Robustness, Stability, Recoverability and Reliability in Dynamic Constraint Satisfaction Problems

Federico Barber and Miguel A. Salido

Instituto de Automática e Informática Industrial
Universidad Politecnica de Valencia
email {fbarber,msalido}@dsic.upv.es

Abstract. [1] Many real-world problems in Artificial Intelligence (AI) as well as in other areas of computer science and engineering can be efficiently modeled and solved using constraint programming techniques. In many real-world scenarios the problem is partially known, imprecise and dynamic, so that some effects of actions are undesired and/or several un-foreseen incidences or changes can occur. Whereas expressivity, efficiency, and optimality have been the typical goals in the area, several is-sues regarding robustness appear with a clear relevance in dynamic constraint satisfaction problems (DCSPs). However, there is still no a clear and common definition of robustness-related concepts in CSPs. In this paper, we propose two clearly differentiated definitions for *robustness* and *stability* in CSP solutions. We also introduce the concepts of *recoverability* and *reliability* which arise in temporal CSPs. All these definitions are based on related well-known concepts addressed in engineering and other related areas.

1 Introduction

Nowadays many real problems can be modeled as constraint satisfaction problems (CSP) and are solved using constraint programming techniques. Much effort has been spent to increase the efficiency of the constraint satisfaction algorithms: filtering, learning and distributed techniques, improved backtracking, use of efficient representations and heuristics, etc. This effort resulted in the design of constraint reasoning tools which were used to solve numerous real problems. However, all these techniques assume that the set of variables and constraints, which compose the CSP, is completely known and fixed. This is a strong limitation when dealing with real situations where the CSP under consideration may evolve because of (i) changes in the environment or in their execution conditions, (ii) evolution of user requirements in the framework of an interactive design, and (iii) changes in other agents in the framework of a distributed system [13].

Real world is dynamic in its nature, so techniques attempting to model it should take this dynamicity in consideration. A Dynamic Constraint Satisfaction Problem (DCSP) [2] is an extension to a static CSP that models addition and retraction of constraints and hence it is more appropriate for handling dynamic real-world problems. It is indeed easy to see that all possible changes to a CSP (constraint or domain modifications, variable additions or removals) can be expressed in terms of constraint additions or removals [13].

[1] Riunet digital repository: http://riunet.upv.es/handle/10251/10705

Several proactive or reactive techniques have been developed to manage incidences in dynamic problems. In this context, computing a new solution from scratch after each problem change is possible (reactive technique), but it has two important drawbacks: inefficiency and instability of the successive solutions [13].

In [14], a proactive approach is presented to explore methods for finding solutions that are more likely to remain valid after changes that temporarily alter the set of valid assignments. Other works are focused on searching for a new solution that minimizes the number of changes from the original solution. For instance, in [5] the concept of super-solution is introduced for constraint programming: 'A solution is a super solution if it is possible to repair the solution with only a few changes'. This is a generalization of both fault tolerance solutions in constraint programming [16] and supermodels in propositional satisfiability [9]. In [12], a method is proposed for reusing the previous solution to produce a new solution, by means of local changes on the previous one.

By reading the research carried out in dynamic constraint satisfaction, we found a clear misunderstanding regarding the concepts of robustness and stability. Some authors refer to *robust solutions* in the same meaning than others refer to *stable solutions*. For instance, one of the most recent papers regarding dynamic constraint satisfaction [15] states that the strategies that have been devised to handle DCSPs are: "methods for finding "robust" solutions that are either more likely to remain solutions after change or are guaranteed to produce a valid solution to the altered problem with a fixed number of assignment changes."

In engineering, there is an agreement to distinguish between stable and robust concepts. Thus, what's the difference between stable and robust CSP solutions? Answering this question is not always easy by the fact that robustness has multiple, sometimes conflicting, interpretations [6]. Only some works make a tiny distinction between robustness and stability in constraint satisfaction [4]. Under our consideration, both robustness and stability terms are being mixed. Robust solutions refer to solutions that are either more likely to remain valid after change meanwhile stable solutions are solutions that are guaranteed to produce a new valid solution with only few assignment changes.

In this paper we focus our attention on the 'robustness' and 'stability' concepts in CSPs. We propose general engineering-based and clearly different definitions for robust and stable CSP solutions. Moreover, we also introduce the concepts of 'recoverability' and 'reliability' which are relevant in real-world temporal-CSP domains. Clear and common definitions is needed for evaluating different alternatives. Afterwards, new questions will appear: How many definitions related to robustness and stability can be identified? How can we assess robustness or stability of a solution? What does it guaranties? How can we get a more robust solution? What is the relationship between robustness and other problem parameters, like optimality and constrainedness? Is it possible to obtain a model of robustness?

1.1 Definitions

Following some standard notations and definitions in the literature, we have summarized the basic definitions that will be used along the paper.

Definition 1. A Constraint Satisfaction Problem (CSP) is a triple $P =< X, D, C >$ where, X is a finite set of variables $\{x_1, x_2, ..., x_n\}$, D is a set of domains $D =$

$\{d_1, d_2, ..., d_n\}$ such that each variable $x_i \in X$ has a finite set of possible values d_i. C is a finite set of constraints $C = \{C_1, C_2, ..., C_m\}$ which restrict the values that the variables can simultaneously take. C can be extensionally represented by a set of feasible tuples or intensionally represented by a set of logical-mathematical functions.

Definition 2. A dynamic constraint satisfaction problem (DCSP) is a a sequence of static CSPs, where each successive CSP is the result of changes in the preceding one [2]. In the original definition, changes could be due either to addition or removal of constraints.

Definition 3. The Solution Space is the portion of search space ($\prod_{i=1,n} d_i$) that satisfies all constraints. A solution S is an instantiation of all variables that satisfy all constraints.

2 Changes/Incidences in DCSPs

Many real problems are dynamic so unexpected changes/incidences in the environment or problem scenario occur due to its dynamism, spurious actions, lack of complete knowledge, etc. In its dynamic evolution, a DCSP should face to a set of incidents Z. Each change or incidence $z_i \in Z$ can be modeled as a set of changes in variable domains or constraints. Since changes in domains can be represented as new unary constraints, it can be assumed that each incidence z_i can be represented by (actually, it leads to) a new set of constraints Cz_i.

$$z_i \rightarrow Cz_i \tag{1}$$

Once the occurrence of each incidence z_i, the new set of constraints Cz_i must be added to the previous set of constrains (C) making the problem more restricted, or even inconsistent. Thus, the new problem after an incidence z_i becomes:

$$CSP_{z_i} =< X, D, C \oplus Cz_i > \tag{2}$$

We only consider incidences that add new constraints (which includes modification of the existing ones). Only the removal of constraints is not considered since it does not restrict the solution space, so that each solution remain feasible when some constraints are removed. Moreover, we assume the incidences only restrict (but do not make empty) the initial solution space; otherwise the problem would become inconsistent. Therefore, some of the feasible solutions of the initial CSP are also solutions of CSP_{z_i}. Let's also assume that a finite set of incidences (Z) is expected, each one $z_i \in Z$ with a probability $p(z_i)$ and an affectation degree $d(z_i)$.

- The function $p(z_i)$ introduces a probability distribution over Z, so that $p(z_i)$ describes the relative likelihood for z_i to occur. A function for each z_i can be possible in some problem domains, by using statistical or historical analysis, typology of expected incidences, etc. In other domains it could not be possible (uninformed DCSP) so we could assume a uniform probability of incidences.
- The function $d(z_i)$ is related to the restriction imposed by z_i on the initial solution space. Therefore, $d(z_i)$ can be evaluated as the proportion of tuples in C that

become unfeasible after z_i. The function $d(z_i)$ can be related to the concept of *sensitiveness* of the problem (the CSP) regarding the incidence z_i, that is, how much the problem is affected due to z_i

Obviously, it is not possible to concrete robustness-related features of a system if no information about the incidences is given. In this case, we can obtain a rough estimation by means of the inclusion of random incidences (i.e.: random values for $p(z_i)$, $d(z_i)$).

For simplicity reasons, we will generalize our notation and denote z_i as z and Cz_i as Cz.

3 Robustness

Robustness is a common feature in our environment. Biological life, functional systems, physical objects, etc [11], persists, remain running and maintain their main features despite continuous perturbations, changes, incidences or aggressions. Thus, robustness is a concept related to the *persistence* of the system, of its structure, of its functionality, etc., against external interference: *A system is robust, if it persists*.

Thus, in a general way, "*robustness*" can be defined as the ability of a system to withstand stresses, pressures, perturbations, unpredictable changes or variations in its operating environment without loss of functionality. A system designed to perform functionality in an expected environment is "*robust*" if it is able to maintain its functionality under a set of incidences. As a closer example, an algorithm is robust if it continues to operate despite unexpected inputs or erroneous calculations.

Intuitively, the notion of robustness is easy to define, but its formalization depends on the system, on its expected functionality and on the particular set of incidences to face up [10]. No general formal definition of robustness has been proposed, except few exceptions or particular cases. Particularly, Kitano [7] mathematically defines the robustness (R) of a system (SYS) with regard to function (F) against a set of perturbations (Z), as (in a simplified way):

$$R_{F,Z}^{SYS} = \int_Z p(z) * F(z) dz \tag{3}$$

Where, $p(z)$ is the probability for incidence $z \in Z$, and $F(z)$ is an evaluation function that returns zero when the system fails under z or it returns a relative viability $]0, 1]$ in other case. For instance, *if production drops 20% under a certain perturbation (z), compared with standard production, then 0.8 is returned*. The expression (1) formalizes how a system (SYS) is able to maintain a certain level of its expected functionality (F) against a given set of perturbations (Z). According to (1), a system SYS_1 is more robust than SYS_2 with regard to an expected functionality F against a set of perturbations Z, when:

$$R_{F,Z}^{SYS_1} > R_{F,Z}^{SYS_2} \tag{4}$$

The application of definitions of robustness is highly problem-dependent. Let's apply (1) to DCSPs:

– S is a solution of the DCSP, which we want to assess its robustness. Thus, S is the system (SYS). Robustness is a concept related to DCSP solutions, not to DCSP itself.
– Z is the discrete set of unexpected incidences (i.e.: changes in constraints).
– F is the expected functionality of the system. In DCSP, the expected functionality of a solution is its feasibility.

Therefore, the expression (3) becomes:

$$R^S_{F,Z} = \sum_Z p(z) * F(z) \tag{5}$$

Where function F is defined, in the case of a DCSP as:

– $F(z) = 1$ iff S also holds $C \oplus Cz$. Taking into account that S holds C (since S is a solution of the CSP), $F(z) = 1$, iff S also holds Cz.
– $F(z) = 0$, iff S does not hold $C \oplus Cz$. More concretely, iff S does not hold Cz.

A 1-robust solution is a solution that maintains its feasibility over the whole set of expected incidences.

Note that robustness does not require insensitiveness of the CSP. For instance, the constraints of the problem could dramatically vary due to z, so that Cz could greatly reduce the solution space. However, a robust solution S with respect z would remain feasible after the incidence.

Also note that robustness of a solution S does not depends on the behavior of S against an incidence z, but on how feasibility of S is maintained over a set of unexpected incidences Z. Thus, robustness of S depends on the probability $p(z)$ of each possible incidence $z \in Z$ and on how z affects to feasibility of the solution $F(z)$.

We remark that we do not take into account other aspects, which have been usually taken into account when robustness of a DCSP solution is assessed by other authors. For instance, the number of variables that must change their values to make the initial solution feasible after the incidence, the number of unsatisfied constraints by the initial solution, etc. In our approach, a solution is not more/less robust under a given incidence if the solution need be more/less repaired to deal with the incidence. We claim that robustness cannot be assessed on the basis that only small changes are necessary to obtain a new feasible solution. In problems related with satisfiability, robustness should be related to feasibility maintenance.

3.1 Example

Let's apply the above definition (5) on the following example. *P* be a CSP with two variables x_1 and x_2 with domains $D_1 : \{3..7\}$ and $D_2 : \{2..6\}$ respectively. The dynamic constraints are:

– $C_1 : x_1 + x_2 \leq 12$
– $C_2 : x_2 + x_1 \geq 6$
– $C_3 : x_2 - x_1 \leq 2$
– $C_4 : x_1 - x_2 \leq 4$

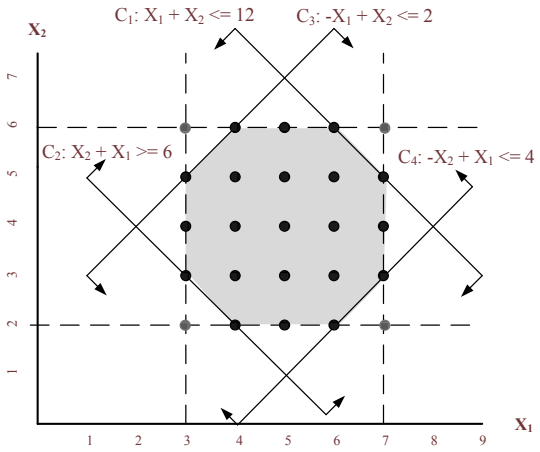

Fig. 1. Example of CSP P.

The Figure 1 represents the solution's space of the CSP, which is composed of 21 solutions.

Let's suppose the following set Z of expected incidences:

Incidence z_i	Likelihood $p(z_i)$	$z_i \rightarrow C_{z_i}$	$d(z_i)$
z_1	0.15	$\{x_1 + x_2 <= 9, x_2 <= 5\}$	$9/21$
z_2	0.1	$\{x_1 + x_2 >= 10, x_1 >= 4\}$	$12/21$
z_3	0.25	$\{-x_1 + x_2 <= 0\}$	$9/21$
z_4	0.3	$\{x_1 - x_2 <= 2\}$	$5/21$
z_5	0.2	$\{x_1 > 4\}$	$8/21$

The column related to $d(z_i)$ is included only for reasons of information, since the new constraints C_{z_i} corresponding to each z_i are known. Note that $\sum_Z p(z_i) = 1$.

The robustness of each solution can be assessed according 5. For instance, the solution $S = \{x_1 = 3, x_2 = 4\}$ is no longer valid when z_2, z_3 or z_5 occurs. Its robustness can be assessed as $R_Z^S = p(z_1) + p(z_4) = 0.15 + 0.3 = 0.45$.

From Table 1, we can deduce that $\{x_1 = 5, x_2 = 3\}$ and $\{x_1 = 5, x_2 = 4\}$ are the more robust solutions, according the above set of expected incidences. Likewise $\{x_1 = 4, x_2 = 6\}$ is the less robust solution.

Although the solution space of the above example is convex, note that it is not required for assessment robustness of CSP solutions, nor it is necessary an implicit representation of the CSP. Moreover, robustness of each solution can be assessed independently of assessment for other solutions

4 Stability

Stability is an old concept that derives from astronomy and physics [17]. Loosely speaking, a solution (meaning an equilibrium state) of a dynamical system is said to be stable

Table 1. Robustness of each solution

Solution (x_1 x_2)	holds z_1?	holds z_2?	holds z_3?	holds z_4?	holds z_5?	Robustness
4,2	y	n	y	y	n	0,7
5,2	y	n	y	n	y	0,6
6,2	y	n	y	n	y	0,6
3,3	y	n	y	y	n	0,7
4,3	y	n	y	y	n	0,7
5,3	***y***	***n***	***y***	***y***	***y***	***0,9***
6,3	y	n	y	n	y	0,6
7,3	n	y	y	n	y	0,55
3,4	y	n	n	y	n	0,45
4,4	y	n	y	y	n	0,7
5,4	***y***	***n***	***y***	***y***	***y***	***0,9***
6,4	n	y	y	y	y	0,85
7,4	n	y	y	n	y	0,55
3,5	y	n	n	y	n	0,45
4,5	y	n	n	y	n	0,45
5,5	n	y	y	y	y	0,85
6,5	n	y	y	y	y	0,85
7,5	n	y	y	y	y	0,85
4,6	***n***	***y***	***n***	***y***	***n***	***0,4***
5,6	n	y	n	y	y	0,6
6,6	n	y	y	y	y	0,85

if small perturbations to the solution result in a new solution that stays "*close*" to the original solution for all time. Perturbations can be viewed as small differences effected in the actual state of the system [6]. Therefore, by applying this informal definition to DCSPs, a solution is stable if small modifications of the constraint set require a new solution (new consistent variable assignment) that remains close to the original solution:

$$Sol(X, D, C) \text{ is stable (with respect } z, C \oplus z \cong C) \text{ iff}$$
$$\exists Sol(X, D, C \oplus z) : Sol(X, D, C) \cong Sol(X, D, C \oplus z)$$

Definition 4. A solution S of a DCSP is r-stable, with respect to to an incidence z, if a new feasible solution S exists in the r-neighborhood of S.

The neighborhood of solutions can be formally defined in terms of norms on n-dimensional spaces [3].

Definition 5. A solution $S = (x_1 = v_1, x_2 = v_2, , x_n = v_n)$ is *r-stable* if, given an incidence z, there is a solution $S' = (x_1 = v'_1, x_2 = v'_2, , x_n = v'_n)$, such that: $\|S' - S\| < r$, where $\|.\|$ is any n-dimensional norm on the difference of S and S'. In relation to the implementation of n-dimensional norms:

– On metric domains, we can apply the Euclidean distance between S and S', with a optional weighted factor $\rho : i$ for each x_i:

$$\|S' - S\|_z = \sqrt{\sum_{i=1}^{n} \rho_i (x'_i - x_i)^2} \tag{6}$$

Note that the similarity given by (6) between S and S' may be very low if the two solutions S and S' are very close in the n-dimensional space although all the variables of S change their values. This n-dimensional norm measures the amount of change of values in the variables that change.

– On non metric domains (like non-ordered sets of values) the Hamming distance (H) can be applied. This n-dimensional norm measures the number of variables that have different values in S and S'.

$$\|S' - S\|_z = \sum_{i=1}^{n} \rho_i H(x'_i, x_i) \tag{7}$$

where $H(x'_i, x_i)$ is equal 0 iff $x'_i = x_i$, and 1 otherwise. This criteria evaluates the number of variables that change their values (which is related with the super-solutions concept).

These measures evaluate the closeness of solutions in the state space. Therefore, given an incidence z, the r-stability for a solution S quantify the r-proximity to S of the closest feasible solution S' in the n-dimensional space of the DCSP. That is, how much the new solution S' differs from the initial one S in order to address the incidence. A robust solution is a 0-stable solution.

The proposed measures of r-stability require finding a solution in the closest neighborhood of S, among the complete set of new feasible solutions, so that deviation with respect to the previous solutions S is minimized. Let's denote $N(S, z)$ as the value of $\|S' - S\|$ for the closest solution S' to S, after the occurrence of z:

$$N(S, z) = min_{S'} \|S' - S\| \tag{8}$$

To obtain $N(S, z)$, it is needed to generate a Constraint Satisfaction and Optimization Problem (CSOP) which constrains are $C \oplus z$ and the optimality criteria is to minimize $\|S' - S\|$. The search space of the obtained CSOP can be reduced to the closest neighborhood of S so that the required computational effort becomes considerably reduced.

According the definition 5, we can define the stability (STA) of a solution (S) against a given set of perturbations (Z), as:

$$STA_Z^S = \sum_{Z} p(z) \cdot N(S, z) \cdot d(z) \tag{9}$$

In equation 9, $N(S, z)$ is ponderated by d(z) in order to normalize N(S,z) with respect to d(z).

4.1 Example

Let's apply the above definition of stability (9) on the previous example (Figure 1) for the most and less robust solutions given by Table 1.

Table 2. Stability of some robust solutions (5,3) and (5,4) and non-robust solution (4,6).

Solution (x_1, x_2)	Closest sol. with z_1	Closest sol. with z_2	Closest sol. with z_3	Closest sol. with z_4	Closest sol. with z_5	Robustness	Stability
(5,3)	***holds***	***(6,4)***	***holds***	***holds***	***holds***	***0,9***	*0.08-stable*
(5,4)	***holds***	***(6,4)***	***holds***	***holds***	***holds***	***0,9***	*0.06-stable*
(4,6)	***(4,5)***	***holds***	***(5,5)***	***holds***	***(5,6)***	***0,4***	*0.29-stable*

For instance, the stability of solution $(4, 6)$ is according to expression (9):

$$STA_Z^{(4,6)} = 0.15 * 1 * 9/21 + 0.25 * \sqrt{2} * 9/21 + 0.2 * 1 * 8/21 = 0.29 \qquad (10)$$

Thus, following Table 2, $\{x_1 = 5, x_2 = 4\}$ is the most robust (0.9) and the most stable solution (0.06) according the given set of expected incidences.

5 Temporal Constraint Satisfaction

In dynamic environments, variables of a DCSP usually have a temporal dimension in such a way the solution is projected over time. In this case, the problem is called Dynamic Temporal Constraint Satisfaction Problem (DTCSP). This is the typical case of scheduling problems, where variables can be associated to starting or ending times of tasks (see Figure 2). In these problems, not only is important to know how different is the new feasible solution S' from the original solution S, given an incidence z, but how long S' differs from S (recoverability) or how long S can be maintained (reliability) from the incidence. Thus, two new properties related to *temporal stability* appear: *recoverability* and *reliability*.

5.1 Recoverability

Recoverability refers to the ability to restore a system to the point at which a failure occurred. Despite of proactive approaches, it is clear that robustness is not always completely guarantied. Therefore, *recovery strategies* should be used when disturbing events occur, in order to keep the feasibility of the pre-computed solution. Robustness and recoverability are closely related and, in some optimization frameworks, they have been unified into an integrated notion of *recoverable robustness* [8]. Regarding DTCSP, where solutions project over time, the recoverability of a solution can be measured by the amount of time (Δt) required, after incidence occurs, to restore part of the initial solution. The temporal variables in a solution of a DTCSP are distributed over time. Thus, a *Δt-recovered* solution maintains the same initial values to variables related from Δt after incidence:

$$Sol_{\Delta t}(X, D, C \oplus z) \equiv Sol_{\Delta t}(X, D, C)$$

Where $Sol_{\Delta t}$ covers the set of DCSP variables from Δt after incidence. The objective of a recovery process is to minimize Δt. Moreover, since the variables of DTCSP are temporally ordered (i.e.: they are projected over time), the objective of a recovery process becomes minimize the set of variables (from the time t when incidence occurs until $t + \Delta t$) that require change their value in a new feasible solution. The initial solution S is recovered after the temporal interval $[t, t + \Delta t]$.

Definition 6. A solution S is *q-recoverable* iff, at most, q variable's assignments (consecutive variables after the incidence occurs) require change its value in the new feasible solution S'.

Thus, a solution $S = (x_1 = v_1, x_2 = v_2, , x_n = v_n)$ is q-recoverable iff given an incidence z that affects from x_t, $S' = (x_1 = v_1, x_2 = v_2, , x_t = v'_t, x_{t+1} = v'_{t+1}, , x_{t+q} = v'_{t+q}, x_{t+q+1} = v_{t+q+1}, , x_n = v_n), 1 \leq t \leq n, 1 \leq t + q \leq n$, is a solution of $DTCSP_{CH}$. The initial solution is recovered after $x_t + q$.

Note that the definition of q-recoverability is similar to the definition of $(q, 0)-$*super solutions* where if q variables lose their values, we can find another solution by reassigning these variables with a new value. The only difference is that in $q-$*recoverability*, the variables to be repaired are consecutive in time meanwhile in (q,0)-super solutions the variables to be repaired are not consecutive.

5.2 Reliability

In engineering, reliability is associated to the confidence that a system will perform its intended function during a specified period of time under stated conditions, as well as under unexpected circumstances. Mathematically it can be expressed as:

$$R(t) = \int_t^{\infty} f(x)dx \tag{11}$$

where $f(x)$ is the failure probability density function and t is the length of the period of time (which is assumed to start from time zero). There is always some chance for failure, but $R(t)$ means that the system has a specified chance that it will operate without failure before time t.

In DTCSP, variables of solution are distributed on time. Thus, a solution found initially may be invalid for variables which are related to a time greater than Δt after incidence. Thus, by applying the above concepts, we can assess that a DTCSP solution is Δt*-reliable*, if given an incidence at time t, the solution remains valid until $t + \Delta t$. Thus, the set of variables that represents the solution of the problem from time t until $t + \Delta t$ are not required to change their value:

$$Sol_{t \to \Delta t}(X, D, C \oplus z) \equiv Sol_{t \to \Delta t}(X, D, C)$$

Where $Sol_{\Delta t}$ covers the set of DCSP variables from time t until $t+\Delta t$.The objective for obtaining a reliable solution is maximizing t, or alternatively, maximizing the set of variables (from time t, when incidence occurs, until $t+\Delta t$) that can maintain their value. Thus, in a similar way that recoverability, reliability of a solution S can be defined in

terms of the number of assignments in S that remains valid, i.e: they can take part of a solution of $DTCSP_{CH}$.

Definition 7. A solution S is *u-reliable* if, at least, u variable's assignments in S (consecutive variables after the incidence occurs) can take part of a solution of the $DTCSP_{CH}$. Thus, a solution $S = (x_1 = v_1, x_2 = v_2, , x_n = v_n)$ is u-reliable, iff given an incidence z that affects from x_t, $S' = (x_1 = v_1, x_2 = v_2, , x_t = v'_t, x_{t+1} = v_{t+1}, , x_{t+u} = v_{t+u}, x_{t+u+1} = v'_{t+u+1}, , x_n = v'_n), 1 \leq t \leq n, 1 \leq t + u \leq n$, is a solution of $DTCSP_{CH}$. The initial solution is maintained until x_{t+u}.

6 Example: A Scheduling Problem

Figure 2 shows an example with different solutions to a scheduling problem with two jobs, each one with three activities, and one resource. Each row corresponds to a job, and an activity (x_{ij}) is represented as a rectangle whose length corresponds to its duration. This problem can be modeled as a TCSP, where variables represent time points (starting or ending times) of different activities ($x_{ij.on}$, $x_{ij.off}$). There exist constraints referring to non-overlap and precedence constrains among activities. Moreover, x_{23} should be performed unless k-units after x_{22} (Constraint C_{23-22}). The first solution (Figure 2a) minimizes the makespan and it is considered to be the optimal solution. Figure 2b shows a robust solution. To this end, some buffer times have been included between some activities in order to absorb incidences. For instance, if a resource is broken for a short time, (Incidence in Figure 2b), the solution is not affected by the incidence. Thus, all assignments to variables remain valid. Furthermore this solution is also stable. If variables $x_{21.off}, x_{22.off}, x_{12.off}$ or $x_{13.off}$ are tiny delayed, the rest of variables maintain the same values. It can be observed the typical trade-off between robustness and optimality in Figure 2a/b. Figure 2c shows a 3-recoverable solution for an incidence z: $x_{21.off}$ is delayed to $x'_{21.off}$ in time t. In this case, only 3 variables must change their values ($x_{21.off}, x_{22.on}, x_{22.off}$) meanwhile assigned variables with assigned values greater than $t + \Delta t$: ($x_{12.on}, x_{12.off}, x_{13.on}, x_{13.off}, x_{23.on}, x_{23.off}$) do not change their values. Figure 2d shows a 4-reliability solution for an incidence z: $x_{22.off}$ is delayed to $x'_{22.off}$ in time t. In this case, the next 4 variables ($x_{12.on}, x_{12.off}, x_{13.on}, x_{13.off}$) do not change their values. The solution is maintained until $t + \Delta t$. However, activity x_{23} must satisfy C_{23-22}, so that $x_{23.on}$ and further variables must change their values.

7 Generalizing the concepts

In the previous sections, the concepts of robustness, stability, recoverability and reliability have been defined by analyzing how a solution S absorbs or can be adapted to cope up an incidence z. These concepts can be generalized, such that we can assess the achievable levels of robustness, stability, recoverability and reliability of solutions of a DCSOP for a given typology of incidences $(Z, p(z), d(z))$, optimality levels, and the given constrainedness of the DCSOP which is inherent to the problem:

- Robustness guaranties that perturbations can be absorbed by the solution. Thus, robustness decreases as the degree of incidences increases.

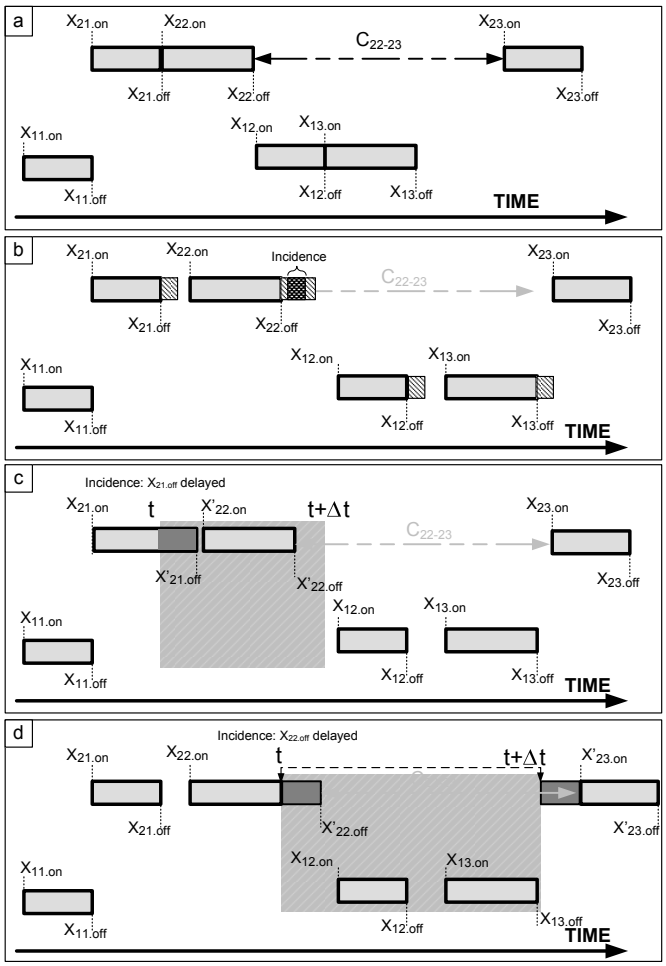

Fig. 2. A scheduling problem: four solutions.

- Stability guaranties that perturbations can be minimized by the solution. Thus, stability decreases as degree of the incidences increases.
- A low restricted DCSP, with large solution space, will usually allow more robust and stable solutions.
- A more optimized solution will usually be more sensible to changes in the environment. There exists a clear trade-off between robustness and optimality [1].

These ideas introduce the main concepts to which robustness, stability, recoverability and reliability of solutions in DCSOP can be related. These mutual relations are represented in Figure 3, which mutually relate robustness, stability, recoverability and reliability of solutions of DCSPs with: *(i)* the constrainedness of DCSPs (which is a problem-dependent feature), *(ii)* incidence's degree (which is a feature of the problem

and/or application scenario), and *(iii)* optimality of S (which is a feature of each solution).

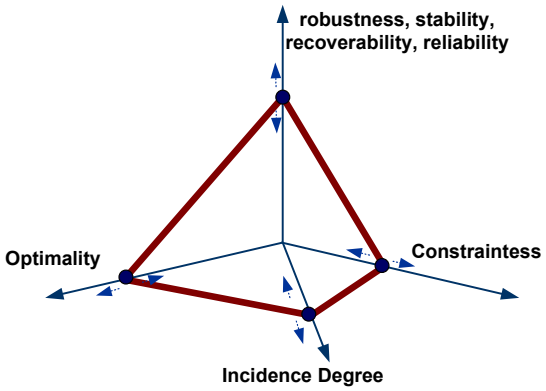

Fig. 3. A scheduling problem: four solutions.

From Figure 3, the goal would be to model and parameterize the above relations. This requires a clear definition of related concepts and methods for analytical or empirical assessment. The evaluation of robustness, stability, recoverability and reliability of a solution in a DCSP can be view as a guaranty of the behavior of S with respect to Z. Thus, some questions are straightforward: How can it be evaluated? What does it guarantee? Afterwards, methods for obtaining more robust, stable, recoverable and reliable solutions should be reached.

8 Conclusions

Whereas expressivity, efficiency, and optimality have been the typical goals in the development of CSP techniques; robustness-related issues appear with a clear relevance in dynamic environments, usually with incomplete or imprecise knowledge.

The general notion of robustness includes several and different concepts. However, despite several works on DCSP, there is still no a clear and common definition of robustness-related concepts. We have identified the concepts of robustness, stability, recoverability and reliability of DCSP solutions, on the basis on their related concepts in engineering and other areas. These definitions can be used to assess robustness-related features of solutions in DCSPs.

However, other relevant open issues remain open: How robustness can be efficiently measured and what does it guarantee? What factors does it depend on? How can it be obtained?

Typology of expected incidences, optimality of a solution, and constrainedness of a problem appear as the main factors that bound the desired level of robustness, stability, recoverability and reliability of solutions in DCSPs. Here, a clear further goal appears

addressed to define and parameterize a general model of robustness of DCSOPs in the broadest sense.

Acknowledgments

This work has been partially supported by the research projects TIN2010-20976-C02-01 (MICINN, Spain) and P19/08 (Min. de Fomento, Spain-FEDER).

References

1. D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.
2. R. Dechter and A. Dechter. Dynamic constraint networks. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88)*, pages 37–42, 1988.
3. M. Hazewinkel. Encyclopaedia of mathematics. Springer, 2002.
4. E. Hebrard. Robust solutions for constraint satisfaction and optimisation under uncertainty. phd thesis. University of New South Wales, 2007.
5. E. Hebrard, B. Hnich, and T. Walsh. Super solutions in constraint programming. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR-04)*, pages 157–172, 2004.
6. E. Jen. Stable or robust? whats the difference? *Complexity*, 8(3):12–18, 2003.
7. H. Kitano. Towards a theory of biological robustness. *Molecular Systems Biology*, 3(137), 2007.
8. C. Liebchen, M. Lbbecke, R. Mhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *LNCS 5868*, 2009.
9. A. Parkes, M. Ginsberg, and A. Roy. Supermodels and robustness. *Proceedings The Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, (334-339), 1998.
10. A. Rizk, G. Batt, F. Fages, and S. Solima. A general computational method for robustness analysis withapplications to synthetic gene networks. *Bioinformatics*, 25(12):168–179, 2009.
11. E. Szathmary. A robust approach. *Nature*, 439:19–20, 2006.
12. G. Verfaillie and N. Jussien. Constraint solving in uncertain and dynamic environments a survey. volume 10, pages 253–281, 2005.
13. G. Verfaillie and T. Schiex. Solution reuse in dynamic constraint satisfaction problems. In *Proc. of the 12th National Conference on Artificial Intelligence (AAAI-94)*, page 307 312, 1994.
14. R. Wallace and E. Freuder. Stable solutions for dynamic constraint satisfaction problems. In *Proc. 4th Int. Conf. on Principles and Practice of Constraint Programming (CP-98)*, page 447461, 1998.
15. R. Wallace, D. Grimes, and E. Freuder. Solving dynamic constraint satisfaction problems by identifying stable features. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI-09)*, pages 621–627, 2009.
16. R. Weigel and C. Bliek. On reformulation of constraint satisfaction problems. In *Proceedings European Conference on Artificial Intelligence (ECAI-98)*, pages 254–258, 1998.
17. S. Wiggins. Introduction to applied nonlinear dynamical systems and chaos. Springer, 1990.