_{up}] \times [y_{low}, y_{up}]$$
$$= [\min\{x_{low}\, y_{low}, x_{low}\, y_{up}, x_{up}\, y_{low}, x_{up}\, y_{up}\},$$
$$\max\{x_{low}\, y_{low}, x_{low}\, y_{up}, x_{up}\, y_{low}, x_{up}\, y_{up}\}]$$

Division:

$$[x_{low}, x_{up}]/[y_{low}, y_{up}] = [x_{low}, x_{up}] \times [1/y_{low}, 1/y_{up}]$$

For $x_{low}, x_{up}, y_{low}, y_{up} \geq 0$, $\min\{x_{low}\, y_{low}, x_{low}\, y_{up}, x_{up}\, y_{low}, x_{up}\, y_{up}\} = x_{low}\, y_{low}$, and $\max\{x_{low}\, y_{low}, x_{low}\, y_{up}, x_{up}\, y_{low}, x_{up}\, y_{up}\} = x_{up}\, y_{up}$. Therefore, $[x_{low}, x_{up}] \times [y_{low}, y_{up}] = [x_{low}\, y_{low}, x_{up}\, y_{up}]$. In our application, the dimensions measured and the functional requirements are always positive; therefore, the above simplification can be used to make presentation and computation easier. Interval constraint functions for other basic functions, such as X^n, $\mathrm{Exp}(X)$, $\mathrm{Sin}(X)$ and so forth, also can be derived easily.

3.2. Satisfaction of interval constraint network

The definition of satisfaction in an interval constraint network must always be appropriate for the application being considered. For any set of problems, a good understanding of the goal of constraint satisfaction for the problem investigated and an appropriate definition of the satisfaction of a constraint network are the foundations of a successful application of constraint networks to the problem.

3.2.1. Satisfaction of interval constraint network for ICSP

According to Hyvonen (1992), the satisfaction of an interval constraint network with n variables $\mathbf{V_1}$ to $\mathbf{V_n}$ and n intervals V_1 to V_n is defined as follows:

> A variable, $\mathbf{V_i}$, is consistent if and only if $\forall(v_i \in V_i | \mathbf{V_i} = v_i)$, $\exists(v_1 \in V_1, \ldots, v_{i-1} \in V_{i-1}, v_{i+1} \in V_{i+1}, \ldots, v_n \in V_n | \mathbf{V_1} = v_1, \ldots, \mathbf{V_{i-1}} = v_{i-1}, \mathbf{V_{i+1}} = v_{i+1}, \ldots, \mathbf{V_n} = v_n$, such that all constraints are satisfied.)

The constraint network is satisfied if and only if all variables are consistent. In other words, given a constraint network with n variables, $\mathbf{V_1}$ to $\mathbf{V_n}$, and a constraint between the variable $\mathbf{V_i}$ and the other variables described by $\mathbf{V_i} = f(\mathbf{V_1}, \ldots, \mathbf{V_n})$, $\mathbf{V_i}$ is consistent if and only if $V_i \subseteq F(V_1, \ldots, V_n)$

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

where F is the interval constraint function with V_i as its output interval and $V_1,\ldots,V_n$ as the input intervals. The purpose of the ICSP as formulated in this way is to refine the intervals of the variables in the constraint network as far as possible without losing possible exact solutions of the constraints. However, in tolerance design, it is desired to have the computed interval using the input intervals and the interval constraint function, $F(V_1,\ldots,V_n)$, to be a subset of the assigned output interval, V_i, for each constraint in the network ($F(V_1,\ldots,V_n) \subseteq V_i$).

The properties of the consistency of a variable as described in ICSP are not appropriate for tolerance design, as can be illustrated by the following example. Let us assume that the constraint function between the area, **A**, and the length and width, **L** and **W**, of a rectangle is $\mathbf{A} = f(\mathbf{L}, \mathbf{W}) = \mathbf{L} \times \mathbf{W}$. Given the tolerances for the length, width, and area respectively, L, W, and A, F(L, W) must be a subset of A so that the designed tolerances for length and width satisfy the expected tolerance of the rectangular area. Given that all the variables are consistent according to the definition of ICSP ($A \subseteq F(L,W)$, $L \subseteq F(A,W)$, $W \subseteq F(A,L)$), the condition $F(L, W) \subseteq A$ is not guaranteed to be satisfied. For instance, given the following tolerances of L, W, and A (L = [3,5], W = [10,12], and A = [30,50]), all the variables, L, W, and A, are consistent. ([30,50] $\subset$ [3,5] $\times$ [10,12] = [30,60], [3,5] $\subset$ [30,50]/[10,12] = [2.5,5], and [10,12] $\subset$ [30,50]/[3,5] = [6,16.7].) However, the requirement of $F(L,W) \subseteq A$ for tolerance design is not satisfied. ([3,5] $\times$ [10,12] = [30,60] $\not\subset$ [30,50].) Although all the variables in the network are consistent, the tolerance assigned is not correct with respect to mechanical design.

3.2.2. Satisfaction of interval constraint networks for tolerance design

Because the purpose of tolerance design in an interval constraint network is different from the purpose of ICSP, a new definition for satisfaction is required. In ICSP, the satisfaction of the network is defined in terms of the consistency of the variables, however, for tolerance design, the constraint consistency should be used. In a constraint network for tolerance design, each constraint is MISO and is represented as a double, $C_i(U,k)$. U is the set of indexes for the input variables and k is the index of the output variable for the constraint C_i. For example, U = {1,2} and k = 3, if the constraint function of C_i is $\mathbf{V_3} = f(\mathbf{V_1}, \mathbf{V_2})$.

The definitions of consistency of a constraint and satisfaction of an interval constraint network for tolerance design are as follows:

DEFINITION 1. A constraint, C_i, is consistent if and only if $\bigcap_{j\in U}(\forall v_j \in V_j | \mathbf{V_j} = v_j)$, $(\exists v_k \in V_k | \mathbf{V_k} = v_k)$ such that C_i is satisfied, where U is the set of indexes for the input variables and k is the index of the output variable for the constraint C_i. ■

DEFINITION 2. The interval constraint network for tolerance design is satisfied if and only if all the constraints are consistent. ■

Based on the definition of consistency of constraints and satisfaction of the network, the tolerances assigned to the entities are ensured to satisfy the tolerances of the functional requirements. The interval computed from the input intervals and the interval constraint function is expected to be a subset of the assigned output interval for each constraint in the network. Taking the earlier example for ICSP, with A = [30,50], L and W could be refined to some other intervals, such as L = [3,4.5] and W = [10,11], such that the constraint is consistent. After refining L and W, we could take any values, l and w, in intervals L and W, and there is always a value, a, in A such that $l \times w = a$.

4. PROPAGATION

These new definitions of consistency and satisfaction for tolerance design require new tolerance propagation techniques. With such techniques, tolerances can be propagated from variable to variable in the constraint network to ensure that the constraints are consistent.

4.1. Forward propagation for a single constraint

The forward propagation is based on the constraint function such that the intervals of the input variables are propagated to the interval of the single output variable. If the interval propagated from the input intervals is not a subset of the output interval, the output interval is updated (relaxed) to the union of the propagated interval and the original assigned output interval, otherwise, the constraint is consistent and nothing is changed. The algorithm for forward propagation is given as follows:

Forward Propagation for constraint, $C(\{1,2,\ldots,n\}, k)$, $FP(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n}; \mathbf{Xk})$

Propagated from Input Tolerance to the Upper Limit of the Output Tolerance

$x'_{k_{up}} = f(x_{1_\varphi},\ldots,x_{n_\varphi})$

where $x_{i_\varphi} = x_{i_{up}}$ if $\mathbf{X_k}$ is monotonically increasing with respect to $\mathbf{X_i}$.

$x_{i_\varphi} = x_{i_{low}}$ if $\mathbf{X_k}$ is monotonically decreasing with respect to $\mathbf{X_i}$.

Propagated from Input Tolerance to the Lower Limit of the Output Tolerance

$x'_{k_{low}} = f(x_{1_\kappa},\ldots,x_{n_\kappa})$

where $x_{i_\kappa} = x_{i_{low}}$ if $\mathbf{X_k}$ is monotonically increasing with respect to $\mathbf{X_i}$.

$x_{i_\kappa} = x_{i_{up}}$ if $\mathbf{X_k}$ is monotonically decreasing with respect to $\mathbf{X_i}$.

Relaxing the Output Tolerance

If $x'_{k_{up}} < x_{k_{low}}$ or $x'_{k_{low}} > x_{k_{up}}$,

NO SOLUTION

Otherwise,

$x_{k_{up}} = x'_{k_{up}}$ if $x'_{k_{up}} > x_{k_{up}}$.

$x_{k_{low}} = x'_{k_{low}}$ if $x'_{k_{low}} < x_{k_{low}}$.

Figure 3(a) shows the interval constraint network for a single constraint with input intervals, $X_1, X_2, \ldots, X_n$, and out-

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

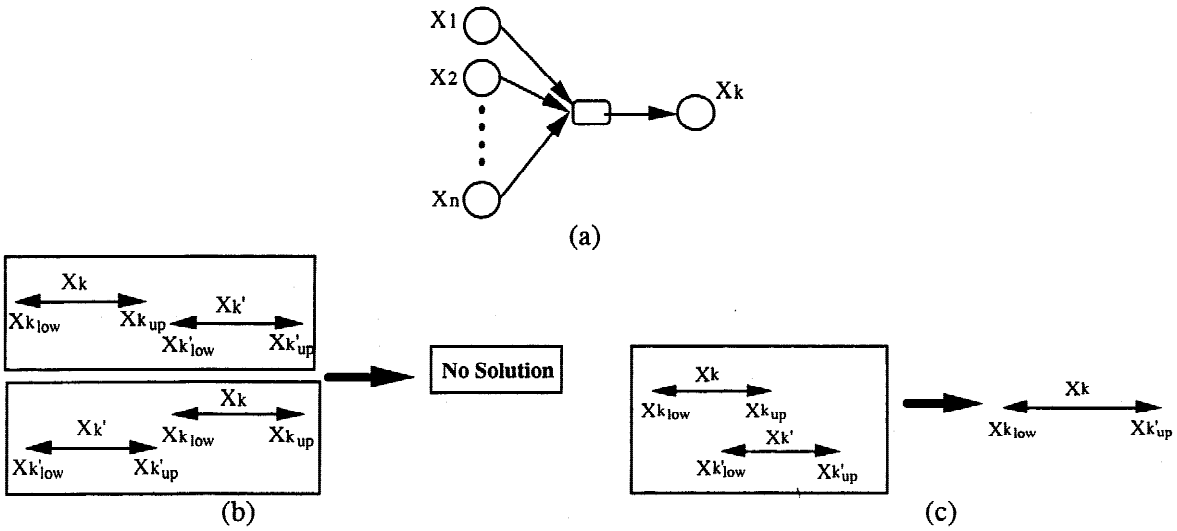


Fig. 3. (a) An interval constraint network with single constraint. (b) Two situations where there is no solution of the constraint for forward propagation. (c) An example where the output interval is updated by the union of the propagated interval and the original output interval for forward propagation.

put interval, X_k. If the propagated interval, X'_k, does not intersect with the output interval, X_k, (i.e., $X'_k \cap X_k = \varnothing$), there is no solution for the constraint. Figure 3(b) illustrates two situations when there is no solution. If $x'_{k_{up}} < x_{k_{low}}$ or $x'_{k_{low}} > x_{k_{up}}$, there is no feasible solution to make the constraint consistent. However, if X'_k and X_k intersect, the output interval is updated to the union of X'_k and X_k, (i.e., $X'_k \cup X_k$). Figure 3(c) shows an example of the output interval being updated by the union of the propagated interval and the original output interval. An example to illustrate the forward propagation is given as follows:

EXAMPLE. In Figure 3, given $n = 3$, $k = 4$, $X_1 = [1,5]$, $X_2 = [8,10]$, $X_3 = [50,55]$, $X_4 = [71,75]$, and $\mathbf{X_1 + X_2 + X_3 = X_4}$, the constraint is not consistent because $X_1 + X_2 + X_3 \not\subset X_4$. Therefore, a forward propagation is processed to relax X_4. First, propagated values for the upper and lower limits of the output interval of the constraint are calculated based on intervals for input variables. $x'_{4_{up}} = 5 + 10 + 55 = 70$ and $x'_{4_{low}} = 1 + 8 + 50 = 59$. Since $x_{4_{low}} = 71 \geq x'_{4_{up}}$, therefore, no solution is obtained. If X_4 is originally $[65,72]$ instead of $[71,75]$, X_4 can be updated to $[59,70] \cup [65,72] = [59,72]$. ■

4.2. Backward propagation for a single constraint

The backward propagation is also based on the constraint function such that the interval of the output variable is propagated to one or more of the intervals of the input variables. If the constraint is not consistent, the output interval is propagated to the input intervals by tightening each of the input intervals. There are several options for tightening the input intervals: (1) tightening uniformly on every interval, (2) tightening the intervals proportional to their corresponding nominal values of the variables, (3) tightening the intervals proportional to the width of the intervals. The algorithm for backward propagation is given as follows:

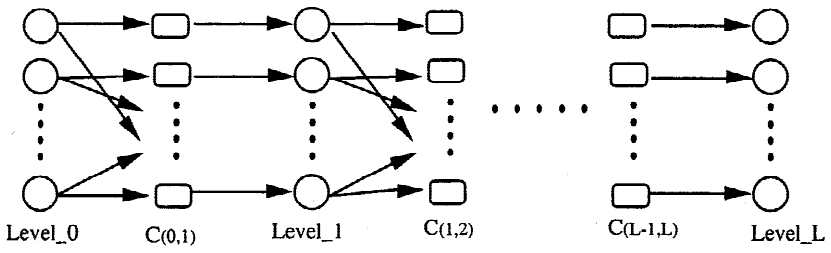


Fig. 4. An interval constraint network with L levels, Level_0 to Level_L. Between each adjacent pairs of levels, Level_i and Level_i+1, there is a set of constraints representing the relationship between the intervals in Level_i and Level_i+1, C(i, i+1).

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

Backward Propagation for constraint, C({1,2,…,n}, k), BP($\mathbf{X_k}$; $\mathbf{X_1}$,$\mathbf{X_2}$,…,$\mathbf{X_n}$)

Compute the Propagation Parameter, α_i and β_i

If the changes on X_1 to X_n are *uniform*,

$\alpha_i = \beta_i = 1$ for $1 \le i \le n$.

If the changes on X_1 to X_n depend on the *nominal* value of $\mathbf{X_1}$ to $\mathbf{X_n}$,

$\alpha_i = \beta_i = x_{inom}$ for $1 \le i \le n$.

If the changes on X_1 to X_n depend on the *width* of the intervals,

$\alpha_i = x_{i\varphi} - x_{inom}$ if $x_{i\varphi} = x_{iup}$

$\alpha_i = x_{inom} - x_{i\varphi}$ if $x_{i\varphi} = x_{ilow}$

and $\beta_i = x_{i\kappa} - x_{inom}$ if $x_{i\kappa} = x_{iup}$

$\beta_i = x_{inom} - x_{i\kappa}$ if $x_{i\kappa} = x_{ilow}$ for $1 \le i \le n$.

where $x_{i\varphi} = x_{iup}$ if $\mathbf{X_k}$ is monotonically increasing with respect to $\mathbf{X_i}$.

$x_{i\varphi} = x_{ilow}$ if $\mathbf{X_k}$ is monotonically decreasing with respect to $\mathbf{X_i}$.

$x_{i\kappa} = x_{ilow}$ if $\mathbf{X_k}$ is monotonically increasing with respect to $\mathbf{X_i}$.

$x_{i\kappa} = x_{iup}$ if $\mathbf{X_k}$ is monotonically decreasing with respect to $\mathbf{X_i}$.

Solve the Upper Limit Parameter, τ_{up}, based on the Constraint Function

Solve $x_{kup} = f(x_{1\varphi} + \Delta x_{1\varphi},\ldots,x_{n\varphi} + \Delta x_{n\varphi})$ for τ_{up}

where $\Delta x_{i\varphi} = \alpha_i \tau_{up}$ if $\mathbf{X_k}$ is monotonically increasing with respect to $\mathbf{X_i}$.

$\Delta x_{i\varphi} = -\alpha_i \tau_{up}$ if $\mathbf{X_k}$ is monotonically decreasing with respect to $\mathbf{X_i}$.

If $\tau_{up} > 0$, $\tau_{up} = 0$.

Solve the Lower Limit Parameter, τ_{low}, based on the Constraint Function

Solve $x_{klow} = f(x_{1\kappa} + \Delta x_{1\kappa},\ldots,x_{n\kappa} + \Delta x_{n\kappa})$ for τ_{low}

where $\Delta x_{i\kappa} = \beta_i \tau_{low}$ if $\mathbf{X_k}$ is monotonically increasing with respect to $\mathbf{X_i}$.

$\Delta x_{i\kappa} = -\beta_i \tau_{low}$ if $\mathbf{X_k}$ is monotonically decreasing with respect to $\mathbf{X_i}$.

If $\tau_{low} < 0$, $\tau_{low} = 0$.

Update the Input Intervals

$x'_{i\varphi} = x_{i\varphi} + \alpha_i \tau_{up}$ for $1 \le i \le n$,

$x'_{i\kappa} = x_{i\kappa} + \beta_i \tau_{low}$ for $1 \le i \le n$.

EXAMPLE. In Figure 3, given n = 3, k = 4, $X_1 = [1,5]$, $X_2 = [8,10]$, $X_3 = [50,55]$, $X_4 = [62,68]$, and $\mathbf{X_1} + \mathbf{X_2} + \mathbf{X_3} = \mathbf{X_4}$, the constraint is not consistent because $X_1 + X_2 + X_3 \not\subset X_4$. A backward propagation needs to be processed to tighten X_1, X_2, and X_3. First, the propagation parameters are computed. Assuming uniform allocation, $\alpha_i = \beta_i = 1$ for i = 1, 2, and 3, the upper limit parameter and the lower limit parameter, τ_{up} and τ_{low}, are solved based on the following equations:

$$68 = 5 + \tau_{up} + 10 + \tau_{up} + 55 + \tau_{up}$$

$$62 = 1 + \tau_{low} + 8 + \tau_{low}, + 50 + \tau_{low},$$

$\tau_{up} = -0.67$ and $\tau_{low} = 1.00$. Subsequently, X_1, X_2, and X_3 are updated to [2.00,4.33], [9.00,9.33], and [51.00,54.33] where $x'_{iup} = x_{iup} + \tau_{up}$ and $x'_{ilow} = x_{ilow} + \tau_{low}$ for i = 1, 2, and 3. ■

Backward propagation provides three alternative ways to propagate the output interval to the input intervals, based on the characteristics of the input intervals. The first alternative is to allocate the propagation to the input intervals uniformly. The second and third alternatives allocate the propagation proportionally based on the nominal value and the width of the input intervals, respectively. The choice of alternative depends on the purpose of the propagation and the properties of the alternatives (capability to find the feasible solution). For example, if the absolute amount of tightening on each input interval is desired to be identical, the first alternative will be chosen. If the nominal value of the variable has significant meaning for the propagated intervals, the second alternative will be chosen. Only the third alternative guarantees finding a feasible solution if a feasible solution exists.

A feasible solution in the backward propagation for a constraint, C, should have the following two properties:

1. There are subsets of the input intervals such that C is satisfied.
2. For each input interval, the subset, $[x'_{ilow}, x'_{iup}]$ that satisfies C should include the nominal value, (i.e., $x'_{ilow} \le x_{inom} \le x'_{iup}$).

A good backward propagation technique should be able to find a feasible solution if one exists.

Although the backward propagation technique provided in this section can always find a feasible solution when one exists, the correct assignment of the propagation parameters, α_i and β_i must be chosen. An assignment that is able to find a feasible solution is described in Theorem 1 and assignments that are unable to find the feasible solution are described in Corollary 1. The proofs of these theorems are provided in the Appendix.

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

THEOREM 1. *Given the interval of the output variable, if there is a set of intervals for the input variables that includes the variables' nominal values and satisfies the constraint, the backward propagation technique with the assignment of α_i and β_i based on the width of the intervals can always find a feasible set of intervals.* ■

COROLLARY 1. *Given the interval of the output variable, if there is a set of intervals for the input variables that includes the variables' nominal values and satisfies the constraint, the backward propagation technique with the assignment of α_i and β_i based on the uniform distribution and the nominal value of the input variables is not guaranteed to find the feasible set of intervals.* ■

4.3. Parallel propagation

In Sections 3.1. and 3.2., the propagation technique through a single constraint has been introduced. However, an interval constraint network usually consists of more than one constraint. In a directed interval constraint network for tolerance design, there are three levels of variables and there is a set of constraints between each pair of adjacent levels of variables. In parallel propagation (see Fig. 4), all the tolerances in one level, Level_i, are propagated to all the tolerances in a lower level (or a higher level), Level_i−1 (or Level_i+1), through all the constraints between these two levels by backward propagation (or forward propagation) simultaneously. In parallel propagation, before the propagated tolerances, in Level_i−1 (or Level_i+1) are propagated further to another level, Level_i−2 (or Level_i+2), all the tolerances in Level_i−1 (or Level_i+1) have to be updated. Thus in tolerance analysis, all the entities' tolerances are propagated to the attributes' tolerances before all the attributes' tolerances can be propagated to the functional requirements' tolerances. A similar situation holds for tolerance synthesis. All functional requirements' tolerances are propagated to the attributes' tolerances and then propagated to the entities' tolerances. In some methods the order of propagation may have an impact on the final solution obtained. In this work, parallel propagation in the interval constraint network achieves the same solution regardless of the order of propagation.

The algorithms for parallel forward and backward propagation are shown below:

Parallel Forward Propagation from Level_i to Level_i+1.

For each constraint in $L_{(i,\,i+1)}$

Propagate the input intervals to the output intervals simultaneously using the technique in Section 3.1

Parallel Backward Propagation from Level_i+1 to Level_i.

1. **Propagate to the intervals in Level_i for which out-degree is greater than one**

   A. Initialize: Let M^I be the set of variable intervals in Level_i, which are constrained by more than one unsatisfied constraint between Level_i and Level_i+1, and let M^C be the set of unsatisfied constraints each of which has more than one input interval among the intervals in the set M^I:

   For each interval $X_j \in M^I$, let $X_j^{\min} = X_j$,

   where X_j is the currently assigned interval for the variable before backpropagation.

   B. Propagation: For each unsatisfied constraint $C^u \in M^C$, propagate its output interval in Level_i+1 to the its input intervals in M^I, X_1^u, X_2^u, …, X_n^u, using the backward propagation algorithm $BP(\mathbf{X_k}; \mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n})$.

   C. Tightening: For each interval $X_j \in M^I$, update the interval using

   $X_j = X_j^{\min} \cap X_j^{u^1} \cap X_j^{u^2} \cap \ldots \cap X_j^{u^q}$, if updated interval $X_j^{\min} \cap X_j^{u^1} \cap X_j^{u^2} \cap \ldots \cap X_j^{u^q} \neq \varnothing$

   where $X_j^{u^1}, X_j^{u^2}, \ldots X_j^{u^q}$ are the intervals obtained in step B from the propagation of different constraints that use X_j as input.

   D. Iteratively Relax: While all constraints in M^C are satisfied:

   For each variable with an interval in M^I:

   Relax X_j by a small amount using the α_j and β_j values used in the backward propagation to X_j in Step 2.

2. **Propagate to the intervals in Level_i for which the out-degree is equal to one**

   For each constraint in L(i,i+1)

   Propagate the output intervals in Level_i+1 to the intervals in Level_i, which are not elements of M^I, with α_j and $\beta_j = 0$ for each interval $X_j \in M^I$, using the technique in Section 4.2.

In parallel forward propagation, all the intervals in Level_i are propagated to all the intervals in Level_i+1 simultaneously. Because all the constraints between the two levels are MISO, the updated intervals in Level_i+1 do not affect one another.

The above two-step parallel backward propagation is developed to ensure the consistency of all the constraints between the two levels of variables. In parallel backward propagation, each of the updated intervals in Level_i may be propagated from several intervals in Level_i+1 through more than one constraint. Step 1 in the propagation is for the intervals in Level_i that are constrained by more than one constraint, (i.e., the corresponding nodes in the network in which the out-degree is larger than one), and the second propagation step is for the rest of the intervals in Level_i. In Step 1, the intervals in Level_i+1 are propagated to those intervals in Level_i that are constrained by more than one constraint, and the tightest constraint from all the output intervals on these intervals is found and saved. In part 1.D, any overtightening resulting from the interactions between constraints is relaxed by small steps to ensure that all feasible solutions are retained. This relaxing can be necessary, when the intersections in part 1.B. result in some of the variable intervals in M^I being constrained more than necessary because of propagation from different

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

constraints in which the variable participates. In Step 2, the rest of the intervals in Level_i are updated without changing the intervals that have already been computed in Step 1.

If a feasible solution does not exist after backward propagation, it is because of a serious flaw between the original intervals and the design represented by the network. For example, if in backward propagation all of the input intervals were reduced to their nominal values, this should produce a feasible solution in a well-conceived design. Backward propagation that fails identifies the specific location of the design flaw.

4.3.1. Example

Figure 5 shows an example of a partial interval constraint network. The intervals, X_1, X_2, X_3, and X_4, are propagated simultaneously, to the intervals, Y_1, Y_2, and Y_3, through the constraints, C_1, C_2, and C_3. Therefore, three forward propagations, $FP(X_1; Y_1)$, $FP(X_1,X_2,X_3; Y_2)$, and $FP(X_3,X_4; Y_3)$ are processed simultaneously to update Y_1, Y_2, and Y_3.

Given $X_1 = [5,10]$, $X_2 = [20,25]$, $X_3 = [15,18]$, $X_4 = [9,10]$, $Y_1 = [1,3]$, $Y_2 = [40,50]$, $Y_3 = [140,180]$, $x_{1_{nom}} = 7$, $x_{2_{nom}} = 22$, $x_{3_{nom}} = 17$, $x_{4_{nom}} = 9.5$, assume the constraints are $Ln(\mathbf{X_1}) = \mathbf{Y_1}$, $\mathbf{X_1} + \mathbf{X_2} + \mathbf{X_3} = \mathbf{Y_2}$, and $\mathbf{X_3} \times \mathbf{X_4} = \mathbf{Y_3}$. Since $Ln(X_1) = [1.61,2.30] \subset Y_1$, the constraint C_1 is consistent and relaxing Y_1 is not necessary. As a result, only $FP(X_1,X_2,X_3; Y_2)$, and $FP(X_3,X_4; Y_3)$ are processed simultaneously. Y_2 and Y_3 are then updated to [40,53] and [135,180], respectively. Now, all the constraints between X_i and Y_j are consistent where i = 1, 2, 3, 4, and j = 1, 2, 3.

For parallel backward propagation, Y_1, Y_2, and Y_3 are first propagated through C_1, C_2, and C_3 to X_1 and X_3. ($BP(Y_1; X_1)$, $BP(Y_2; X_1, X_2, X_3)$ and $BP(Y_3; X_3, X_4)$ are used.) Then, Y_2 and Y_3 are propagated through C_2 and C_3 to X_2 and X_4 with X_1 and X_3 fixed as the intervals propagated the first time. (Only $BP(Y_2; X_1, X_2, X_3)$ and $BP(Y_3; X_3, X_4)$ are used.)

In this example, only the original Y_2 and Y_3 are propagated backward simultaneously to X_1, X_2, X_3, and X_4 in parallel backward propagation because C_1 is found consistent in forward propagations. Only $BP(Y_2; X_1, X_2, X_3)$ and $BP(Y_3; X_3, X_4)$ are processed to update X_3 in the first step of parallel backward propagation. However, X_1 is not updated because X_1 is only constrained by C_2 when C_1 is consistent. According to Theorem 1, allocation based on the width of intervals is used in this example so that a feasible solution is guaranteed to be found. $BP(Y_2; X_1, X_2, X_3)$ tightens the upper limit of X_3 to 17.571 and $BP(Y_3; X_3, X_4)$ tightens the lower limit of X_3 to 15.195. X_3 is updated to [15.389,17.572]. Fixing X_3 to the updated interval, $BP(Y_2; X_1, X_2, X_3)$ and $BP(Y_3; X_3, X_4)$ are processed again to update X_1, X_2, and X_4. X_1 and X_2 are updated to [5.000,8.714] and [20.000,23.714], respectively. X_4 is also updated to [9.097,10]. All the constraints between X_i and Y_j are now consistent after parallel backward propagation where i = 1, 2, 3, 4, and j = 1, 2, 3.

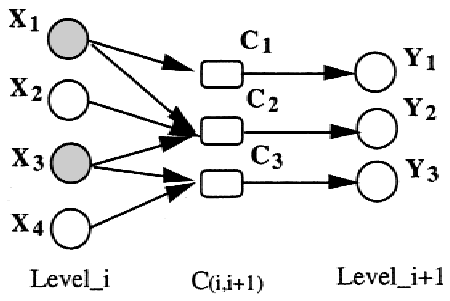


Fig. 5. An example of a partial interval constraint network from Level_i to Level_i+1, where Level_i has four variables, X_1, X_2, X_3, and X_4, Level_i+1 has three variables, Y_1, Y_2, and Y_3, and there are three constraints in C(i,i+1), C_1, C_2, and C_3.

5. TOLERANCE DESIGN

In this section, the tank in Figure 6 is utilized as an example to illustrate tolerance analysis and tolerance synthesis using an interval constraint network. The tank is made up of two cylinders. The functional requirements are the total volume of the tank, **V**, and thicknesses of the tank, **T1**, **T2**, and **T3**, as labeled in Figure 6(b). The attributes of the cylinders are the inner radius and the outer radius, **R1**, **R2**, **R3**, and **R4**, the inner lengths, **L1** and **L2**, and the outer length of the smaller cylinder, **L3** (Figure 6(c)). The entities are several measurable tank lengths, as labeled in Figure 6(d).

The interval constraint network for the tank is given in Figure 7. The highest level has four variables, **V**, **T1**, **T2**, and **T3**. The middle level has seven variables. **L1**, **L2**, **L3**, **R1**, **R2**, **R3**, and **R4**, The lowest level has seven variables, **E1**, **E2**, **E3**, **E4**, **E5**, **E6**, and **E7**. Between the highest level and the middle level, there are four constraints; between the middle level and the lowest level, there are seven constraints. The constraint functions are given in Table. 1.

5.1. Tolerance analysis

After tolerances have been assigned to the entities of the parts, tolerance analysis ensures that the tolerances for the functional requirements are satisfied. Therefore, tolerances are propagated from all the entities to the attributes and then propagated to the tolerances of the functional requirements. In many situations, the tolerances of the attributes are not assigned, so the upper limits and lower limits of these tolerances are initially set to the nominal value in the constraint network.

In the tank example, the tolerances of the entities, **E1**, **E2**, **E3**, **E4**, **E5**, **E6**, and **E7**, are propagated to the tolerances of the functional requirements, **V**, **T1**, **T2**, and **T3**. The nominal values of the entities, **E1**, **E2**, **E3**, **E4**, **E5**, **E6**, and **E7**, are 95 mm, 200 mm, 100 mm, 50 mm, 50 mm, 190 mm, and 200 mm, respectively. The nominal values of the attributes, **L1**, **L2**, **L3**, **R1**, **R2**, **R3**, and **R4**, are 100 mm,

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

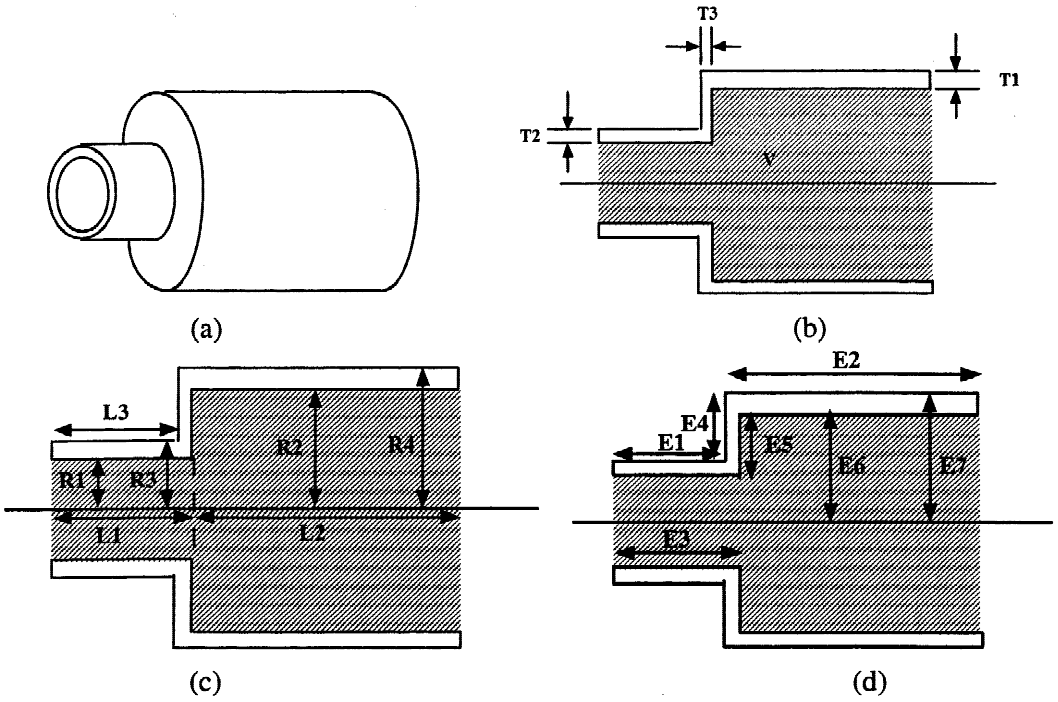


Fig. 6. (a) A tank, (b) the labels of functional requirement, (c) the labels of attributes of the cylinders, and (d) the labels of the measurable entities.

200 mm, 95 mm, 140 mm, 190 mm, 150 mm, and 200 mm, respectively. The nominal values of the functional requirements, **V**, **T1**, **T2**, and **T3**, are 2.9×10^7 mm^3, 10 mm, 10 mm, and 5 mm, respectively. The initial tolerances of the entities, attributes, and functional requirements are given in Table 2. The upper limits and the lower limits of all the attributes are set to their nominal values. The tolerances of the entities are first propagated to the attributes' tolerances through the corresponding interval constraint functions in the first column of Table 1; the results are given in Table 2. The tolerances of the attributes are then propagated to the functional requirements' tolerances through the corresponding interval constraint functions in the second column of Table 2. The results are also given in Table 2. The propagated tolerances of the functional requirements are [2.8241×10^7 mm^3, 2.9419x10^7 mm^3], [8 mm,12 mm], [6 mm,14 mm], and [3 mm,7 mm] for **V**, **T1**, **T2**, and **T3**, respectively. As a result of this analysis, we see that the propagated tolerances of **T1**, **T2**, and **T3** do not satisfy the functional requirements because they are not subsets of

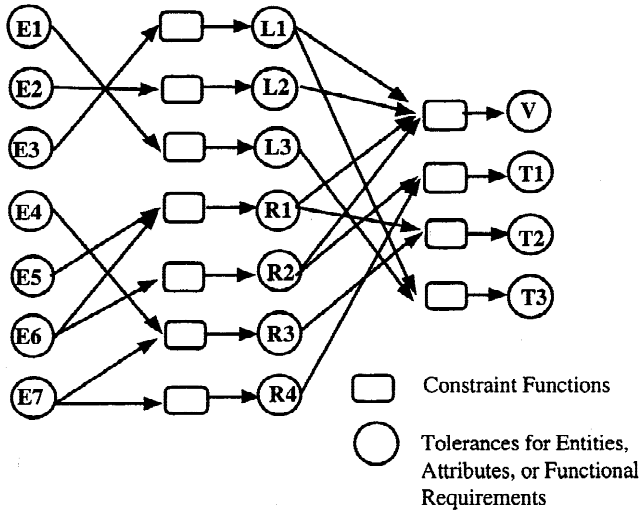


Fig. 7. The interval constraint network representing the relationship of the tolerance of entities, attributes, and functional requirements for the tank in Figure 6.

Table 1. *Constraint functions for the interval constraint network in Figure 7*

Constraint Functions between Entities and Attributes	Constraint Functions between Attributes and Functional Requirements.
L1 = E3	**V = πR1 R1 L1 + π R2 R2 L2**
L2 = E1 + E2 − E3	**T1 = R4 − R2**
L3 = E1	**T2 = R3 − R1**
R1 = E6 − E5	**T3 = L1 − L3**
R2 = E6	
R3 = E7 − E4	
R4 = E7	

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

Table 2. *The tolerances of the entities, attributes, and functional requirements initially, after tolerance analysis, and after tolerance synthesis*

		Initial	Tolerance Analysis	Tolerance Synthesis
Entities	E1	[94 mm, 96 mm]	[94 mm, 96 mm]	[94.75 mm, 95.25 mm]
	E2	[204 mm, 206 mm]	[204 mm, 206 mm]	[204 mm, 206 mm]
	E3	[99 mm, 101 mm]	[99 mm, 101 mm]	[99.75 mm, 100.25 mm]
	E4	[49 mm, 51 mm]	[49 mm, 51 mm]	[49.75 mm, 50.25 mm]
	E5	[49 mm, 51 mm]	[49 mm, 51 mm]	[49.75 mm, 50.25 mm]
	E6	[189 mm, 191 mm]	[189 mm, 191 mm]	[189.75 mm, 190.25 mm]
	E7	[199 mm, 201 mm]	[199 mm, 201 mm]	[199.75 mm, 200.25 mm]
Attributes	L1	[100 mm, 100 mm]	[99 mm, 101 mm]	[99.75 mm, 100.25 mm]
	L2	[200 mm, 200 mm]	[197 mm, 203 mm]	[197 mm, 203 mm]
	L3	[95 mm, 95 mm]	[94 mm, 96 mm]	[94.75 mm, 95.25 mm]
	R1	[140 mm, 140 mm]	[138 mm, 142 mm]	[139.5 mm, 140.5 mm]
	R2	[190 mm, 190 mm]	[189 mm, 191 mm]	[189.5 mm, 190.5 mm]
	R3	[150 mm, 150 mm]	[148 mm, 152 mm]	[149.5 mm, 150.5 mm]
	R4	[200 mm, 200 mm]	[199 mm, 201 mm]	[199.5 mm, 200.5 mm]
Functional Requirements	V	$[2.8 \times 10^7\ \text{mm}^3, 3.0 \times 10^7\ \text{mm}^3]$	$[2.8 \times 10^7\ \text{mm}^3, 3.0 \times 10^7\ \text{mm}^3]$	$[2.8 \times 10^7\ \text{mm}^3, 3.0 \times 10^7\ \text{mm}^3]$
	T1	[9 mm, 11 mm]	[8 mm, 12 mm]	[9 mm, 11 mm]
	T2	[9 mm, 11 mm]	[6 mm, 14 mm]	[9 mm, 11 mm]
	T3	[4.5 mm, 5.5 mm]	[3 mm, 7 mm]	[4.5 mm, 5.5 mm]

the designed tolerance as shown in Column 3 of Table 2. However, the propagated tolerances for **V** do satisfy that assignment.

5.2. Tolerance synthesis

In tolerance synthesis, the tolerances of functional requirements that do not satisfy the assignments in tolerance analysis are propagated backward to obtain new necessary tolerances for the entities. If all the functional requirements' tolerances are satisfied, tolerance synthesis is not necessary. As a result, not every node in the network will be visited during the backward propagation, only those that are related to the unsatisfied functional requirements. To ensure finding a feasible solution in this example, the assignments of α_i and β_i are based on the width of the tolerances. After tolerance synthesis, a new set of tolerances for the entities that will satisfy all the constraints in the network is generated.

In the tank example, only the tolerances for **T1**, **T2**, and **T3** are propagated backward to the attributes' tolerances **L1**, **L3**, **R1**, **R2**, **R3**, and **R4**. **V**'s tolerance is not propagated to any of the attributes' tolerances because V's tolerance is satisfied as described in tolerance analysis in Section 4.2. **L2**'s tolerance is not propagated from **T1**, **T2**, and **T3** because **L2** is not related to any of **T1**, **T2**, and **T3**. The propagated tolerances for the attributes are given in Table 2. Because none of **L1**, **L3**, **R1**, **R2**, **R3**, and **R4** are constrained by more than one of **T1**, **T2**, and **T3**, only the second step in the parallel backward propagation method is needed.

After the attributes' tolerances are updated, they are propagated to the entities' tolerances. In this example, the tolerances of **L1**, **L3**, **R1**, **R2**, **R3**, and **R4**, are propagated to **E1**, **E3**, **E4**, **E5**, **E6**, and **E7**. **E2** is not propagated because it is constrained only by **L2** and **L2** is not changed during the process of tolerance synthesis. In this propagation, **E6** and **E7** are constrained by more than one attribute. **E6** is constrained by **R1** and **R2**. **E7** is constrained by **R3** and **R4**. However, **E1**, **E3**, **E4**, and **E5** are constrained by only one attribute. Therefore, a parallel backward propagation is first processed to propagate the tolerances from **R1**, **R2**, **R3**, and **R4** to **E6** and **E7**. A second parallel backward propagation is then processed to propagate from **L1**, **L3**, **R1**, and **R3** to **E1**, **E3**, **E4**, and **E5** with α_{E6}, α_{E7}, β_{E6}, β_{E7} equal to 0.

6. DISCUSSION AND COMPARISON

Davis (1987) and Hyvonen (1992) investigated the ICSP, where real interval and n-ary mathematical operations are used, but, as discussed in Section 2, the proposed definitions for satisfaction of the network are not appropriate for tolerance satisfaction. In ICSP, the purpose is to refine the intervals of the variables as far as possible without losing possible exact solutions of the constraints. For all values in the interval of a variable, there are always some values in each of the other intervals that satisfy all the constraints. However, in tolerance design, the purpose is to ensure that the propagated interval from the input variables is a subset of the output variable's interval for each constraint.

Lu and Wilhelm (1991) and Wilhelm and Lu (1992) have utilized constraint networks for tolerance synthesis in a system named CASCADE-T. Their work is the most closely related work; however, our work takes a different approach from techniques developed in their research. Examples in

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

the following section demonstrate and compare the two approaches. CASCADE-T is different from the work presented here in the following ways:

- The sequences and direction of the tolerance propagation are not specified in CASCADE-T. A user can select the propagation paths subjectively. Nonunique solutions may be obtained as a result of constraint propagation through different propagation paths.
- One is not guaranteed to obtain a solution in CASCADE-T even when a feasible solution to the constraint network exists. The procedures presented here always find a feasible solution when one exists.
- The solution obtained in CASCADE-T may not satisfy the notion of constraint consistency as defined in Definition 1 (Section 2.2.2.).

An example of two prisms is provided in Wilhelm and Lu (1992). This example, shown in Figure 8(a), can be utilized to compare CASCADE-T and the approach presented here. CASCADE-T's constraint network for this example is shown in Figure 8(b). The constraint network constructed according to the method presented here for the same example is shown in Figure 8(c). The structures of the constraint networks in Figures 8(b) and 8(c) are significantly different. In the network shown in Figure 8(b), the entities of the prisms are not considered. The constraint representing the equivalence between W1 and W2 is represented in the network, but the equivalence between L1 and L2 is not required. On the other hand, in our network (Fig. 8(c)), all the entities and the attributes are considered. Although the equivalencies of W1 and W2 and L1 and L2 are not represented in the network, these equivalent constraints are already implied by the constraints, C5 and C6, C7 and C8, C9 and C10, and C11 and C12.

Not only the networks are different; the techniques of tolerance synthesis and the results are also different. Initially, the tolerances of W1, H2, and W2 are not available in this example in Wilhelm and Lu (1992). As a result, there is only one possible path of propagation, as shown in Figure 9(a) and the results given in the third column of Table 3. Indeed, the unavailability of tolerances for a variable is the equivalent of unlimited tolerance, which makes the problem easier to solve. Before applying our parallel propagation as given in Section 3.3, the tolerances of W1 and W2 and H2 are first assigned based on the nominal values of $W1_{nom}$, $W2_{nom}$, and $H2_{nom}$ (assumed to be the mid-value of the tolerance). The results are given in the fifth column of Table 3. Unlike the results in Wilhelm and Lu (1992), the tolerances for all the entities are obtained. Moreover, the wide ranges in their results for W1, W2, and H2 do not occur in our results.

The results in Table 3 also show that constraint consistency according to Definition 1 is not satisfied in Wilhelm and Lu (1992) because some propagations in CASCADE-T's technique are based on the solution function instead of the constraint function. The solution function is the function derived from the constraint function such that an input variable is represented as a function of the output variable and the other input variables of the constraint function. For example, given the constraint function $\mathbf{C} = \mathbf{A} + \mathbf{B}$, the solution functions are $\mathbf{A} = \mathbf{C} - \mathbf{B}$ and $\mathbf{B} = \mathbf{C} - \mathbf{A}$. In the example of two prisms, W1 is propagated from A1 and V1 through the solution function $\mathbf{W1} = \mathbf{V1}/\mathbf{A1}$ such that the condition $V1/A1 \subseteq W1$ is satisfied using CASCADE-T's technique. However, $V1/A1 \subseteq W1$ is not the same as $W1 \times A1 \subseteq V1$, which is the required condition for the consistency of the constraint $\mathbf{V1} = \mathbf{W1} \times \mathbf{A1}$. As a result, constraint consistency is not guaranteed. In the example of $\mathbf{V1} = \mathbf{W1} \times \mathbf{A1}$, W1 and A1 are [2.5,50] and [1,8], respectively, after CASCADE-T's propagation. $W1 \times A1$ is [2.5,400], which is not a subset of V1, [20,50]. On the other hand, using the technique presented here, the consistency of constraints is guaranteed.

In most situations, there is always more than one propagation path for tolerance synthesis using CASCADE-T's technique, which means that more than one possible solution can be obtained. However, no methodology to determine the appropriate path is available in Wilhelm and Lu (1992). Only one propagation is possible in the previous situation when A1, W1, W2, A2, and H2 are not available initially. As a result, the propagation path is forced to start from L1 and H1 to find unavailable tolerances sequentially. Using the results in the third column of Table 3 and updating W1 to [2.5,3], Propagation Path 2 (Fig. 9(b)) is provided to synthesize the other tolerances. In this situation, Propagation Path 2 is not the only possible path. Propagation Path 3 (Fig. 9(c)) and Propagation Path 4 (Fig. 9(d)) are the other two possible paths available. The results of these three paths are shown in Table 4, and the resulting tolerances are different. Using the parallel propagation technique given here, tolerances are propagated from level to level simultaneously, multiple propagation paths are not possible, and a unique solution is obtained.

The tolerances resulting from using CASCADE-T may not be feasible. For example, the result of W1 in all possible paths is actually considered infeasible. $W1_{nom}$ computed according to the constraint, C25, and $L1_{nom}$, $H1_{nom}$, and $V1_{nom}$ is 9.33. ($L1_{nom}$, $H1_{nom}$, and $V1_{nom}$ are assumed to be the mid-values of the initial tolerance in the second column in Table 3). W1 in Table 4, [2.5,3], does not include $W1_{nom}$.

7. CONCLUSION

Tolerance design is essential in manufacturing and it plays an important role in relating performance to the design of a product. A good tolerance design mechanism should be able to determine a set of tolerances for the dimensioning of a component's entities such that maximum ranges for the tolerances are obtained while satisfying allowable variations in the functional requirements of the product.

In this paper we have described an approach for worst-case tolerance analysis and synthesis exploiting a directed

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

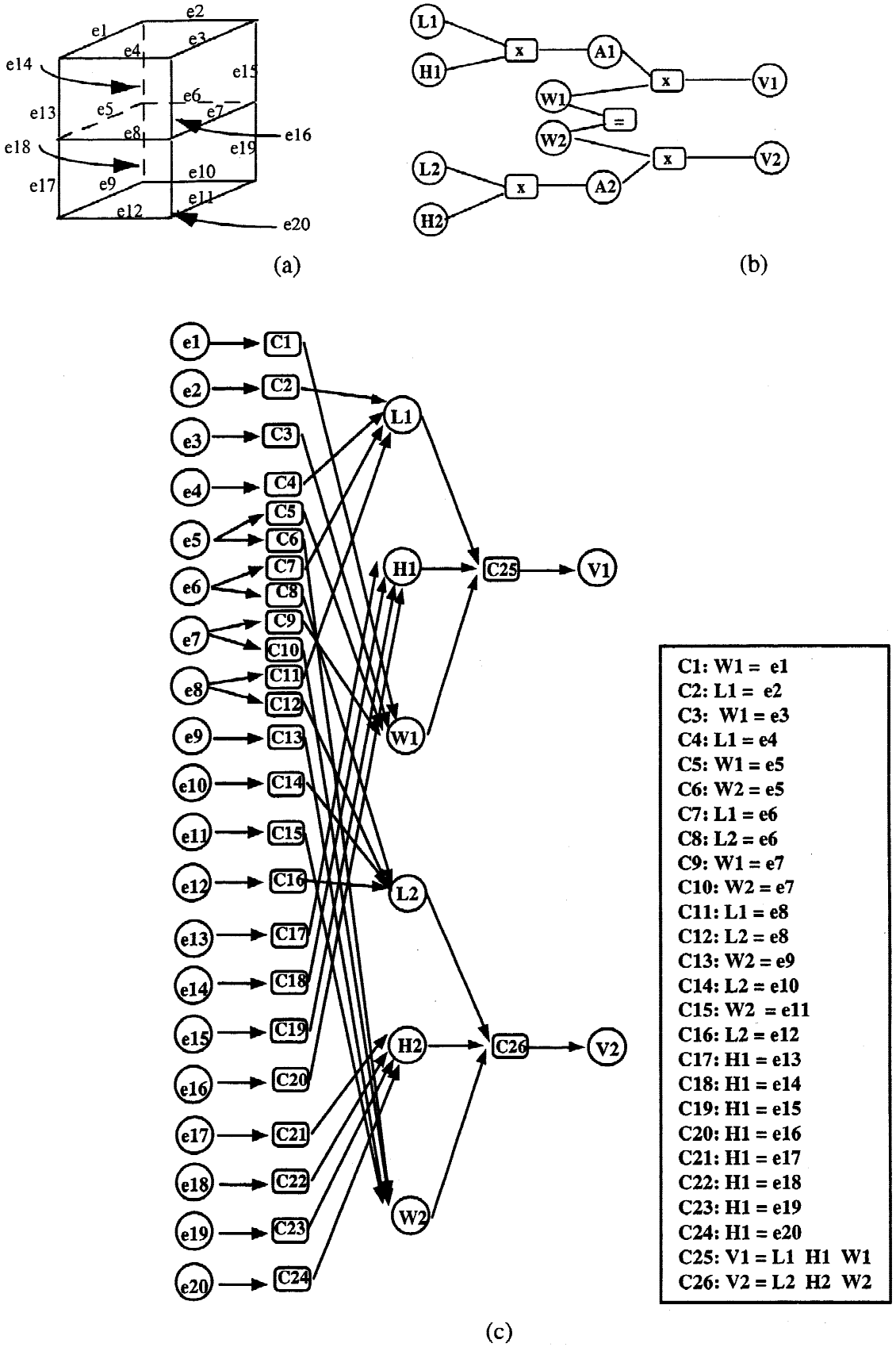


Fig. 8. (a) An example of two prisms, (b) the constraint network provided in Lu and Wilhelm (1991) and Wilhelm and Lu (1992) for the example in (a), (c) our constraint network for the same example.

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

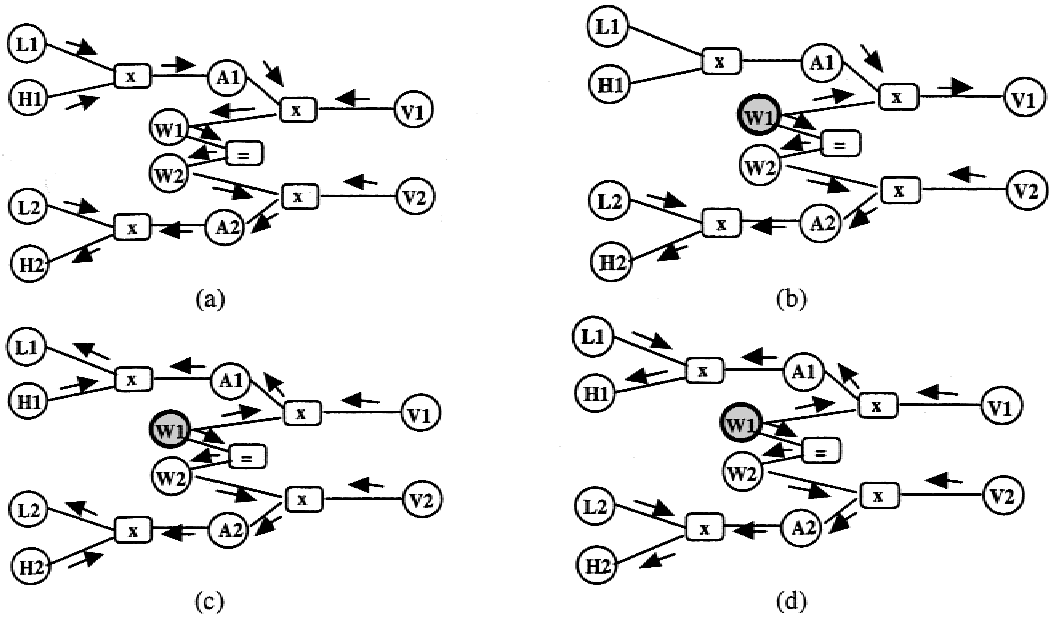


Fig. 9. (a) Propagation Path 1, (b) Propagation Path 2, (c) Propagation Path 3, and (d) Propagation Path 4 for CASCADE-T's constraint network in Example 1.

Table 3. *The result of tolerance synthesis using Propagation Path 1 in CASCADE-T's constraint network (Figure 9(a)) and the parallel propagation in our constraint network (Figure 8(c)) for Example 1*

	Lu's Constraint Network		New Interval Constraint Network	
	Initial	After Propagation	Initial	After Propagation
e1	NA	NA	NA	[8,43,9.97]
e2	NA	NA	NA	[1.28,1.66]
e3	NA	NA	NA	[8,43,9.97]
e4	NA	NA	NA	[1.28,1.66]
e5	NA	NA	NA	[8.47,9.89]
e6	NA	NA	NA	[1.48,1.66]
e7	NA	NA	NA	[8.47,9.89]
e8	NA	NA	NA	[1.48,1.66]
e9	NA	NA	NA	[8.47,9.89]
e10	NA	NA	NA	[1.58,2.29]
e11	NA	NA	NA	[8.47,9.89]
e12	NA	NA	NA	[1.58,2.29]
e13	NA	NA	NA	[1.84,3.00]
e14	NA	NA	NA	[1.84,3.00]
e15	NA	NA	NA	[1.84,3.00]
e16	NA	NA	NA	[1.84,3.00]
e17	NA	NA	NA	[1.48,2.19]
e18	NA	NA	NA	[1.48,2.19]
e19	NA	NA	NA	[1.48,2.19]
e20	NA	NA	NA	[1.48,2.19]
L1	[1,2]	[1,2]	[1,2]	[1.28,1.66]
H1	[1,4]	[1,4]	[1,4]	[1.84,3.00]
W1	NA	[2.5,50]	[7.3,11.3]	[8,43,9.97]
L2	[1,3]	[1,3]	[1,3]	[1.58,2.29]
H2	NA	[0.133,20]	[0.9,2.9]	[1.48,2.19]
W2	NA	[2.5,50]	[7.3,11.3]	[8.47,9.89]
V1	[20,50]	[20,50]	[20,50]	[20,50]
V2	[20,50]	[20,50]	[20,50]	[20,50]

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

Table 4. *Result of tolerance synthesis using CASCADE-T's constraint network and Propagation Path 1, Propagation Path 2, and Propagation Path 3, as shown in Figures 9(b), 9(c), and 9(d), respectively*

	Lu's Constraint Network		New Interval Constraint Network	
	Initial	After Propagation	Initial	After Propagation
e1	NA	NA	NA	[8,43,9.97]
e2	NA	NA	NA	[1.28,1.66]
e3	NA	NA	NA	[8,43,9.97]
e4	NA	NA	NA	[1.28,1.66]
e5	NA	NA	NA	[8.47,9.89]
e6	NA	NA	NA	[1.48,1.66]
e7	NA	NA	NA	[8.47,9.89]
e8	NA	NA	NA	[1.48,1.66]
e9	NA	NA	NA	[8.47,9.89]
e10	NA	NA	NA	[1.58,2.29]
e11	NA	NA	NA	[8.47,9.89]
e12	NA	NA	NA	[1.58,2.29]
e13	NA	NA	NA	[1.84,3.00]
e14	NA	NA	NA	[1.84,3.00]
e15	NA	NA	NA	[1.84,3.00]
e16	NA	NA	NA	[1.84,3.00]
e17	NA	NA	NA	[1.48,2.19]
e18	NA	NA	NA	[1.48,2.19]
e19	NA	NA	NA	[1.48,2.19]
e20	NA	NA	NA	[1.48,2.19]
L1	[1,2]	[1,2]	[1,2]	[1.28,1.66]
H1	[1,4]	[1,4]	[1,4]	[1.84,3.00]
W1	NA	[2.5,50]	[7.3,11.3]	[8,43,9.97]
L2	[1,3]	[1,3]	[1,3]	[1.58,2.29]
H2	NA	[0.133,20]	[0.9,2.9]	[1.48,2.19]
W2	NA	[2.5,50]	[7.3,11.3]	[8.47,9.89]
V1	[20,50]	[20,50]	[20,50]	[20,50]
V2	[20,50]	[20,50]	[20,50]	[20,50]

interval constraint network approach. The concept of consistency of a constraint and the satisfaction of the constraint networks were defined, and techniques for forward and backward propagation between the different levels of such interval constraint networks were developed. Experiments and examples were conducted to illustrate these procedures and to compare with previous related work.

This work has focused on the interval constraint network representations and constraint propagation for automated maintenance of tolerances, not on tolerance allocation. The subjects of cost and optimal tolerance allocation are issues that we intend to investigate in our future research. In this regard, we intend to develop interval propagation methods, which incorporate costs and relative weights in the interval update procedures for the constraints and variables in the network.

REFERENCES

Cagan, J., & Kurfess, T.R. (1992). Optimal tolerance allocation over multiple manufacturing alternatives. *Advances in Design Automation*, DE-Vol. 44-2.

Davis, E. (1987). Constraint propagation with interval labels. *Artificial Intelligence 32*, 281–331.

Dechter, R., & Pearl, J. (1988). Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence 34*, 1–38.

Dechter, R., & Pearl, J. (1989). Tree clustering for constraint networks. *Artificial Intelligence 38*, 353–366.

Fortini, E.T. (1967). *Dimensioning for Interchangeable Manufacture*, Industrial Press, New York.

Foster, L.W. (1983). *Geo-metrics II : The Application of Geometric Tolerancing Techniques*, Addison-Wesley, Reading, Massachusetts.

Guilford, J., Sethi, M., & Turner, J. (1992). Worst case and statistical tolerance analysis of the daughter card assembly. *Proceedings of the 1992 ASME International Computers in Engineering Conference and Exposition*, 343–350.

Hoffmann, P. (1982). Analysis of tolerances and process inaccuracies in discrete part manufacturing. *Computer-Aided Design 14(2)*, 83–88.

Hyvonen, E. (1992). Constraint reasoning based on interval arithmetic: The tolerance propagation approach. *Artificial Intelligence 58(1–3)*, 71–112.

Krishnaswami, M., & Mayne, R.W. (1994). Optimizing tolerance allocation based on manufacturing cost and quality loss. *Advances in Design Automation*, DE-Vol. 69-1.

Ladkin, P.B., & Reinefeld, A. (1992). Effective solution of qualitative interval constraint problems. *Artificial Intelligence 57*, 105–124.

Lu, S.C-Y., & Wilhelm, R.G. (1991). Automating tolerance synthesis: A framework and tools. *Journal of Manufacturing Systems 10(4)*, 279–296.

Mackworth, A.K., & Freuder, E.C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence 25*, 65–74.

Michael, W., & Siddall, J.N. (1981). The optimization problem with optimal tolerance assignment with full acceptance. *Journal of Mechanical Design 103*, 842–848.

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

Michael, W., & Siddall, J.N. (1982). The optimal tolerance assignment with less than full acceptance. *Journal of Mechanical Design 104*, 855–860.
Ostwald, P.F., & Huang, J. (1977). A method for optimal tolerance selection. *ASME Transactions, Journal of Engineering for Industry 99*, 558–565.
Parkinson, D.B. (1982). The application of reliability methods to toleranc-ing. *Transactions of the ASME Journal of Mechanical Design 104*, 612–618.
Parkinson, D.B. (1984). Tolerancing of component dimensions in CAD. *Computer-Aided Design 16(1)*, 25–32.
Parkinson, D.B. (1985). Assessment and optimization of dimensional tol-erances. *Computer-Aided Design 17(4)*, 191–199.
Soderberg, R. (1994). Robust design by tolerance allocation considering quality and manufacturing cost. *Advances in Design Automation*, DE-Vol. 69-1.
Speckhart, F.H. (1972). Calculation of tolerance based on a minimum cost approach. *ASME Transactions, Journal of Engineering for Industry 94*, 447–453.
Spotts, M.F. (1973). Allocation of tolerances to minimize cost of assem-bly. *ASME Transactions, Journal of Engineering for Industry 95*, 762–764.
Spotts, M.F. (1983). *Dimension and Tolerancing for Quantity Production*, Prentice Hall, Englewood Cliffs, New Jersey.
Turner, J.U. (1987). Tolerances in computer-aided geometric design. Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY.
Wilhelm, R.G., & Lu, S.C.-Y. (1992). *Computer Methods for Tolerance Design*, World Scientific Publishing Co.

APPENDIX

Proof of theorem 1 and corollary 1

In backward propagation, solutions are determined based on the constraint function. For the constraint function, $Y = F(X_1,\ldots,X_n)$, the upper limit of Y, y_{up}, is propagated to the upper limits or lower limits of X_i, where $i = 1, 2,\ldots,n$. The propagation technique is based on the propagation function of y_{up}, $y_{up} = f(x_{1\varphi} \pm \alpha_1 \tau_{up},\ldots,x_{n\varphi} \pm \alpha_n \tau_{up})$. Given y_{up}, $x_{i\varphi}$, and α_i $(i = 1, 2,\ldots,n)$, τ_{up} are solved. The updated value of $x_{i\varphi}$, $x'_{i\varphi}$, is $x_{i\varphi} \pm \alpha_i \tau_{up}$.

For feasible solution, $x_i \le x'_{i\varphi} \le x_{i\varphi}$ if $x_{i\varphi} = x_{i_{up}}$, is $x_{i\varphi} \le x'_{i\varphi} \le x_i$ if $x_{i\varphi} = x_{i_{low}}$. That means,

$$\text{if } x_{i\varphi} = x_{i_{up}} \quad x_i \le x_{i\varphi} + \alpha_i \tau_{up} \le x_{i\varphi}$$
$$-(x_{i\varphi} - x_i) \le \alpha_i \tau_{up} \le 0$$

$$\text{if } x_{i\varphi} = x_{i_{low}} \quad x_{i\varphi} \le x_{i\varphi} - \alpha_i \tau_{up} \le x_i$$
$$0 \le -\alpha_i \tau_{up} \le x_i - x_{i\varphi}$$

Similar situations hold for the propagation of the lower limit of Y, y_{low}.

$$y_{low} = f(x_{1\kappa} \pm \beta_1 \tau_{low},\ldots,x_{n\kappa} \pm \beta_n \tau_{low})$$

Given y_1, $x_{i\kappa}$, and $\beta_i (i = 1, 2,\ldots,n)$, τ_{low} are solved. The updated value of $x_{i\kappa}$, $x'_{i\kappa}$, is $x_{i\kappa} \pm \beta_i \tau_{low}$.

For feasible solution, $x_{i\kappa} \le x'_{i\kappa} \le x_i$ if $x_{i\kappa} = x_{i_{low}}$, is $x_i \le x'_{i\kappa} \le x_{i\kappa}$ if $x_{i\kappa} = x_{i_{up}}$. That means,

$$\text{if } x_{i\kappa} = x_{i_{low}} \quad x_{i\kappa} \le x_{i\kappa} + \beta_i \tau_{low} \le x_i$$
$$0 \le \beta_i \tau_{low} \le x_i - x_{i\kappa}$$

$$\text{if } x_{i\kappa} = x_{i_{up}} \quad x_i \le x_{i\kappa} - \beta_i \tau_{low} \le x_{i\kappa}$$
$$-(x_{i\kappa} - x_i) \le -\beta_i \tau_{low} \le 0$$

For example, Figure A1 shows a simple single constraint with subtraction operation. The amounts of tightening on the lower limits and the upper limits of X_1 and X_2 in terms of α_1, β_1, α_2, β_2, τ_{low}, and τ_{up} are shown in Figure A1(b).

For uniform allocation, $\alpha_i = \beta_i = 1$. Therefore,

$$-(x_{i\varphi} - x_i) \le \tau_{up} \le 0 \qquad \text{if } x_{i\varphi} = x_{i_{up}}$$
$$0 \le -\tau_{up} \le x_i - x_{i\varphi} \qquad \text{if } x_{i\varphi} = x_{i_{low}}$$
$$0 \le \tau_{low} \le x_i - x_{i\kappa} \qquad \text{if } x_{i\kappa} = x_{i_{low}}$$
$$-(x_{i\kappa} - x_i) \le -\tau_{low} \le 0 \qquad \text{if } x_{i\kappa} = x_{i_{up}}$$

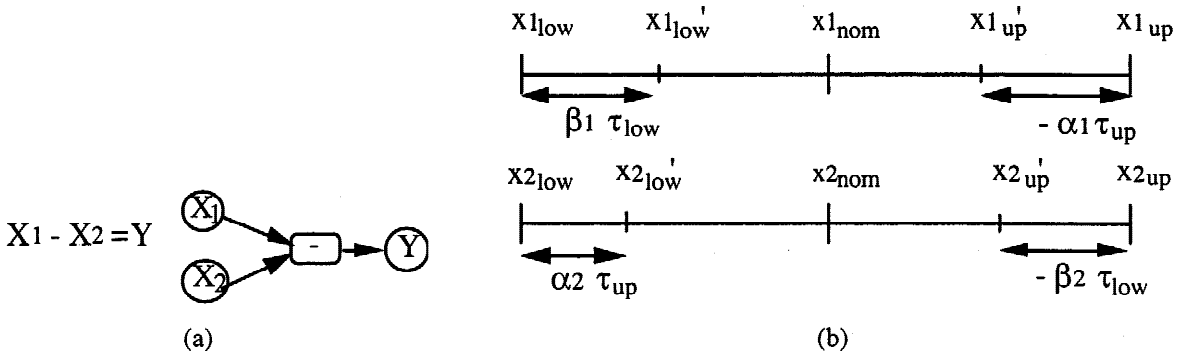


Fig. A1. (a) A single constraint with subtraction operation, (b) the amount of tightening on X_1 are $\beta_1 \tau_{low}$ and $-\alpha_1 \tau_{up}$ on the lower limit and the upper limit, respectively, and the amount of tightening on X_2 are $\alpha_2 \tau_{up}$ and $-\beta_2 \tau_{low}$ on the lower limit and the upper limit, respectively.

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press

For allocation based on nominal values, $\alpha_i = \beta_i = x_i$. Therefore,

$$
\begin{aligned}
&-(x_{i\varphi}/x_i - 1) \le \tau_{up} \le 0 && \text{if } x_{i\varphi} = x_{i_{up}} \\
&0 \le -\tau_{up} \le 1 - x_{i\varphi}/x_i && \text{if } x_{i\varphi} = x_{i_{low}} \\
&0 \le \tau_{low} \le 1 - x_{i\kappa}/x_i && \text{if } x_{i\kappa} = x_{i_{low}} \\
&-(x_{i\kappa}/x_i - 1) \le -\tau_{low} \le 0 && \text{if } x_{i\kappa} = x_{i_{up}}
\end{aligned}
$$

For allocation based on the interval sizes, $\alpha_i = x_{i\varphi} - x_i$ if $x_{i\varphi} = x_{i_{up}}$, $\alpha_i = x_i - x_{i\varphi}$ if $x_{i\varphi} = x_{i_{low}}$, $\beta_i = x_i - x_{i\kappa}$ if $x_{i\kappa} = x_{i_{low}}$, and $\beta_i = x_{i\kappa} - x_i$ if $x_{i\kappa} = x_{i_{up}}$. Therefore,

$$
\begin{aligned}
&-1 \le \tau_{up} \le 0 && \text{if } x_{i\varphi} = x_{i_{up}} \text{ and } x_{i\varphi} = x_{i_{low}} \\
&0 \le \tau_{low} \le 1 && \text{if } x_{i\kappa} = x_{i_{low}} \text{ and } x_{i\kappa} = x_{i_{up}}
\end{aligned}
$$

Based the above analysis, it is possible that allocation techniques based on uniform or nominal values are not able to obtain the feasible solution, even if there is one. However, allocation techniques based on tolerance sizes can always find the feasible solution if there is one. The allocation techniques based on uniform or nominal values cannot find a solution if any of the upper limits or lower limits of the updated input intervals do not satisfy one of the conditions listed. However, the unsatisfactory condition produced when one of the upper or lower limits of the updated input intervals fails to satisfy one of the pertinent conditions does not imply that there is no feasible solution. This is because it is possible to relax one or more of the other intervals such that the unsatisfactory condition becomes satisfied. In the allocation techniques based on tolerance size, the conditions for obtaining a solution are $-1 \le \tau_u \le 0$ and $0 \le \tau_1 \le 1$, which are the same for all intervals. If one of these conditions is not satisfied, it is not satisfied for all upper limits or lower limits for all intervals and no relaxation is available to obtain a feasible solution. In other words, there is no feasible solution if the allocation technique based on interval sizes cannot obtain a feasible solution.

Christopher C. Yang was born in Hong Kong. He received his B.S., M.S., and Ph.D. in Electrical Engineering from the University of Arizona in 1990, 1992, and 1997, respectively. He was a research associate in the Intelligent Systems Laboratory of the University of Arizona from 1992 to 1997. He is a currently an assistant professor in Computer Science Department, University of Hong Kong. His current research lies mainly in active vision, color image processing, visual inspection, tolerance design, and computer-integrated manufacturing.

Michael M. Marefat is a senior member of IEEE, a member of ACM, and a member of AAAI. He received the PhD and MS in Electrical Engineering from the Purdue University (1991 and 1988 respectively), and the BS in Electrical and Computer Engineering and BA in Mathematical Sciences from the Rice University in 1986. He is the Director of the Intelligent Systems Laboratory and an associate professor at the Department of Electrical and Computer Engineering at the University of Arizona. Dr. Marefat has been involved in research in the areas of knowledge-based systems, visualization, computer graphics, machine vision, and CAD for the last 10 years. He is the author or co-author of about 100 refereed research articles. His research has been supported by the National Science Foundation, NASA, the state of Arizona, and the Department of Energy. He has been the supervisor of more than 12 masters theses and Ph.D. dissertations. His previous experience includes Schlumberger Laboratory for Computer Science, Purdue Engineering Research Center for Intelligent Manufacturing Systems, and Baylor Digital Imaging Laboratory.

Frank W. Ciarallo is an assistant professor in the Department of Systems and Industrial Engineering at the University of Arizona. He received his Ph.D. (1993) and M.S. (1988) degrees in manufacturing and operation systems from the Graduate School of Industrial Administration at Carnegie Mellon University. He received his B.Sc. (1986) degree in electrical engineering, and engineering and public policy, also from Carnegie Mellon. His research interests include production planning and inventory control with process uncertainties, optimization and uncertainties in inspection planning and applications of simulation. Dr. Ciarallo is a member of IEEE, and the Institute for Operations Research and the Management Sciences.

https://doi.org/10.1017/S0890060400144014 Published online by Cambridge University Press