JAWS2012

JAWS2012

Formalizing the resilience of open dynamic systems

Kazuhiro Minami — Institute of Statistical Mathematics, kminami@ism.ac.jp

Tenda Okimoto — Transdisciplinary Research Integration Center, tenda@nii.ac.jp

Tomoya Tanjo — Transdisciplinary Research Integration Center, tanjo@nii.ac.jp

Nicolas Schwind — National Institute of Informatics, schwind@nii.ac.jp

Hei Chan — Transdisciplinary Research Integration Center, hei@nii.ac.jp

Katsumi Inoue — National Institute of Informatics, inoue@nii.ac.jp

Hiroshi Maruyama — Institute of Statistical Mathematics, hm2@ism.ac.jp

keywords: Systems resilience, Dynamic systems, Constraint satisfaction problems Multi-agent systems

Summary

Many people have realized the importance of building resilient systems that can absorb shocks from unexpected incidents (e.g., a large tsunami) and recover from unavoidable damages in a graceful way. However, there is no formal definition of resilience based on which researchers in various fields can study general design principles for building resilient systems. In this paper, we formalize the notion of resilience within the theoretical framework of dynamic constraint satisfaction problems (DCSPs) and identify several important directions for systems resilience research.

1. Introduction

After the 3.11 earthquake, many people realized that there are events that cannot be reasonably anticipated. These "unexpected" events occur outside of the anticipated envelope (e.g., Tsunami of 14m high vs the anticipated max of 5.7m), or something completely unheard of (e.g., Tokyo subway gas attack in 1995). We recognize that these unexpected events do happen, but because they are "unexpected," we cannot prepare for the event and protect our systems. The only thing we can do is to give resistance that locally contains damages from the event and recover from that damage as quickly and as inexpensively as possible. We call this combination of resistance and recovery the *resilience* of the system.

Many researchers have recognized the importance of establishing a new research discipline concerning the resilience of complex systems to provide a set of general principles for building resilient systems in various domains. Although we have seen many examples of seemingly resilient systems in various fields, such as biology and computer science, researchers have not agreed on a common definition on resilience yet and it is thus not clear how we should adopt an effective strategy in one domain to systems in another. Therefore, we set out to establish a new research discipline that we call "systems resilience," which provides a set of unified design principle for building resilient systems [Maruyama 12].

In this paper, we present a system model based on the framework of dynamic constraint satisfaction problems (DCSPs) [Faltings 05, Verfaillie 05] and formally define the notion of *resilience* of open dynamic systems, which consist of a dynamically changing set of entities (i.e., agents). Our system model represents a snapshot of a dynamic discrete system with a set of variables and specify the requirements for the system as a set of constraints on those system variables. In addition, to express laws of causality governing dynamic systems, we introduce the notion of *transitional* constraints that restrict transitions among snapshots of the system.

Our resilience definition consists of two separate, yet closely related, properties: One is the *recoverability* of a system from a certain undesirable state. We consider a system snapshot desirable or not depending on whether

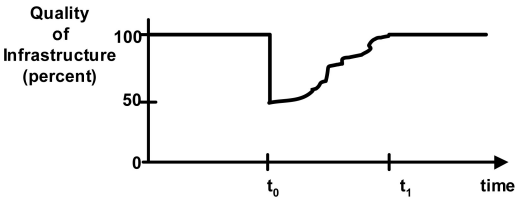

Fig. 1 Bruneau's definition of resilience.

it satisfies all the constraints, and measure how long the system takes to recover from an undesirable state to a desirable state. The other is the *resistance* to a disturbance to a system, which is represented as a number of unsatisfied constraints. We finally obtain a general definition of resilience by integrating those two concepts.

The rest of the paper is organized as follows. We first look over definitions of resilience in previous research in Chapter 2 and present our system model based on the DCAP framework in Chapter 3. In Chapter 4, we formally define the notion of resilience, which captures systems' ability for resistance and recoverability precisely. We then discuss open research issues and related work in Chapter 5 and Chapter 6, respectively, and finally conclude in Chapter 7.

2. Background

The concept of resilience appears in various disciplines such as environmental science, materials science, sociology, and so on. Holling [Holling 73] first introduced the concept of resilience in ecology. Holling defined the resilience as the capacity of an ecosystem to respond to a perturbation or disturbance by resisting damage and recovering quickly.

Bruneau [Bruneau 03]'s definition of seismic resilience for disaster prevention elaborates the concept of the resilience by introducing quantitative measures. Bruneau considers a situation where a system's quality degrades abruptly at time t_0 due to some unexpected event and fully recover to the original state at time t_1, as shown in Figure 1. If we denote by $Q(t)$ the quality of the system at time t, the resilience of the system is measured as follows:

$$\int_{t_0}^{t_1} [100 - Q(t)]dt$$

As the measured triangle area gets smaller, the system becomes more resilient. That is, there are two dimensions concerning the resiliency of the system.

- Resistance (reduced service degradation from failures at time t_0)
- Recoverability (reduced time to recovery (i.e., time interval between t_0 and t_1))

Although Bruneau points out that the notion of resilience is the combination of the above two properties, there is no clear way to quantify them. We formalize Bruneau's definition of resilience to provide quantitative metrics for those properties.

3. System Model

We model systems as dynamic constraint satisfaction problems (DCSPs) to give a precise definition of resilience. In a DCSP, a system is represented as a sequence of constraint satisfaction problems (CSPs) [Mackworth 92] where a set of variables that correspond to the system's state must satisfy a constraint. Adopting the framework of DCSPs into the problem of resilience gives us the following benefits. First, we can represent an open system consisting of a dynamically changing set of members by adding or removing system variables over time. Second, we can express its changing environment or context by modifying the current constraint accordingly. Third, we can precisely specify which system states are desirable by defining a constraint over the system variables.

3·1 System Snapshot

A system snapshot is essentially a constraint satisfaction problem (CSP), which is similar to a set of state variables in a state transition model, but the former is a more general concept than the latter; that is, a set of variables and their domains dynamically change over time. Also, each snapshot is associated with a constraint on its variables. We formalize the notion of a time-dependent system snapshot below.

Definition 1 (System Snapshot)**.** *Let N^t be the number of variables in a system at time t. A system snapshot at time t is a triple $S^t \equiv (X^t, D^t, C^t)$ where $X^t \equiv < X_1^t, X_2^t, \ldots, X_{N^t}^t >$ is a set of variables, $D^t \equiv < D_1^t, D_2^t, \ldots, D_{N^t}^t >$ is a set of finite domains, and a constraint C^t is a subset of the Cartesian product $D_1^t \times D_2^t \times \ldots \times D_{N^t}^t$.*

Definition 2 (Configuration)**.** *Each variable X_i^t takes a value x_i^t from a finite domain D_i^t; that is, $x_i^t \in D_i^t$ for all i. We say that a set of value assignment $\mathbf{x^t} \equiv < x_1^t, x_2^t, ..., x_{N^t}^t >$ is a* configuration *of the system at time t.*

Example 1: We will use the hypothetical spacecraft system below as a running example in this paper. The system consists of a fixed set of N components, each of which has a single variable X_i. The domain of each variable X_i is $\{Green, Red\}$, and the constraint at every time

t is $C = \{Green\} \times \cdots \times \{Green\}$. That is,

$$S^t \equiv (< X_1, X_2, \ldots, X_N >, \\ < \{Green, Red\}, \{Green, Red\}, \ldots, \{Green, Red\} >, \\ \{Green\} \times \{Green\} \times \ldots \times \{Green\}).$$

We introduce the notion of *fitness* to determine whether a given configuration is desirable or not.

Definition 3 (Fitness)**.** *We say that a configuration* $\mathbf{x^t} =< x_1^t, x_2^t, ..., x_{N^t}^t >$ *is* fit *with respect to constraint* C^t *if and only if* $\mathbf{x^t} \in C^t$.

Note that a constraint C is usually expressed by a set of boolean predicates that are defined on the Cartesian product $D_1^t \times D_2^t \times \ldots \times D_{N^t}^t$.

Example 2: A system configuration of the spacecraft in Example 1 is fit if all N components are good (i.e., $X_i = Green$ for all i).

3·2 Transitional Constraints

We call a time sequence of system snapshots a dynamic system, which can be considered as a DCSP.

Definition 4 (Dynamic System)**.** *We call a time series of system snapshots* $< S^1, S^2, \ldots, S^t, \ldots >$ *a* dynamic system.

A dynamic system is not an arbitrary collection of system snapshots, but is to represent a changing nature of the same identity following a certain law of causality. We thus need a way to define a class of dynamic systems feasible in a realistic setting, and introduce the notion of *transitional* constraints that restrict transitions among snapshots. A transitional constraint allows us to represent laws of causality in various scientific disciplines and to specify consequences of events from an environment.

A transitional constraint TC puts constraints concerning a transition from system S^i to S^{i+1} as follows:

Definition 5 (Transitional Constraint TC)**.** *Let* $\mathcal{S}$ *be the domain of all possible system snapshots. A transitional constraint* $TC : \mathcal{S} \rightarrow \mathcal{S}$ *is a* nondeterministic *algorithm that takes a system snapshot* S *as an input and output another snapshot* S' *such that, for every dynamic system* $< S^1, S^2, \ldots, S^t, \ldots >$, $S^{t+1} = TC(S^t)$ *possibly holds for all* t.

Although a transitional constraint TC is defined as a general function above, we expect that many of the transitional constraints of realistic systems take a simple form, which can be expressed in a declarative definition language.

Example 3: Suppose that the spacecraft in Example 2 is occasionally hit by space debris and has at most k component failures. We express such accidental component failures caused by the external events using the transitional constraint TC below.

$$TC(X^t, D^t, C^t) = \begin{cases} (X^t, D_{all}, C^t) \\ (X^t, D_{fail}, C^t) \end{cases}$$

D_{all} is a set of domains with every $D_i \in D_{all}$ is $\{Green, Red\}$, and D_{fail} is a set of domains where at most k domains in D_{fail} has domain $\{Red\}$. Notice that TC does not change a set of variables X^t and a constraint C^t for time $t+1$. The first case covers the situation where there is no component failure at time t, and the second case covers the situation where at most k component failures occur at time t. In the second case, TC shrinks the domains of at most k variables such that only Red value can be assigned to those variables.

In Definition 5, we assume that a system holds the Markov property where a set of possible successive system snapshots are determined solely based on the current snapshot of the system. We can generalize the definition where the function TC considers a historical sequence of the system to specify constraints on the transition to a next step.

Example 4: Suppose that if the spacecraft has component failures at time t, it will not have another component failure until time $t + k$. We can express such a historical constraint by defining a new TC, which takes k previous snapshots as inputs.

Different classes of transitional constraints constitute a taxonomy of dynamic systems. A few examples of our interests are as follows:

(1) **Static systems**
$X^t = X^{t+1}, D^t = D^{t+1}, C^t = C^{t+1}$ for all t. In this case, the system continues to have the same snapshot over time. Thus once the system finds a fit configuration $\mathbf{x^t}$, it can stay that way indefinitely.

(2) **Changing domains**
$X^t = X^{t+1}, C^t = C^{t+1}$ for all t. The variables and the constraint do not change, but the domains of the variables change over time. Suppose that we keep track of a position of a train i with a single positional variable X_i. When an accident occurs at a railway crossing, we might need to define a smaller domain for variable X_i to represent the fact that the train cannot pass the point of the accident. Another example in this category is the spacecraft system in Example 3.

(3) **Changing constraints**
$X^t = X^{t+1}, D^t = D^{t+1}$ for all t. The variables and their domains do not change, but the constraint changes over time. For example, we might say that every building on earth is fit if it is above sea level. If the current sea level rises due to global warming in the

future, a system modeling buildings will change the constraint about the locations of those buildings.

(4) **Unconstrained**
All variables, their domains, and constraints change over time. This class considers an open system where its system components dynamically join or leave the system. We expect to use the most general model to study an ecosystem consisting of various living organisms with different life cycles.

3·3 Adaptation Strategy

Although a transitional constraint TC determines the domain of each variable in the current snapshot, the system still has freedom of choosing the best configuration to adapt to the changing environment. We represent actions taken by system operators or the system itself as an adaptation strategy AS that determines the next configuration $\mathbf{x^{t+1}}$ from the domains of the variables in snapshot S^{t+1}. We define such an adaptation strategy as the function AS below.

Definition 6 (Adaptation Strategy AS)**.** *An adaptation strategy is defined as the function AS that takes a system snapshot S^{t+1} at time $t+1$ and the previous configuration $\mathbf{x^t}$ at time t as inputs and determines the configuration $\mathbf{x^{t+1}}$ at time $t+1$ as follows:*

$$\mathbf{x^{t+1}} = AS(S^{t+1}, \mathbf{x^t}).$$

The above definition implies that a system might consider the previous configuration to determine the next one. We could support a more general and probably more powerful adaptation strategy that considers previous system sequence $< S^1, S^2, \ldots, S^t >$ and the configuration sequence $< \mathbf{x^1}, \mathbf{x^2}, ..., \mathbf{x^t} >$. Also the system might be able to incorporate some knowledge regarding the future environments by taking the transitional constraint TC as an input of its adaptation strategy.

Example 5: Suppose that the spacecraft system adopts an adaption strategy of fixing one component a day (i.e., at each time t). This adaptation strategy is algorithmically defined as follows:

(1) Pick a variable X_i^t whose value is Red.
(2) Assign $X_i^{t+1} := Green$.
(3) For all $j \neq i$, assign $X_j^{t+1} := X_j^t$.

4. Definition of Resilience

Now we are ready to define the resilience of a dynamic system in Chapter 3. We define the resilience by formalizing Bruneau's definition in Chapter 2 with the framework of DCSPs.

In Section 4·1, we define the resilience of a dynamic system whose service quality in each snapshot is binary (*fit* or *unfit*). This definition considers a recovery time after the system becomes an undesirable state; it captures the recoverability of the system. In Section 4·2, we consider the resistance aspect of a system by measuring the degradation of a system's service quality as the number of unsatisfied conditions in the constraint. We next obtain a general definition of resilience combining those two definitions.

4·1 Recoverability

We first consider a particular sequence of configurations in a dynamic system and define the notion of k-Recoverability.

Definition 7 (k-Recoverable)**.** *Let $\mathbf{S} \equiv < S^1, S^2, \ldots, S^t, \ldots >$ be a dynamic system where $S^t \equiv (X^t, D^t, C^t)$. Given a constant k, a pair of a dynamic system $\mathbf{S}$ and a configuration sequence $< \mathbf{x^1}, \mathbf{x^2}, \ldots >$ is said to be* k-Recoverable *if, for every time t, a subsequence $< \mathbf{x^t}, \mathbf{x^{t+1}}, ..., \mathbf{x^{t+k-1}} >$ of length k contains a* fit *configuration $\mathbf{x^j} \in C^j$ where $t \leq j < t+k$.*

Alternately, if a dynamic system is k-Recoverable, its configuration sequence does not contain any subsequence of length k whose configurations are *all* unfit.

Example 6: Suppose that the spacecraft had k failed components at time t. If we use the adaptation strategy in Example 5 of fixing a failed component at each time, the spacecraft can resume the fit state from component failures within k days. Therefore, this spacecraft is k-Recoverable.

Given a non-deterministic transitional constraint TC and an adaptation strategy AS, we can enumerate all the possible configuration sequences from every possible pair of an initial system snapshot S^1 and an initial configuration $\mathbf{x^1}$. We can thus define the notion of recoverability for a class of dynamic systems by generalizing k-Recoverability in Definition 7.

Definition 8 (k-ClassRecoverable)**.** *Given a constant k and a transitional constraint TC, we say that an adaptation strategy AS is* k-ClassRecoverable *if, for every pair $(S^1, \mathbf{x^1})$ of initial system snapshot and configuration, every pair of a dynamic system $\mathbf{S} \equiv < S^1, S^2, \ldots >$ and a configuration sequence $< \mathbf{x^1}, \mathbf{x^2}, \ldots >$ where*

$$S^{t+1} = TC(S^t) \text{ and } \mathbf{x^{t+1}} = AS(S^{t+1}, \mathbf{x^t})$$

is k-Recoverable in Definition 7.

Informally, k-ClassRecoverability requires that every possible configuration sequence produced by a transitional constraint TC and an adaptation strategy AS is k-Recoverable.

We might want to relax the above definition and the following definitions in this Chapter by considering only the *fit* initial configurations where $\mathbf{x^1} \in \mathbf{C^1}$.

Example 7: We assume in Example 3 that the spacecraft has at most k component failures. Also, we assume in Example 4 that the spacecraft does not have an occurrence of component failures more than once in the time frame of k days. If we use the adaptation strategy in Example 5 of fixing a failed component at each time, the spacecraft can resume the fit state from any component failures within k days since there is no further disruption during that period. Therefore, this spacecraft is k-ClassRecoverable.

4·2 Resistance

We assume that every constraint C^t can be expressed as the intersection of multiple constraints $C^t_1, \ldots, C^t_{M^t}$ for some M^t; i.e.,

$$C^t = \bigcap_{i=1}^{M_t} C^t_i.$$

Let $SET(C^t)$ be the set $\{C^t_1, \ldots, C^t_{M^t}\}$. Then, we can quantify the service degradation level of a given configuration by computing the weighted sum of unsatisfied constraints in $SET(C^t)$. Informally, the resistance property of a dynamic system ensures that the service degradation level of every possible configuration is less than a given threshold l.

We define the weight function W, which assigns a non-negative integer to each constraint element C^t_i at time t. Each weight value represents a penalty corresponding to the degradation level of the service due to a failure to satisfy the constraint.

Definition 9 (Weight function W)**.** *Let $\mathcal{C}$ and T be the sets of all possible constraints and timestamps respectively. The weight function $W : \mathcal{C} \times T \rightarrow \mathcal{N}$ takes a constraint element C^t_i and time t as inputs and outputs a non-negative integer called a weight.*

The weighted loss of a given configuration $\mathbf{x^t}$ with respect to a set of constraint elements $SET(C^t)$ is defined as follows:

Definition 10 (Weighted loss)**.** *The Weighted loss function WL takes a given set of constraint elements $SET(C^t)$ and a configuration $\mathbf{x^t}$ as inputs and outputs the weighted sum as follows:*

$$WL(SET(C^t), \mathbf{x^t}) = \sum_{C^t_i \in SET(C^t) \wedge \mathbf{x^t} \notin C^t_i} W(C^t_i, t).$$

The weighted loss corresponds to the degradation of the service quality of a system at time t. We are now ready to define the resistance property of a dynamic system.

Definition 11 (l-Resistance)**.** *Let $\mathbf{S} \equiv < S^1, S^2, \ldots, S^t, \ldots >$ be a dynamic system where $S^t \equiv < X^t, D^t, C^t >$. Given a service degradation level l, a pair of a dynamic system $\mathbf{S}$ and a configuration sequence $< \mathbf{x^1}, \mathbf{x^2}, \ldots >$ is said to be* l-Resistant *if, for every time t, the weighted loss $WL(SET(C^t), \mathbf{x^t})$ is less than a given service degradation level l.*

Similarly, we can extend the notion of l-Resistance to a class of dynamic systems.

Definition 12 (l-ClassResistance)**.** *Given a constant l and a transitional constraint TC, we say that an adaptation strategy AS is* l-ClassResistant *if, for every pair $(S^1, \mathbf{x^1})$ of initial system snapshot and configuration, a pair of a dynamic system $\mathbf{S}$ and a configuration sequence $< \mathbf{x^1}, \mathbf{x^2}, \ldots >$ where*

$$S^{t+1} = TC(S^t) \text{ and } \mathbf{x^{t+1}} = AS(S^{t+1}, \mathbf{x^t})$$

is l-Resistant in Definition 12.

Example 8: The constraint $C = \{Green\} \times \cdots \times \{Green\}$ of the spacecraft can be represented as $\bigcap_{i=1}^{N} C_i$ such that

$$C_i = \{\mathbf{x} \mid \mathbf{x} \in \prod_{i=1}^{N} \{Green, Red\} \wedge x_i = Green\}.$$

We here assume the weight loss function WL simply outputs the number of unsatisfied constraint elements. Since the spacecraft violates at most k constraint element of the form C_i, we say that the spacecraft is l-ClassResistant.

We finally integrate the two properties concerning resilience and obtain the integrated resilience definition below. We first define the function that computes an accumulated service degradation level, which roughly corresponds to the triangle area in Figure 1.

Definition 13 (Accumulated loss AL)**.** *We define the accumulated loss function AL that takes a dynamic system $\mathbf{S} \equiv < S^1, S^2, \ldots >$, a sequence of configurations $\hat{\mathbf{x}} \equiv < \mathbf{x^1}, \mathbf{x^2}, \ldots >$ and a given time t_k and computes the accumulated weighted loss as follows:*

$$AL(\mathbf{S}, \hat{\mathbf{x}}, t_k) = \sum_{t=t_k}^{t_l} WL(SET(C^t), \mathbf{x^t}),$$

where t_l is the first occurrence of a fit configuration after time t_k; that is, $(\forall t_k < t < t_l : \mathbf{x^t} \notin C^t) \wedge x^l \in C^l$.

Proposition 1. *If we define the degradation of the service quality of a system as a weighted loss in Definition 10, the accumulated loss function AL in Definition 13 computes the area of Breneau's resilience triangle in Figure 1.*

Proof. Since we consider a discrete-time dynamic system, the area of Breneau's triangle is approximately computed

as follows:

$$\begin{aligned}\int_{t_0}^{t_1}[100-Q(t)]dt = & \sum_{t=t_0}^{t_1}(100-Q(t)) \\ = & \sum_{t=t_0}^{t_1} WL(SET(C^t),\mathbf{x^t}) \\ = & AL(\mathbf{S},\hat{\mathbf{x}},t_0).\end{aligned}$$

Note that time t_1 in Figure 1 corresponds to time t_l in Definition 13. □

We now give the definition of resilience, which specifies a quantitative upper bound on the size of Breneau's triangle.

Definition 14 (s-Resilience)**.** *Given a constant s, a pair of a dynamic system $\mathbf{S} \equiv < S^1, S^2, \ldots >$ and a sequence of configurations $\hat{\mathbf{x}} \equiv < \mathbf{x^1}, \mathbf{x^2}, \ldots >$ is said to be* s-Resilient *if, for every time t, the accumulated loss $AL(\mathbf{S},\hat{\mathbf{x}},t)$ is less than s.*

Definition 15 (s-ClassResilience)**.** *Given a constant s and a transitional constraint TC, we say that an adaptation strategy AS is* s-ClassResilient *if, for every pair $(S^1,\mathbf{x^1})$ of initial system snapshot and configuration, a pair of a dynamic system $\mathbf{S} \equiv < S^1, S^2, \ldots >$ and a configuration sequence $< \mathbf{x^1}, \mathbf{x^2}, \ldots >$ where*

$$S^{t+1} = TC(S^t) \text{ and } \mathbf{x^{t+1}} = AS(S^{t+1},\mathbf{x^t})$$

is s-Resilient in Definition 14.

Example 9: We consider the worst case where the spacecraft system had k component failures at time t_0. The adaptation strategy fixes one component at each time. Therefore, the number of components at time t_0+i is $(k-i)$, and the accumulated weighed loss is thus computed as

$$\sum_{t=t_0}^{t_0+k}[k-(t-t_0)] = k(k+1)/2.$$

Therefore, the spacecraft system is $k(k+1)/2$-ClassResilient.

5. Discussion

We discuss open research issues and possible research directions based on the formal framework of resilience in Chapter 4.

5·1 Efficient verification of s-resilience

One of the most important research issues in systems resilience is to develop an efficient algorithm for finding s-resilient adaptation strategy for a given dynamic system. We can, off course, naively find all sequences of s-resilient configurations by solving every possible system snapshot independently. Since such a brute-force approach is extremely inefficient, we need to pursue more efficient solutions to find a sequence of s-resilient configurations. Researchers in the field of DCSPs [Verfaillie 05] have been studying *reactive* techniques for efficiently solving a new CSP by reusing the previous solutions. However, since we can predict possible future changes of constraints to some extent by examining the transitional constraint, a *proactive* approach of finding a robust solution that remains valid under the presence of changing constraints is more promising. We plan to develop a new proactive technique for finding a *weakly* robust configuration such that subsequent configurations will have an accumulated loss under a given threshold s.

5·2 Uncertainty and incompleteness of system states

Our system model in Chapter 3 assumes that we have full knowledge about a set of variables, the value assignments to those variables (i.e., a configuration), and a transitional constraint at each time t. However, in the real world, a system often has only a partial and/or inaccurate knowledge about them. So we may need to introduce probabilistic mechanisms into the model, i.e., the variable values x_i^t could have a probabilistic distribution, the constraint C^t could be a probabilistic distribution, an adaptation strategy AS be a non-deterministic (or probabilistic) function, etc. We, therefore, believe that statistical time-series modeling based on the generalized state model [Kitagawa 87, Kitagawa 91] is an important technique for estimating a system model of a large system with many hidden parameters.We also expect that robustness and sensitivity analysis of models of dynamic systems, such as Bayesian networks [Chan 05], would be important research topics in this direction. Another important research topic is the revision of uncertain beliefs among entities (i.e., agents) to update the model based on probabilistic estimates of certain events [Chan 09].

Also, many large-scale systems consist of many entities (i.e., agents), each of which only has a partial knowledge about the whole system. A distributed constraint satisfaction problem (DisCSP) [Yokoo 00], which is an extended version of CSP where variables and constraints are distributed among multiple agents, addresses such issues in decentralized environments. We plan to employ DisCSP techniques in our research. Especially, we consider that faster approximate methods/algorithms are necessary for large-scale applications, since DisCSPs are NP-complete in general.

5·3 Utility function as the metrics of service quality

In Section 4·2, we consider the number of unsatisfied constraint elements to define the service quality. However, it might not be easy to define the total service quality of a system with the *reductionist* approach of considering each constraint element separately.

Another promising way to measure the quality of services as the output of a given utility function that takes multiple variables as inputs. Such utility functions have been widely used in decision theories in artificial intelligence and economics. We could use an output of a utility function as the metrics of service quality and define the resistance property of a dynamic system as a lower bound of the outputs of a utility function.

Taking this alternate approach, we can formulate l-resistance with the framework of constraint *optimization* problems (COPs) [Dechter 03, Schiex 95] rather than constraint *satisfaction* problems (CSPs). A COP is a problem to find an assignment of values to variables that maximize the output of a given utility function while satisfying all constraints.

Furthermore, many real world problems involve multiple criteria that should be considered separately and optimized simultaneously. We will consider to extend our system model to support a distributed constraint optimization problem (DCOP) [Modi 05, Petcu 05], which is an extended version of COP where variables and constraints are distributed among multiple agents.

6. Related work

Barber and Salido [Barber 11] introduce the notions of robustness, stability, recoverability, and reliability in DCSPs. Although the above notions are related to the concepts in this paper, there are a few fundamental differences as follows: First, they are only concerned with the case where a certain solution (i.e., a configuration) for a given CSP satisfies all the constraints. In their paper, robust solutions refer to solutions that remain valid after constraints are changed in the subsequent CSPs of the DCP, and stable solutions refer to solutions that are guaranteed to produce a new valid solution with only few assignment changes. Thus, those concepts do not consider the case where a given set of constraints are *partially* satisfied, as we define the notion of resistance in this paper.

Second, they define the notions of recoverability and reliability in the context of Dynamic Temporal Constraint Satisfaction Problems (DTCSPs) where system variables take temporal values on a certain time line. Thus, their definitions are orthogonal to our definition of recoverability; that is, the recoverability in their paper means that, even if there is some event at time t, the variables whose values are greater than $t + \delta$ can keep the same values to satisfy all the constraints. Finally, they define the above four concepts as the properties of *CSP solutions* in DCSPs. On the other hand, we formalize the notions in this paper as the *system* properties based on the system model in Chapter 3.

Baral et al. propose the concept of k-maintainability [Baral 08], which is similar to k-recoverability in Section 4·1. They model a system as a state transition machine in which some of the states are labeled as "normal." K-maintainability ensures that a system can come back to a normal state within k steps from any state in the system when there is no disruption from an external environment. They also develop a polynomial-time algorithm for finding a control strategy for achieving k-maintainability.

However, our system model based on DCSP is more general than that of k-maintainability in a number of ways. First, we specify whether a system snapshot is fit in a flexible way by giving a set of constraint elements. Second, since the domains of variables representing a system state are dynamic, we can much more easily model an open system with a dynamically changing set of members. Third, our system model can naturally quantify the quality of service at each time point by counting the number of satisfied constraint elements. On the other hand, there is no obvious way to introduce the metrics of service quality in Baral's state transition model.

Recovery, which is usually an expensive strategy for physical systems, has been studied in computer systems since rebooting a computer system is a relatively cheap process. Patterson [Patterson 02] introduces the paradigm of Recovery-oriented computing (ROC) whose focus is Mean Time to Repair of a system rather than Mean Time to Failure. The recoverability of software-based systems have been studied in the context of self-healing systems [Ghosh 07], in which many researchers proposed various recovery techniques such as replicating components, feedback control loops, and so on. However, the main focus in this area is to develop concrete strategies for system recovery rather than to formalize the notion of resilience based on a general system model.

7. Conclusion

We set out to establish a new research discipline of resilience engineering. In this paper, we formalize the notion of resilience, which considers both the resistance and recovery aspects of dynamic systems. We obtain the def-

inition of resilience by modeling a dynamic system based on the framework of dynamic constraint satisfaction problems (DCSPs). Our resilience definitions give us precise metrics for the resistance and recoverability of dynamic systems and combine those two properties into the single property, which corresponds to Bruneau's informal but widely accepted resilience definition.

We plan to evaluate the validity of our resilience system model by applying it to realistic systems in various domains such as evolutionary biology, social science, and so on.

Acknowledgments

This research is supported by grant for the Systems Resilience project from the Transdisciplinary Research Integration Center of Research Organization of information and Systems in Japan.

♢ References ♢

[Baral 08] Baral, C., Eiter, T., Bjäreland, M., and Nakamura, M.: Maintenance goals of agents in a dynamic environment: Formulation and policy construction, *Artificial Intelligence*, Vol. 172, No. 12-13, pp. 1429–1469 (2008)

[Barber 11] Barber, M., Federico ; Salido: Robustness, Stability, Recoverability and Reliability in Dynamic Constraint Satisfaction Problems, Technical Report DSIC-IA-PS:1, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia (2011)

[Bruneau 03] Bruneau, M.: A Framework to Quantitatively Assess and Enhance the Seismic Resilience of Communities, *Earthquake Spectra*, Vol. 19, No. 4 (2003)

[Chan 05] Chan, H. and Darwiche, A.: On the revision of probabilistic beliefs using uncertain evidence, *Artificial Intelligence*, Vol. 163, No. 1, pp. 67–90 (2005)

[Chan 09] Chan, H.: *Sensitivity Analysis of Probabilistic Graphical Models: Theoretical Results and Their Applications on Bayesian Network Modeling and Inference*, VDM Verlag, Saarbrücken, Germany, Germany (2009)

[Dechter 03] Dechter, R.: *Constraint Processing*, Morgan Kaufmann Publishers (2003)

[Faltings 05] Faltings, B. and Macho-Gonzalez, S.: Open constraint programming, *Artificial Intelligence*, Vol. 161, pp. 181–208 (2005)

[Ghosh 07] Ghosh, D., Sharman, R., Raghav Rao, H., and Upadhyaya, S.: Self-healing systems - survey and synthesis, *Decision Support Systems*, Vol. 42, No. 4, pp. 2164–2185 (2007)

[Holling 73] Holling, C.: Resilience and Stability of Ecological Systems, *Annual Review of Ecology and Systematics*, Vol. 4, pp. 1–23 (1973)

[Kitagawa 87] Kitagawa, G.: Non-Gaussian state-space modeling of non-stationary time series (with discussion), *Journal of the American Statistical Association*, Vol. 82, pp. 1032–1063 (1987)

[Kitagawa 91] Kitagawa, G.: A nonlinear smoothing method for time series analysis, *Statistica Sinica*, Vol. 1, No. 2, pp. 371–388 (1991)

[Mackworth 92] Mackworth, A.: Constraint Satisfaction, in *Encyclopedia of Artificial Intelligence*, pp. 285–293 (1992)

[Maruyama 12] Maruyama, H., Inoue, K., Tsubaki, H., Akashi, H., Okada, H., and Minami, K.: Systems Resilience, in *Forum on Information Technology*, Information and Systems Society (2012)

[Modi 05] Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M.: ADOPT: asynchronous distributed constraint optimization with quality guarantees, *Artif. Intell.*, Vol. 161, No. 1-2, pp. 149–180 (2005)

[Patterson 02] Patterson, D.: Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, Technical report, Berkeley, CA, USA (2002)

[Petcu 05] Petcu, A. and Faltings, B.: A Scalable Method for Multiagent Constraint Optimization, in *IJCAI*, pp. 266–271 (2005)

[Schiex 95] Schiex, T., Fargier, H., and Verfaillie, G.: Valued constraint satisfaction problems: Hard and easy problems, in *IJCAI*, pp. 631–639 (1995)

[Verfaillie 05] Verfaillie, G. and Jussien, N.: Constraint Solving in Uncertain and Dynamic Environments: A Survey, *Constraints*, Vol. 10, No. 3, pp. 253–281 (2005)

[Yokoo 00] Yokoo, M. and Hirayama, K.: Algorithms for Distributed Constraint Satisfaction: A Review, *Journal of Autonomous Agents and Multi-agent Systems*, Vol. 3, No. 2, pp. 189–211 (2000)