Proceedings of the 1997 IEEE
International Conference on Robotics and Automation
Albuquerque, New Mexico - April 1997

Tolerance Analysis and Synthesis by Interval Constraint Networks

Christopher C. Yang*, Michael M. Marefat*, and Frank W. Ciarallo**
*: Department of Electrical and Computer Engineering
**: Department of Systems and Industrial Engineering
The University of Arizona, Tucson, AZ 85721

Abstract

This paper proposes interval constraint network and interval propagation techniques for automatic tolerance design. A hierarchical representation is utilized in the interval constraint network. The consistency of a constraint is defined for the purpose of tolerance design. Forward and backward propagation techniques are introduced in the interval constraint network for tolerance analysis and synthesis, respectively. Both a propagation technique for a single constraint and a parallel propagation technique for multiple constraints between two adjacent levels in the network are introduced. Experiments conducted to illustrate the procedures of tolerance analysis and synthesis for the tank problem are described.

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

1. Introduction

Tolerance design plays an important a role in the relationship between performance and the manufacturing cost of a product. Decreasing the tolerance range will improve performance but will also increase manufacturing costs. It is desirable to optimize the tolerance range under such constraints of product design as 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 a given set of 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: $Y = f(X_1, X_2, ..., X_n)$ where Y is the functional requirement and X_i is the i^{th} entity. n is the number of entities that are related by the equation to the corresponding functional requirement.

In *tolerance analysis*, the entity tolerances, $X_1, X_2, ..., X_n$, 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, X_i and Y, respectively. If the assigned functional requirement tolerances are not met, the tolerances for the entities need to be reassigned by *tolerance synthesis* in order 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, $X_1, X_2, ..., X_n$, to 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.

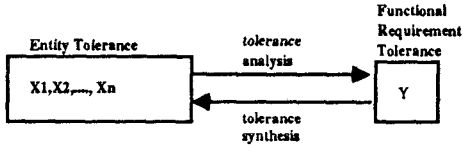

Figure 1. Relationship of tolerance synthesis and tolerance analysis

1.1 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.

The previous works are summarized as below. Foster [5] developed formulas for calculating size position tolerances to achieve a desired class and grade of cylindrical fit between mating parts. Turner et al. [12] presented a worst-case tolerance analysis on an industrial assembly using GEOS (an automated tolerance analysis package developed at the Rensselaer Design Research Center). Fortini [4] and Parkinson [11] applied statistical analysis techniques. Michael and Siddall [10], and Cagan et al. [1] have applied optimization techniques for tolerance synthesis. These techniques include linear and nonliear programming, and simulated annealing. Lu and Wilhelm [8] proposed a tolerance synthesis approach, CASCADE-T, that used a representation of the conditional tolerance relations that exist between features of a part.

1.2 Related Work in Interval Constraints

Constraint satisfaction problems (CSPs) are often formulated in AI tasks. 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. 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 [3] 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. Mackworth and Freduer [9] analyzed the time complexity of several node, arc and path consistency algorithms in CSPs. However, the domains of the variables

considered by [3,9] are discrete, finite sets instead of real intervals.

Ladkin and Reinefeld [7] 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 and Hyvonen's 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 intervals. Davis [2] adapted the Waltz filtering algorithm for screening impossible values from the variable domain to solve the ICSPs. Hyvonen [6] 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. While these methods are able to determine the solutions for ICSPs, their definition of consistency and satisfaction of the constraint network is not appropriate in the problem of tolerance synthesis and analysis in mechanical parts. Therefore, these techniques cannot solve our problem. The differences of the definitions and applications between their work and our work will be discussed in the later sections.

1.3 Contributions

We have approached the problem of tolerance design by combining interval constraint network and interval propagation techniques. The contribution of our work can be summarized as follows:

- A hierarchical interval constraint network is developed to represent the relationships (constraints) between the entities, attributes, and functional requirement of a mechanical part.
- The consistency of a constraint is defined for the purpose of tolerance design in the interval constraint network, since the traditional definition of variable's consistency used in interval constraint satisfaction problem (ICSP) cannot be applied to tolerance design. The definition of constraint consistency is then used to define the satisfaction of the interval constraint network for tolerance design.
- Forward and backward propagation for a single constraint are developed.
- A parallel propagation method between adjacent levels of the hierarchical constraint network is developed for tolerance analysis and tolerance synthesis.

2. Interval Constraint Network for Tolerance Design

2.1. Hierarchical Interval Constraint Network and Constraint Functions

For each mechanical design, the relationship between the highest level, functional requirement, and the lowest level, entity, can be represented by a hierarchical network. 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 attribute, *inner radius*. An attribute is also described as a function in terms of the mechanical part's entities. The inner radius can be computed as a function in terms of the outer radius and the thickness of the material. These relationships are described as a hierarchical interval constraint network as shown in Figure 2 in our approach.

The *constraint functions* in the constraint network, $O = f(I_1, I_2, \ldots, I_n)$, describe the relationships between the multiple input variables, $I_1, I_2, \ldots, I_n$, and the single output variable, 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.

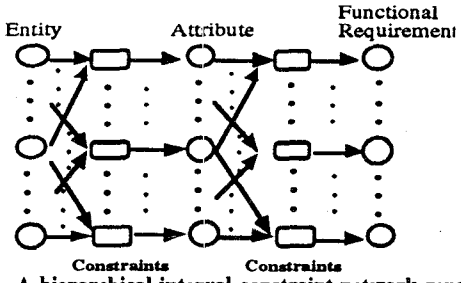

Figure 2. A hierarchical interval constraint network representing the constraints between the entities, attributes, and functional requirements of the mechanical design.

2.2 Satisfaction of Interval Constraint Network

The definition of satisfaction in the interval constraint network always depends on the purpose of application. A good understanding of the goal of constraint satisfaction for the problem investigated and an appropriate definition of the satisfaction of the constraint network are the foundations of a successful constraint network solution to the problem.

The interval constraint satisfaction problem (ICSP) has been studied by Davis [2], Hyvonen [6] and others. The satisfaction of the network is defined in terms of the consistency of the variables. The purpose of the ICSP is to refine the intervals of the variables as far as possible without losing possible exact solutions of the constraints. However, this definition does not fit the purpose of our application to tolerance design. The purpose here is to refine the tolerances of all the input and output variables of the constraints such that the tolerance propagated from the input variables based on the interval constraint function is a subset of the output variable tolerances. As a result, the definition of constraint network satisfaction in our problem should be modified and described in terms of the consistency of the constraints not variables.

2.2.1 Satisfaction of Interval Constraint Network for ICSP

In ICSP, according to Hyvonen [6], the satisfaction of the interval constraint network is defined as follows:

- A variable, V_i, is consistent if and only if $\forall(v_i \in V_i \mid V_i = v_i), \exists(v_1 \in V_1, \ldots, v_{i-1} \in V_{i-1}, v_{i+1} \in V_{i+1}, \ldots, v_n \in V_n \mid V_1 = v_1, \ldots, V_{i-1} = v_{i-1}, V_{i+1} = v_{i+1}, \ldots, V_n = v_n)$, 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, V_1 to V_n, and the constraint between the variable, V_i, and the other variables is described by $V_i = f(V_1, \ldots, V_n)$, V_i is consistent if and only if $V_i \subseteq F(V_1, \ldots, V_n)$. However, in tolerance design, it is desired to have the interval computed using the input intervals and the interval constraint function to be a subset of the assigned output interval for each constraint in the network.

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 rectangal is $V = f(L, W) = L \times W$. Given the tolerances, 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 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.

2.2.2 Satisfaction of Interval Constraint Network for Tolerance Design

Since the purpose of tolerance design in an interval constraint network is different from the purpose of ICSP, a new definition of satisfaction is required. The satisfaction of the network depends on the consistency of the components in the network. In ICSP, the satisfaction of the network is defined in terms of the consistency of the variables. However, based on the application of tolerance design, the definition of consistency should focus on constraints.

In a constraint network for tolerance design, the constraint is multiple/single inputs and single output (MISO or SISO) 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 $V_3 = f(V_1,V_2)$.

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

Definition 1:

A constraint, $C_i(U,k)$, is consistent if and only if

$$\bigcap_{j \in U} (\forall v_j \in V_j \mid V_j = v_j) , (\exists v_k \in V_k \mid V_k = v_k)$$

such that $C_i(U,k)$ 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 of 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 L and W, and there is always a value, a, in A such that $l \times w = a$.

3 Tolerance Propagation

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

Tolerance propagation is utilized to update the intervals in the network to make the interval constraints consistent. Tolerance can be propagated from the input intervals of a constraint to the single output interval, which is known as *forward propagation*. Tolerance can also be propagated from the single output interval of a constraint to multiple input intervals, known as *backward propagation*. The forward and backward propagation techniques for tolerance design are developed based on Definitions 1 and 2. Given a constraint with constraint function $X_k = f(X_1,X_2, ...,X_n)$, with input intervals, $X_1,X_2, ...,X_n$, and output interval, X_k, if the constraint is not consistent ($X_k \not\subset F(X_1,X_2, ...,X_n)$), either X_k must be relaxed (widened) or one or more of the input intervals must be tightened (narrowed). X_k is relaxed by propagating X_1,X_2, ..., and X_n forward. X_1,X_2, ..., and X_n are tightened by propagating X_k backward.

3.1 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:

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

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

$$x_{k_{up}}' = f(x_{1\varphi}, ..., x_{n\varphi})$$

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

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

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

$$x_{k_{low}}' = f(x_{1\kappa}, ..., x_{n\kappa})$$

where $x_{i\kappa} = x_{i_{up}}$ if X_k is monotonic increasing with respect to X_i.

$x_{i\kappa} = x_{i_{low}}$ if X_k is monotonic decreasing with respect to 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 $x_{k_{low}}' < x_{k_{low}}$.

3.2 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: (i) tightening uniformly on every interval, (ii) tightening the intervals proportional to the corresponding nominal values of the variables, (iii) tightening the intervals proportional to the width of the intervals. The algorithm for backward propagation is given as:

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

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 X_1 to X_n,

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

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

$\alpha_i = x_{i\varphi} - x_{i_{nom}}$ if $x_{i\varphi} = x_{i_{up}}$

$\alpha_i = x_{i_{nom}} - x_{i\varphi}$ if $x_{i\varphi} = x_{i_{low}}$

and $\beta_i = x_{i\kappa} - x_{i_{nom}}$ if $x_{i\kappa} = x_{i_{up}}$

$\beta_i = x_{i_{nom}} - x_{i\kappa}$ if $x_{i\varphi} = x_{i_{low}}$

for $1 \le i \le n$.

where

$x_{i\varphi} = x_{i_{up}}$ if X_k is monotonic increasing with respect to X_i.

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

$x_{i\kappa} = x_{i_{low}}$ if X_k is monotonic increasing with respect to X_i.

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

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

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

where

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

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

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

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

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

where

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

$\Delta x_{i\kappa} = -\beta_i \tau_{low}$ if X_k is monotonic decreasing with respect to 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 $1 \le i \le n$.

3.3 Parallel Propagation

In Sections 3.1 and 3.2, propagation for a single constraint has been introduced. However, an interval constraint network usually consists of more than one constraint. The order of propagation has a significant effect on the final solution obtained. In this paper, parallel propagation is utilized in the hierarchical interval constraint network.

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 $C_{(i, i+1)}$

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.

Propagate to the intervals in Level_i-1 with out-degree greater than one

Let M be the set of the intervals in Level_i, X_j, which are constrained by more than one constraint between Level_i and Level_i+1.

For each interval in M,

$X_{j_{min}} = X_j$

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

Propagate the output intervals in Level_i+1 to the intervals in M simultaneously.

If updated interval of $X_j \cap X_{j_{min}} \neq \emptyset$,

$X_j = X_j \cap X_{j_{min}}$.

Propagate to the intervals in Level_i with out-degree is equal to one

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

Propagate the output intervals in Level_i+1 to the intervals in Level_i, which are not elements of M, with α_j and $\beta_j = 0$ if $X_j \in M$, simultaneously using the technique in Section 3.1

In parallel forward propagation, all the intervals in Level_i are propagated to all the intervals in Level_i+1 simultaneously. Since all the constraints between the two levels are multiple inputs / single output, the updated intervals in Level_i+1 do not affect one another. However, 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. Therefore, the updated intervals in Level_i correspond only to the last propagated constraint. Other constraints propagated through earlier may no longer be consistent. As a result, two parallel backward propagations are needed to ensure the consistency of all the constraints between the two levels of variables. One propagation is for the intervals in Level_i which 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 another propagation is for the rest of the intervals in Level_i.

In parallel backward propagation, the intervals in Level_i+1 are first propagated to those intervals in Level_i which are constrained by more than one constraint between Level_i and Level_i+1. The tightest constraint from all the output intervals on these intervals is found and saved. Then, a second parallel backward propagation will be processed to update the rest of the intervals in Level_i without changing the intervals which have already been computed during the first stage propagation. Using this technique, all the constraints between Level_i+1 and Level_i are ensured to be consistent.

3.3.1 Example

Figure 3 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 $X_1 = [5,10]$, $X_2 = [20,25]$, $X_3 = [15,18]$, $X_4 = [9,10]$, $Y_1 = [1,3]$, $Y_2 = [40,50]$, $Y_3 = [140,180]$, $x1_{nom} = 7$, $x2_{nom} = 22$, $x3_{nom} = 17$, $x4_{nom} = 9.5$, the constraints are $Ln(X_1) = Y_1$, $X_1 + X_2 + X_3 = Y_2$, and $X_3 \times X_4 = Y_3$. 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 finally 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.

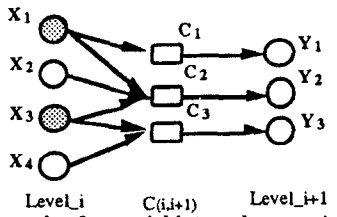

Figure 3. An example of a partial interval constraint network from Level_i to Level_i+1, where Level_i has four variables, X_1, X_2, X_3, and X_4, Level_i+1 has three variables, Y_1, Y_2, and Y_3, and there are three constraints in C(i,i+1), C_1, C_2, and C_3.

For parallel backward propagation, only the original Y_2 and Y_3 are first propagated backward simultaneously to X_1, X_2, X_3, and X_4 because C_1 is already consistent. $BP(Y_2; X_1, X_2, X_3)$ and $BP(Y_3; X_3, X_4)$ are first processed to update X_3. Allocation based on the width of intervals is used in this example. $BP(Y_2; X_1, X_2, X_3)$ 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]. Again, 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.

4 Tolerance Design, Example and Experimental Results

In this section, the tank in Figure 4 is utilized as an example to illustrate tolerance analysis and synthesis in 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 4(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 4(c)). The entities are several measurable tank lengths, as labeled in Figure 4(d).The hierarchical interval constraint network for the tank is given in Figure 5. The constraint functions are given in Table. 1.

4.1 Tolerance Analysis

In tolerance analysis, tolerances are assigned to the entities of the mechanical parts, and analysis ensures that the tolerance of the functional requirements is satisfied. Therefore, tolerances are propagated from all the entities to the tolerances of 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 the tolerances are set to the nominal value in the constraint network.

In the example of the tank, 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 95mm, 200mm, 100mm, 50mm, 50mm, 190mm, and 200mm, respectively. The nominal values of the attributes, **L1**, **L2**, **L3**, **R1**, **R2**, **R3**, and **R4**, are 100mm, 200mm, 95mm, 140mm, 190mm, 150mm, and 200mm, respectively. The nominal values of the functional requirements, **V**, **T1**, **T2**, and **T3**, are $2.9\text{x}10^7 mm^3$, 10mm, 10mm, and 5mm, 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 and the results are also given in Table 2. The propagated tolerances of the functional requirements are $[2.8241\text{x}10^7 mm^3, 2.9419\text{x}10^7 mm^3]$, [8mm,12mm], [6mm,14mm] and [3mm,7mm] 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 the designed tolerance as shown in Column 3 of Table 2. However, the propagated tolerances of **V** do satisfy that assignment.

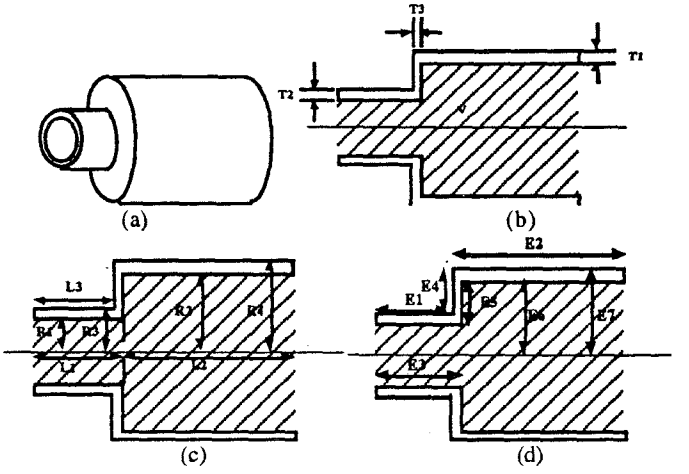

Figure 4. (a) A tank, (b) the labels of functional requirement, (c) the labels of attributes of the cylinders, and (d) the labels of the measurable entities.

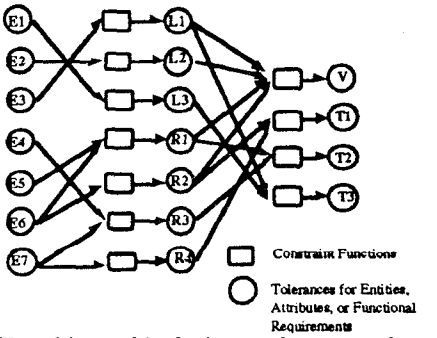

Figure 5. The hierarchical interval constraint network representing the relationship of the tolerance of entities, attributes, and functional requirements for the tank in Figure 4.

4.2 Tolerance Synthesis

In tolerance synthesis, the tolerances of functional requirements that do not satisfy the assignment in tolerance analysis are propagated backward to the entities' tolerances.

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. 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.

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

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

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

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

In the example of the tank, only the tolerances of **T1**, **T2**, and **T3** are propagated backward to the attributes' tolerances of **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 in tolerance analysis as described 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 of the attributes are given in Table 2. Since none of **L1**, **L3**, **R1**, **R2**, **R3**, and **R4** is 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 propagation is first processed to propagate the tolerances from **R1**, **R2**, **R3**, and **R4** to **E6** and **E7**. A second parallel 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.

5. Conclusion

Tolerance design is essential in manufacturing and it plays an important role in relating performance to the manufacturing cost of a product. A good tolerance design method should be able to assign a set of tolerances for the dimensioning entities such that the maximum ranges of tolerance are obtained while satisfying the functional requirement. In this paper, a hierarchical interval constraint network is described and the techniques for tolerance propagation are developed. The contributions are summarized as follows:

(1) A hierarchical interval constraint network to represent the relationships among the functional requirements, attributes, and entities is developed.
(2) The consistency of the constraint and the satisfaction of the constraint network are defined.
(3) The techniques of forward and backward propagation for a single constraint is developed. The techniques for parallel forward and backward propagation between the different levels of such interval constraint networks are also developed.
(4) Techniques for tolerance design and tolerance synthesis based on the proposed hierarchical interval constraint networks are introduced.

6. Acknowledgments

This research has been supported by the National Science Foundation under the grant IRI-9414523 to Dr. Michael M. Marefat. The provided support is gratefully appreciated.

References

[1] Jonathan Cagan and Thomas R. Kurfess, "Optimal Tolerance Allocation over Multiple Manufacturing Alternatives," *Advances in Design Automation*, DE-Vol. 44-2, 1992.
[2] Ernest Davis, "Constraint Propagation With Interval Labels," *Artificial Intelligence*, Vol. 32, p. 281-331, 1987
[3] Rina Dechter and Judea Pearl, "Tree Clustering for Constraint Networks," *Artificial Intelligence*, Vol. 38, p. 353-366, 1989.
[4] E. T. Fortini, *Dimensioning for Interchangeable Manufacture*, Industrial Press, New York, 1967.
[5] Lowell W. Foster, *Geo-metrics II : The Application of Geometric Tolerancing Techniques*, Addison- Wesley, Reading, Massachusetts, 1983.
[6] Eero Hyvonen, "Constraint Reasoning Based on Interval Arithmetic: the Tolerance Propagation Approach," *Artificial Intelligence*, Vol. 58, No. 1-3, December, p. 71-112, 1992.
[7] Peter B. Ladkin, and Alexander Reinefeld, "Effective Solution of Qualitative Interval Constraint Problems," *Artificial Intelligence*, Vol. 57, p. 105-124, 1992.
[8] Stephen C-Y. Lu and Robert G. Wilhelm, "Automating Tolerance Synthesis: a Framework and Tools," *Journal of Manufacturing systems*, Vol. 10, No. 4, p. 279-296, 1991.
[9] Alan K. Mackworth, and Eugene C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems," *Artificial Intelligence*, Vol. 25, p. 65-74, 1985.
[10] W. Michael and J. N. Siddall, "The Optimal Tolerance Assignment with Less Than Full Acceptance" *Journal of Mechanical Design*, Vol. 104, October, p. 855-860, 1982.
[11] D. B. Parkinson, "Assessment and Optimization of Dimensional Tolerances," *Computer-Aided Design*, Vol. 17, No. 4, May, p. 191-199, 1985.
[12] Joshua U. Turner, "Tolerances in Computer-Aided Geometric Design," *Ph.D. Dissertation*, Rensselaer Polytechnic Institute, 1987.