An Adaptation of Path Consistency for Boolean Satisfiability: a Theoretical View of the Concept

Pavel Surynek[*]

Charles University in Prague
Faculty of Mathematics and Physics
Department of Theoretical Computer Science and Mathematical Logic
Malostranské náměstí 25, Praha, 118 00, Czech Republic
pavel.surynek@mff.cuni.cz

Abstract. The task of enforcing certain level of consistency in Boolean satisfiability problem (SAT problem) is addressed in this paper. The concept of path-consistency known from the constraint programming paradigm is revisited in this context. Augmentations how to make path-consistency more suitable for SAT are specifically studied. A stronger variant of path-consistency is described and its theoretical properties are investigated. It combines the standard path consistency on the literal encoding of the given SAT instance with global properties calculated from constraints imposed by the instance – namely with the maximum number of visits of a certain set by the path. Unfortunately, the problem of enforcing this variant of path-consistency turned out to be NP-hard. Hence, various types of relaxations of this stronger version of path-consistency were proposed. The relaxed version of the proposed consistency represents a trade-off between the inference strength and the complexity of its propagation algorithm. A presented theoretical analysis shows that computational costs of the proposed consistency are kept reasonably low. Preliminary experiments also show that the mentioned maximum number of visits calculated on several benchmark SAT instances provide non-trivial amount of information.

Keywords: local consistency, global consistency, path-consistency, CSP, SAT

1 Introduction and Motivation

A method how to increase the inference strength of path-consistency [13, 14] will be described. It combines the standard path-consistency on the literal encoding model [17] of the given Boolean satisfiability (SAT) instance [5] with global properties calculated from the instance. The existence of a path in a graph interpretation of the instance is being checked by the standard path-consistency. In the augmented variants, additional requirements are imposed on the path being checked to exist. Unfor-

[*] This work is supported by The Czech Science Foundation (Grantová agentura České republiky - GAČR) under the contract number 201/09/P318 and by The Ministry of Education, Youth and Sports, Czech Republic (Ministerstvo školství, mládeže a tělovýchovy ČR – MŠMT ČR) under the contract number MSM 0021620838.

tunately, the problem of checking the existence of a path according to augmented requirements turned out to be NP-complete [16]. Hence, various relaxations that still preserve the inference strength of augmented variants above the level of the standard path-consistency were proposed and evaluated. The ultimate goal of whole design of the adaptation of path-consistency is a tool for preprocessing SAT instances. The result of preprocessing should be a simplified SAT instance that is easier to solve.

2 Notations and Definitions

Concepts of *constraint satisfaction problem* (CSP) [7] and *Boolean satisfiability* (SAT) [5] need to be established first to make reasoning about path-consistency in the context of SAT easier to understand.

Definition 1 (*Constraint satisfaction problem - CSP*). Let $\mathbb{D}$ be a finite set representing *domain universe*. A *constraint satisfaction problem* [7] is a triple (X, C, D) where X is a finite set of *variables*, C is a finite set of *constraints*, and $D: X \longrightarrow \mathcal{P}(\mathbb{D})$ is a function that defines *domains* of individual variables from X (that is, $D(x) \subseteq \mathbb{D}$ is a set of values that can be assigned to the variable $x \in X$). Each constraint from $c \in C$ is of the form $\langle (x_1^c, x_2^c, \dots, x_{K^c}^c), R^c \rangle$ where $K^c \in \mathbb{N}$ is called an arity of the constraint c, the tuple $(x_1^c, x_2^c, \dots, x_{K^c}^c)$ with $x_i^c \in X$ for $i = 1,2, \dots, K^c$ is called a *scope* of the constraint, and the relation $R^c \subseteq D(x_1^c) \times D(x_2^c) \times \dots \times D(x_{K^c}^c)$ defines the set of tuples of values for that the constraint c is satisfied. The task is to find a valuation of variables $v: X \longrightarrow \mathbb{D}$ such that $v(x) \in D(x)$ $\forall x \in X$ and $(v(x_1^c), v(x_2^c), \dots, v(x_{K^c}^c)) \in R^c$ $\forall c \in C$. □

A constraint $c \in C$ with the scope $(x_1^c, x_2^c, \dots, x_{K^c}^c)$ will be denoted as $c(\{x_1^c, x_2^c, \dots, x_{K^c}^c\})$; this notation is useful when the ordering of variables in the scope is not known from the context; when ordering of variables in the scope matters, then a notation $c(x_1^c, x_2^c, \dots, x_{K^c}^c)$ will be used instead.

A CSP is called *binary* if all the constraints has the arity of two. The expressive power of a binary CSP is not reduced in comparison with a general one since every CSP can be transformed into an equivalent binary CSP [15]. The key concept of *path-consistency* [14] that is addressed in this paper is defined for binary CSPs only. It is also convenient to suppose, that each pair of variables is constrained by at most one constraint.

Definition 2 (*Boolean satisfiability problem - SAT*). Let B be a finite set of *Boolean variables*; that is, a set of variables that can be assigned either $FALSE$ or $TRUE$. A Boolean formula F over the set of variables B in a so called *conjunctive normal form* (*CNF*) [12] is the construct of the form $\bigwedge_{i=1}^{N} (\bigvee_{j=1}^{K_i} l_j^i)$ where l_j^i with either $l_j^i = y$ or $l_j^i = \neg y$ for some $y \in B$ for $i = 1,2, \dots, N$; $j = 1,2, \dots, K_i$ is called a *literal* and $(\bigvee_{j=1}^{K_i} l_j^i)$ for $i = 1,2, \dots, N$ is called a *clause*. The task is to find a valuation of Boolean variables $b: B \longrightarrow \{FALSE, TRUE\}$ such that F evaluates to $TRUE$ under b while $\neg$ (*negation*), $\vee$ (*disjunction*), and $\wedge$ (*conjunction*) are interpreted commonly in the Boolean algebra. A formula for that such a satisfying valuation exists is called *satisfiable*. □

It is a well known result that the language consisting of satisfiable formulas in *CNF* as well as general ones is an *NP*-complete problem [5, 9]. It is not difficult to observe that the language of solvable instances of CSP is *NP*-complete as well since it just generalizes SAT in fact (constraints are represented by clauses) while membership of CSP into the *NP* class is preserved by the generalization.

2 Path-consistency in CSP

The standard definition of *path-consistency* in CSP will be recalled before the augmented versions and their relaxations are introduced. The following definition refers to general paths of variables which is not necessary in fact. However, this style of definition will be more suitable for making intended augmentations.

Definition 3 (*Path-consistency - PC*). Let (X, C, D) be a binary *CSP* and let $P = (x_0, x_1, \dots, x_K)$ with $x_i \in X$ for $i = 0,1,\dots,K$ be a sequence of variables called a *path*. A pair of values $d_0 \in D(x_0)$ and $d_K \in D(x_K)$ is *path-consistent* with respect to P if there exists a valuation $v: \{x_0, x_1, \dots, x_K\} \longrightarrow \mathbb{D}$ with $v(x_0) = d_0 \land v(x_K) = d_K$ such that constraints $c(\{x_i, x_{(i+1) \bmod K}\})$ are satisfied by v for every $i = 0,1,\dots,K$. The path P is said to be *path-consistent* if all the pairs of values from $D(x_0)$ and $D(x_K)$ respectively are path-consistent with respect to P. Finally, the CSP (X, C, D) is said to be *path-consistent* if it is path-consistent for every path. □

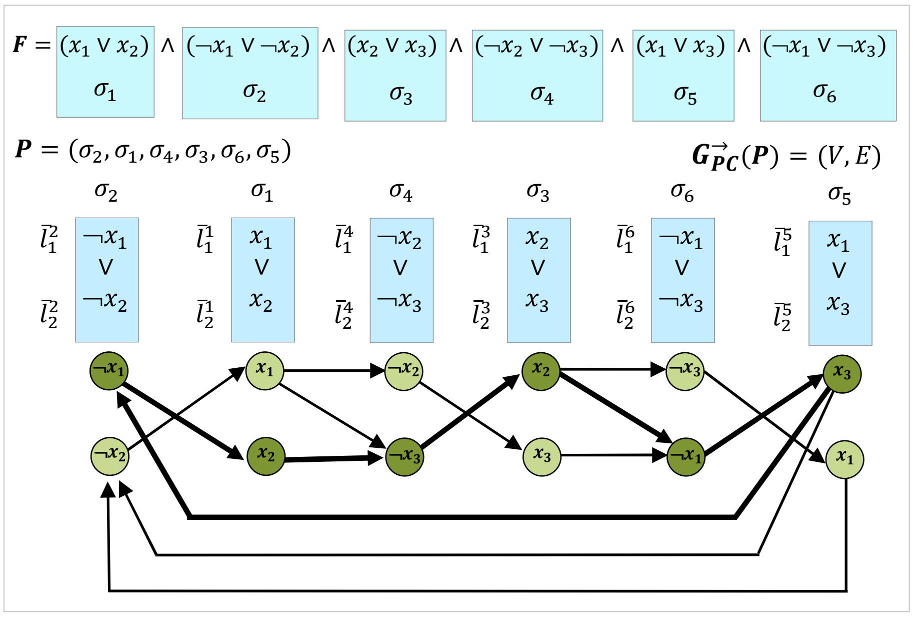
Notice that variables forming the path in the definition do not need to be necessarily distinct. Although the notion of path-consistency seems to be computationally infeasible since there are typically too many paths, it is sufficient to check path-consistency for all the paths consisting of triples of variables only to ensure that the given CSP is path-consistent [13, 14]. In other words, although it seems that path-consistency captures the problem globally (a path can go through large portion of variables of the instance), it merely defines a local property.

There exist many algorithms for enforcing path-consistency in a CSP such as PC-4 [10] and PC-6 [1, 3]. They differ in the representation of auxiliary data structures and the efficiency. The common feature of path-consistency algorithms is however the process how the consistency is enforced. It is done by eliminating pairs of inconsistent values until a path-consistent state is reached (the smallest set of pairs of values such that their elimination makes the problem path-consistent is being pursued). The process of elimination of pairs of values is typically done by pruning extensional representation of constraints (lists of allowed tuples) to forbid more pairs of values.

3 Standard Path-consistency in SAT

The aim of this work is to modify path-consistency to make it applicable on SAT and to increase its inference strength by incorporating certain global reasoning into it. The easier task is to make path-consistency applicable on SAT - it is sufficient to model SAT as CSP. A so called *literal encoding* [17], which of the result is a binary CSP, is particularly used. This kind of encoding is especially suitable since it allows natural expressing of path-consistency in terms of graph constructs.

Let $F = \bigwedge_{i=1}^{N} (\bigvee_{j=1}^{K_i} l_j^i)$ be a Boolean formula in CNF over a set of Boolean variables B. Let $\mathbb{D} = \bigcup_{i=1}^{n} (\bigcup_{j=1}^{k_i} \{\bar{l}_j^i\})$ be a domain universe; that is, a constant symbol with the stripe is introduced into $\mathbb{D}$ for each literal occurrence in F (notice that, each occurrence of a literal corresponds to a different constant symbol). The corresponding CSP (X, C, D) using literal encoding is built as follows: $X = \{\sigma_1, \sigma_2, \dots, \sigma_N\}$; that is, a variable is introduced for each clause of F; it holds for $D: X \longrightarrow \mathcal{P}(\mathbb{D})$ that $D(\sigma_i) = \bigcup_{j=1}^{K_i} \{\bar{l}_j^i\}$; that is, the domain of an i-th clause contains constant symbols corresponding to all its literals. A constraint $c(\{\sigma_{i_1}, \sigma_{i_2}\}) = \langle(\sigma_{i_1}, \sigma_{i_2}), R^c\rangle$ is introduced over every pair of variables with $i_1, i_2 \in \{1,2,\dots,N\} \wedge i_1 \neq i_2$ where a variable $x \in B$ such that either $x \in D(\sigma_{i_1}) \wedge \neg x \in D(\sigma_{i_2})$ or $\neg x \in D(\sigma_{i_1}) \wedge x \in D(\sigma_{i_2})$ exists. Such a constraint $c(\{\sigma_{i_1}, \sigma_{i_2}\})$ then forbids every tuple of values $(\bar{l}_{j_1}^{i_1}, \bar{l}_{j_2}^{i_2})$ such that there exists $x \in B$ for that either $\bar{l}_{j_1}^{i_1} = x \wedge \bar{l}_{j_2}^{i_2} = \neg x$ or $\bar{l}_{j_1}^{i_1} = \neg x \wedge \bar{l}_{j_2}^{i_2} = x$ (that is, the tuple $(\bar{l}_{j_1}^{i_1}, \bar{l}_{j_2}^{i_2})$ is removed from R^c which has been initially set to $D(\sigma_{i_1}) \times D(\sigma_{i_2})$). A solution of the resulting CSP (X, C, D) corresponds to the valuation of Boolean variables of B that satisfies F and vice versa [17].



$\boldsymbol{F} = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$
σ_1 σ_2 σ_3 σ_4 σ_5 σ_6
$\boldsymbol{P} = (\sigma_2, \sigma_1, \sigma_4, \sigma_3, \sigma_6, \sigma_5)$
$\boldsymbol{G_{\overrightarrow{PC}}(P)} = (V, E)$
σ_2 σ_1 σ_4 σ_3 σ_6 σ_5
$\bar{l}_1^2$ $\bar{l}_2^2$ $\bar{l}_1^1$ $\bar{l}_2^1$ $\bar{l}_1^4$ $\bar{l}_2^4$ $\bar{l}_1^3$ $\bar{l}_2^3$ $\bar{l}_1^6$ $\bar{l}_2^6$ $\bar{l}_1^5$ $\bar{l}_2^5$
$\neg x_1 \vee \neg x_2$ $x_1 \vee x_2$ $\neg x_2 \vee \neg x_3$ $x_2 \vee x_3$ $\neg x_1 \vee \neg x_3$ $x_1 \vee x_3$
$\neg x_1$ $\neg x_2$ x_1 x_2 $\neg x_2$ $\neg x_3$ x_2 x_3 $\neg x_3$ $\neg x_1$ x_3 x_1

Fig. 1. *An illustration of path-consistency in the CSP model of a SAT problem.* The SAT problem represented by a formula F shown here is a representation of the requirement of selecting an odd number of variables from every of the following sets to be true: $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_2, x_3\}$. Observe, that there is no satisfying valuation of F. However, the pair of literals $\neg x_1$ and x_3 from the left most variable and from the right most variable respectively are path-consistent with respect to a depicted path P since they are non-conflicting and there exists a path from the left to the right consisting of edges between neighboring variables connecting allowed pairs of values (the path is marked by bold edges and by darker vertices).

Having the CSP model of SAT it is possible to check path-consistency for the corresponding CSP model and proclaim the original SAT path-consistent or path-inconsistent accordingly. If elements of variable domains are interpreted as vertices and allowed tuples of values as directed edges connecting them, then path-consistency with respect to a given path can be interpreted as existence of paths in the resulting directed graph.

More precisely, let $P = (\sigma_{i_0}, \sigma_{i_1}, \dots, \sigma_{i_K})$ with $i_j \in \{1,2,\dots,N\}$ for $j = 0,1,\dots,K$ be a sequence of variables in the literal encoding CSP model (X, C, D). A directed graph $G_{\vec{PC}}(P) = (V, E)$, in which path-consistency can be interpreted as the existence of paths, is defined as follows: $V = \bigcup_{j=0}^{K} D(\sigma_{i_j})$ and if $(\vec{l}_{j_1}^{i_j}, \vec{l}_{j_2}^{i_{(j+1) \bmod K}}) \in R^{c(\sigma_{i_j}, \sigma_{i_{(j+1) \bmod K}})}$ then a directed edge $(\vec{l}_{j_1}^{i_j}, \vec{l}_{j_2}^{i_{(j+1) \bmod K}})$ is included into E. A pair of values $\vec{l}_{j_1}^{i_0} \in D(\sigma_{i_0})$ and $\vec{l}_{j_2}^{i_K} \in D(\sigma_{i_K})$ is path-consistent with respect to the path P if there is an edge $(\vec{l}_{j_1}^{i_K}, \vec{l}_{j_1}^{i_0})$ in $G_{\vec{PC}}(P)$ and there exists a path from the vertex $\vec{l}_{j_1}^{i_0}$ to the vertex $\vec{l}_{j_2}^{i_K}$ in $G_{\vec{PC}}(P)$. The graph $G_{\vec{PC}}(P)$ will be called a *graph interpretation* of path-consistency – see Fig. 1 for illustration.

Notice that path-consistency is *incomplete* in the sense that a pair of values may be path-consistent even if there is no solution of the problem that contains this pair of values (see Fig. 1 again). Analogically, the problem may be path-consistent (that is, path-consistent with respect to all the paths) even if it has no solution actually. The partial reason for this weakness of path-consistency is that many constraints are ignored when a pair of values is checked. This is especially apparent if a longer path of variables is considered. Only constraints over pairs of variables neighboring in the path are considered while many constraints such as that for example over the first and the third variable in the path are ignored. This property is disadvantageous especially in SAT where stronger reasoning is typically more beneficial.

For further augmentation of path-consistency, it is also convenient to prepare a so called *auxiliary constraint graph* for the model with respect to the path P that reflects all the constraints over the variables of the path P. It is an undirected graph $G_{CSP}(P) = (V, E)$ and it is defined as follows: $V = \bigcup_{j=0}^{K} D(\sigma_{i_j})$; an edge $\{\vec{l}_{j_1}^{i_j}, \vec{l}_{j_2}^{i_k}\}$ is added to E if $(\vec{l}_{j_1}^{i_j}, \vec{l}_{j_2}^{i_k}) \in R^{c(\sigma_{i_j}, \sigma_{i_k})}$; and all the edges $\{\vec{l}_{j_1}^{i_j}, \vec{l}_{j_2}^{i_j}\}$ for all $j = 1,2,\dots,N$ and $j_1, j_2 = 1,2,\dots,K_j \wedge j_1 \neq j_2$. Observe that the auxiliary constraint graph subsumes the graph interpretation with respect to the same path. Notice also, that there is a complete subgraph over vertices corresponding to values from the domain of the same variable.

4 Making Path-consistency Stronger

A modification of path-consistency has been proposed to overcome mentioned limitations of the standard version. To increase inference strength of path-consistency additional requirements on the path in the graph model are imposed. These additional requirements reflect constraints over non-neighboring variables in the path of variables. As the auxiliary constraint graph represents an explicit representation of constraints, it is exploited for determining additional requirements.

4.1 An Initial Augmentation of Path-consistency

An approach adopted in this work restricts the size of the intersection of the constructed path with certain subsets of vertices in the graph interpretation of path-consistency. More precisely, let $G_{\overrightarrow{PC}}(P) = (V, E)$ be a graph interpretation of path-consistency in a CSP model of SAT (X, C, D). The set of vertices V is partitioned into disjoint sequences $L_1, L_2, \ldots, L_M$ called *layers* (that is, $\bigcup_{i=1}^{M} \hat{L}_i = V$ and $\hat{L}_i \cap \hat{L}_j$ $\forall i, j \in \{1,2, \ldots, M\} \wedge i \neq j$; where denotes the union of the sequence $\hat{A}$, that is $\hat{A} = \bigcup_{i=1}^{n} \{a_i\}$ for $A = [a_1, a_2, \ldots, a_n]$). The maximum size of the intersection of the path being checked to exist with individual layers is determined using the set of constraints C (notice that all the constraints over P are considered – not only constraints over neighboring variables in P). This proposal will be called an *initial augmentation* of path-consistency in the rest of the text.

The concept of the initial augmentation of path-consistency comes from [16]. The process of decomposition of the set of vertices into layers is done over the corresponding auxiliary constraint graph $G_{CSP}(P)$. Vertices of $G_{CSP}(P)$ are decomposed into vertex disjoint stable sets (a stable set is a subset of vertices of a graph where no two vertices are adjacent with respect to edges). The knowledge of such decomposition can be then used to partition vertices into layers that directly correspond to found stable sets. However, determining a stable subset is a difficult task itself. Hence a greedy approach has been used to obtain an acceptable solution. More details about how to decompose vertices into layers greedily for the initial augmentation can be found in [16].

Since it is possible to assign to a variable at most one value from values corresponding to vertices of the stable set in $G_{CSP}(P)$, the maximum size of the intersection of the path with a layer is thus at most 1. Notice, that at most one value from vertices corresponding to the domain of a variable can be selected (this is due to the presence of the complete subgraph over the set of vertices corresponding to the domain of a variable in $G_{CSP}(P)$). Notice further, that if a value corresponding to a vertex in a stable set is selected than all the values corresponding to other vertices of the stable set are ruled out since they are in conflict with the selected value with respect to constraints.

A quite negative result has been obtained in [16]. It has been shown that finding a path, which conforms to the calculated maximum size of the intersection with individual layers, corresponds to finding a Hamiltonian path [4]. This is known to be an *NP*-hard problem. Hence, it is not tractable to find a path that satisfies defined requirements exactly. Moreover, initial experiments showed that it is almost impossible to make any reasonable relaxation of proposed requirements. Every relaxation of requirements on the path being constructed proposed by the author leads to weakening the modified path-consistency down to the level of the standard version of path-consistency (specifically, several adaptations of the algorithm for finding single source shortest paths [6] have been evaluated by the author).

These initial findings founded an effort to further augment requirements on the constructed path in order to allow developing stronger and more efficient relaxations. The result of this effort is a concept of a so called *modified version of path-consistency*.

4.2 A Modified Version of Path-consistency

Again, partitioning of vertices of $G_{\vec{PC}}(P)$ into layers is supposed. In addition, the sequencing of variables in the path P is exploited for defining the maximum size of the intersection of the constructed path with layers. Particularly, the path being constructed is required to conform to the calculated maximum size of the intersection with vertices of the layer preceding a given vertex of the path with respect to the sequencing of variables in P. The maximum size of the intersection is again imposed by the set of constraints C. More precisely, let $L_1, L_2, \dots, L_M$ be layers of $G_{\vec{PC}}(P)$; let a function $\chi: V \longrightarrow \mathbb{N}$ defines requirements on the maximum size of intersections imposed by constraints as follows: $\chi(v_j^l)$ is the maximum size of the intersection of the constructed path with a set of vertices $\{v_0^l, v_1^l, \dots v_i^l\}$ where $L_l = [v_0^l, v_1^l, \dots, v_{K^l}^l]$ with $l \in \{1,2,\dots,M\}$ and $j \in \{0,1,\dots,K^l\}$. Let a consistency defined by this new requirement on the constructed path be called a *modified path-consistency*. Observe that this new concept is a generalization of the initial augmentation described above (see Fig 2 for illustration).

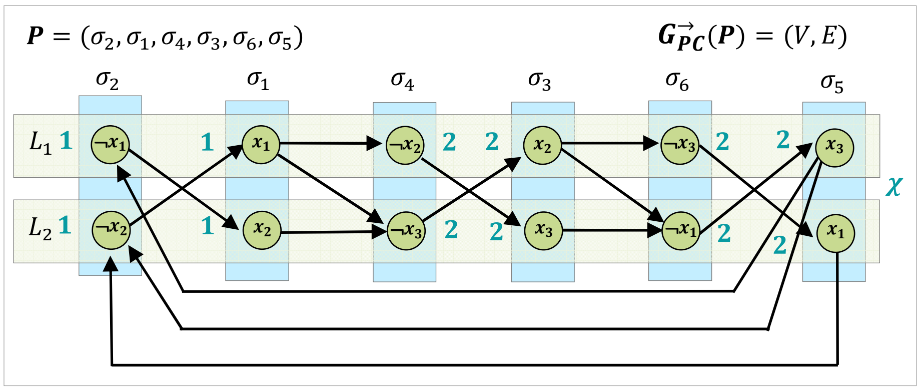


Fig 2. *An illustration of modified path-consistency in the CSP model of a SAT problem.* The maximum size of the intersection of the constructed path with vertices preceding the given vertex (including) in its layer is calculated using constraints for each vertex - these maximum sizes are denoted as the function χ. For example, having $\chi(\bar{l}_1^3) = 2$ then the constructed path can intersect the subset of vertices $\{\bar{l}_1^2, \bar{l}_1^1, \bar{l}_1^4, \bar{l}_1^3\}$ (first occurrences of literals in first four variables of the path P) of the layer L_1 in at most two vertices. Observe, that these requirements on the path being constructed rules out its existence for connecting a pair of vertices $\bar{l}_1^2$ from the left most variable (occurrence of literal $\neg x_1$) and $\bar{l}_1^5$ from the right most variable (occurrence of literal x_3). Compare it with the standard path-consistency in Fig. 1 where the corresponding path connecting the same pair of vertices exists.

It is intractable to construct a path conforming to the maximum sizes of intersections determined by χ as in the case of the initial augmentation. Nevertheless, it is possible to make a tractable relaxation of these requirements which does not collapse down to the level of the standard path-consistency.

Let us now briefly describe such a tractable relaxation. Suppose that χ is already known (the process of calculation of χ will be described in the following section). Let

$d_0 \in D(\sigma_{i_0})$ and $d_K \in D(\sigma_{i_K})$ be a pair of values for that a consistency is being checked. Two assignments will be maintained: $\Sigma: V \longrightarrow \mathbb{N}_0$ and $\psi: V \longrightarrow {\mathbb{N}_0}^{M\times(K+1)}$ where ${\mathbb{N}_0}^{M\times(K+1)}$ denotes matrices of the size $M \times (K+1)$ over $\mathbb{N}_0$. The assignment Σ will express the total number of distinct paths in $G_{PC}^{\rightarrow}(P) = (V, E)$ starting in d_0 and ending in a given vertex. Observe, that it is easy to calculate $\Sigma(v)$. It is determined recursively by the expression: $\Sigma(v) = \sum_{u\in V,(u,v)\in E} \Sigma(u)$, while $\Sigma(d_0) = 1$. The assignment ψ expresses statistical information about paths in $G_{PC}^{\rightarrow}(P)$ starting in d_0 and ending in a given vertex regarding the size of the intersection with layers. More precisely, an element of $\psi(v)$ at i-th row and j-th column (that is, $\psi(v)_{i,j}$) with $v \in V$, $i \in \{1,2,\dots,M\}$, and $j \in \{0,1,\dots,K\}$ represents the number of distinct paths starting in d_0 and ending in v intersecting with the layer L_i in exactly j vertices that conform to relaxed requirements (that is, the size of the intersection of these paths with L_i is j). If the mentioned conformation to relaxed requirements is omitted, the information maintained in ψ is not difficult to be calculated recursively for every vertex of $G_{PC}^{\rightarrow}(P)$. However, as it is algorithmically more complex calculation, it is deferred to the section devoted to algorithms.

Requirements on the size of the intersection of the constructed path with layers represented by χ are relaxed in the following way. If it is detected that all the paths staring in d_0 and ending in v intersects the layer containing v in more vertices than it is allowed by χ, then it is possible to conclude that there is no path connecting d_0 and v that conforms to calculated maximum sizes of intersections with layers. Hence, v is unreachable from d_0 under given circumstances. The described relaxation can be expressed using defined assignments Σ and ψ. Let L_{l^v} be a layer containing v (that is, $v \in \hat{L}_{l^v}$). If there is some $j > \chi(v)$ such that $\psi(v)_{l^v,j} = \Sigma(v)$, then there is no path connecting d_0 and v conforming to the maximum sizes of intersection with layers. Observe, that although there is no $j > \chi(v)$ such that $\psi(v)_{l^v,j} = \Sigma(v)$, the required path still need not to exist. This is the principle which is called the relaxation in the context of this paper.

If it is detected that there is no path connecting d_0 and d_K that conforms to relaxed requirements on the maximum sizes of intersections with layers, the pair of values d_0 and d_K is said to be *inconsistent* with respect to the *modified path-consistency*.

5 Modified Path-consistency Enforcing Algorithms

It is necessary to clarify several essential steps in order to be able to enforce modified path-consistency according to the suggestion in the previous section. These essential steps are: how to construct **layer decomposition** of the graph interpretation of path consistency, then we need to know how to determine maximum **sizes of intersections** with layers, and finally how to perform modified **path-consistency checking**.

Constructions carried out in all these steps must regard the objective that the inference ability of the resulting modified path-consistency should as strong as possible (since every step induces a possible relaxation, this means that all these relaxations should not relax the original constraints too much).

This section is devoted to the algorithmic point of view of suggestions from previous sections. Thus aspects that have been explained informally so far will be now explained in all the details using algorithms written in pseudo-code.

5.1 Construction of the Layer Decomposition

The first step is represented by construction of the layer decomposition $L_1, L_2, \ldots, L_M$ of a given graph interpretation of path consistency $G_{\vec{PC}}(P) = (V, E)$ with respect to a path P. It has been reported how to construct layer decomposition from the knowledge of the auxiliary constraint graph $G_{CSP}(P)$ within the initial augmentation of path-consistency. The process within modified path-consistency is similar.

Algorithm 1. *Construction of a layer decomposition*. The input parameters are: (X, C, D) which is a CSP model of the SAT instance, a path P, and an auxiliary constraint graph with respect to (X, C, D) the path P as $G_{CSP}(P)$. The output of the algorithm is a sequence of vertex disjoint subsets of $G_{\vec{PC}}(P)$ (or equivalently of $G_{CSP}(P)$) which form layer decomposition.

```
function Construct-Layer-Decomposition($((X,C,D), P, G_{CSP}(P) = (V,E))$): sequence
1:  let $P = (\sigma_{i_0}, \sigma_{i_1}, \ldots, \sigma_{i_K})$
2:  $M \leftarrow 1$
3:  $\Pi \leftarrow \emptyset$
4:  for $j = 0,1,\ldots,K$ do
5:      for each $u \in D(\sigma_{i_j})$ do
6:          if $u \notin \Pi$ then
7:              $L_M \leftarrow$ Construct-Next-Layer($u, \Pi, (X,C,D), P, G_{CSP}(P)$)
8:              $\Pi \leftarrow \Pi \cup L_M$
9:              $M \leftarrow M + 1$
10: return $(L_1, L_2, \ldots, L_M)$

function Construct-Next-Layer($(u_0, \Pi, (X,D,C), P, G_{CSP}(P) = (V,E))$): set
1:  let $P = (\sigma_{i_0}, \sigma_{i_1}, \ldots, \sigma_{i_K})$
2:  let $u_0 \in D(\sigma_{i_k})$
3:  $L \leftarrow \{u_0\}$
4:  for $j = k+1, k+2, \ldots, K$ do
5:      $v_{best} \leftarrow \perp$
6:      for each $v \in D(\sigma_{i_j})$ do
7:          if $v \notin \Pi$ then
8:              if $|\{u | u \in L \wedge \{u,v\} \notin E\}| > |\{u | u \in L \wedge \{u, v_{best}\} \notin E\}|$ then
9:                  $v_{best} \leftarrow v$
10:     if $v_{best} \neq \perp$ then
11:         $L \leftarrow L.v_{best}$
12: return $L$
```

It works with auxiliary constraint graphs $G_{CSP}(P)$ again while construction of each layer is done in the direction from the first variable to the last variable of the path P while sequencing of variables in P are respected. The process is formally expressed using pseudo-code as Algorithm 1.

For each vertex of $G_{CSP}(P)$ it is checked whether it has been already included in some of the previously constructed layers. If not, the construction of a new layer is started and the vertex is included in it. Let u_0 be the first vertex of the just started layer. Let it belong to the domain of a variable σ_{i_k}. Then variables following σ_{i_k} in P are traversed and vertices corresponding to their domain elements are checked whether it is advantageous to include them. Vertices not yet included in any of the previously constructed layers are considered only. At most one vertex from each variable domain is included into each layer. If there are multiple vertices remaining in the variable domain, the vertex which is disconnected from the most vertices already included in the currently constructed layer with respect to edges of E is selected for inclusion. The selected vertex is concatenated to the constructed layer finally.

This way of selection of vertices to layers prefers small intersections of the path being constructed with layers. This is due to the fact that selection of vertices into layers according to the described criterion prefers that few variables can have assigned values corresponding to vertices of the layer not to violate constraints. Consequently, this selection is in accordance with the objective of obtaining strongly inferring modified path-consistency. In other words, we are trying to forbid as many paths as possible (in fact, these paths are forbidden by constraints we just need to identify as many of them as possible).

Observe that the worst case time complexity of the algorithm is $\mathcal{O}(|P|^2|\mathbb{D}|)$. The worst case space complexity is $\mathcal{O}(|V| + |E|)$.

5.2 Calculation of Maximum Sizes of Intersection with Layers

The next step is the calculation of maximum sizes of intersections of the path being constructed with individual layers represented by $\chi: V \longrightarrow \mathbb{N}$. It is supposed that layer decomposition has been already constructed. Vertices of each layer are traversed from the beginning. This way of traversal corresponds to the traversal of vertices in the sequence of vertices of the path P. The process is formally expressed using pseudocode as Algorithm 2.

Let $L_l = [v_0^l, v_1^l, \dots, v_{K^l}^l]$ with $l \in \{1,2,\dots,M\}$ be a layer which is currently considered. The value of $\chi(v_k^l)$ with $k \in \{0,1,\dots,K^l\}$ is calculated by considering two alternatives. The first alternative is that the vertex v_k^l is considered to be included in the potential intersection with the constructed path. The second alternative is that the vertex v_k^l is not included in this way. The higher value from these two alternatives is eventually assigned to $\chi(v_k^l)$.

In the first alternative, the vertices from the set $\{v_0^l, v_1^l, \dots, v_{k-1}^l\}$ that are connected with v_k^l are considered. These vertices can be potentially included into the intersection of the layer L_l with the constructed path together with the vertex v_k^l. Then maximum of χ for these vertices is calculated and this value plus 1 represents the maximum size of the intersection in this alternative (in the case when maximum is not defined - the case of the empty set - the value 0 is used).

In the second alternative, the vertex v_k^l is not considered for inclusion into the intersection with the path. In such a case, the value of $\chi(v_k^l)$ can be inherited from $\chi(v_{k-1}^l)$ since the vertex v_k^l has no influence on the size of the intersection.

Algorithm 2. *The process of calculation of maximum intersection sizes.* The input of the algorithm is a layer decomposition of a $G_{PC}^{\rightarrow}(P)$ denoted as $\mathcal{L}$ and the auxiliary constraint graph $G_{CSP}(P)$ with respect to (X, C, D) and the path P as $G_{CSP}(P)$ itself. The output of the algorithm is an assignment $\chi: V \longrightarrow \mathbb{N}$ which determines the maximum size of the intersection of the path being constructed with the set of vertices of the layer preceding the given vertex. The maximum size of the intersection is determined by constrains imposed by the instance.

```
function Calculate-Maximum-Intersection-Sizes($\mathcal{L}, G_{CSP}(P) = (V, E)$): assignment
1:  let $\mathcal{L} = \{L_1, L_2, \dots, L_M\}$
2:  for $l = 1,2,\dots,M$ do
3:      let $L_l = [v_0^l, v_1^l, \dots, v_{K^l}^l]$
4:      $\chi_{prev} \leftarrow 0$
5:      for $k = 0,1,\dots,K^l$ do
6:          $\chi_{best} \leftarrow 0$
7:          for $j = 0,1,\dots,k-1$ do
8:          if $\{v_j^l, v_k^l\} \in E$ then
9:              if $\chi(v_j^l) > \chi_{best}$ then
10:                 $\chi_{best} \leftarrow \chi(v_j^l)$
11:         $\chi(v_k^l) \leftarrow \max\{\chi_{best} + 1, \chi_{prev}\}$
12:         $\chi_{prev} \leftarrow \chi(v_k^l)$
13: return $\chi$
```

The following proposition summarizes the correctness of calculation of the maximum sizes of intersection with layers and their application in forbidding paths. It is stated without the proof.

Proposition 1. Let $d_0 \in D(\sigma_{i_0})$ and $d_K \in D(\sigma_{i_K})$. If there is no path π connecting d_0 and d_K in $G_{PC}^{\rightarrow}(P)$ such that $\pi \cap \{v_0^l, v_1^l, \dots, v_k^l\} \leq \chi(v_k^l)$ for each $l \in \{1,2,\dots,M\}$ and for each $k \in \{0,1,\dots,K^l\}$, then there is no assignment of values to variables of P such that all the constraints over P are satisfied. ■

The worst case time complexity of the algorithm for calculating maximum sizes of intersections is $O(|P|^2|\mathbb{D}|)$ again. The worst case space complexity is $O(|V| + |E|)$.

5.3 Modified Path-consistency Checking

The final step is modified path-consistency checking for a pair of values d_0 and d_K from both ends of the path P being checked supposed that the layer decomposition and maximum sizes of intersections with layers have been constructed. The process is formally expressed using pseudo-code as Algorithm 3.

The algorithm traverses values from domains of the variables of the path P from the beginning of the path P in the graph interpretation of path-consistency $G_{PC}^{\rightarrow}(P) = (V, E)$. For each vertex $v \in V$ along the traversal, assignments Σ and ψ are calculated.

The major idea within the algorithm is represented by the moment when it is detected that all the paths starting in d_0 and ending in a given vertex exceed the maximum size of the intersection with some of the layers (lines 30-35). In such a situation, no of the paths ending in the given vertex can be prolonged into the vertex from the next variable in P without violating the constraints. This claim is formalized by the following proposition which is again stated without the proof. The proposition also formalizes the second stage of relaxation used in our proposal.

Algorithm 3. *The modified path-consistency checking algorithm.* The input of the algorithm is a pair of vertices d_0 and d_K from both ends of the path P which also the parameter. The next parameters are the layer decomposition $\mathcal{L}$, CSP model of path-consistency (X, C, D), and the graph interpretation of path-consistency $G_{PC}^{\rightarrow}(P)$. The output of the algorithm is the Boolean indicator whether the given pair of values is allowed.

```
function Check-Modified-Path-Consistency($d_0, d_K, P, \mathcal{L}, (X, C, D), G_{PC}^{\rightarrow}(P) = (V, E)$): boolean
1:  if $(d_0, d_K) \notin E$ then
2:      return $FALSE$
3:  let $P = (\sigma_{i_0}, \sigma_{i_1}, \dots, \sigma_{i_K})$
4:  let $\mathcal{L} = \{L_1, L_2, \dots, L_M\}$
5:  let $d_0 \in L_l$
6:  $\Sigma(d_0) \leftarrow 1$
7:  $\psi(d_0) \leftarrow 0^{M\times(K+1)}$; $\psi(d_0)_{l,1} \leftarrow 1$
8:  for each $v \in D(\sigma_{i_0})$ do
9:      if $v \neq d_0$ then
10:         $\Sigma(v) \leftarrow 1$
11:         $\psi(v) \leftarrow 0^{M\times(K+1)}$
12: for $k = 1,2, \dots, K$ do
13:     for each $v \in D(\sigma_{i_k})$ do
14:         $\Sigma(v) \leftarrow 0$
15:         $\psi(v) \leftarrow 0^{M\times(K+1)}$
16:         let $v \in \hat{L}_{l^v}$
17:         for each $u \in D(\sigma_{i_{k-1}})$ do
18:             let $u \in \hat{L}_{l^u}$
19:             if $(u, v) \in E$ then
20:                 $\Sigma(v) \leftarrow \Sigma(v) + \Sigma(u)$
21:                 for $i = 1,2, \dots, M$ do
22:                     for $j = 0,1, \dots, K$ do
23:                         if $i = l^v$ then
24:                             if $j = 0$ then
25:                                 $\psi(v)_{i,0} \leftarrow 0$
26:                             else
27:                                 $\psi(v)_{i,j} \leftarrow \psi(v)_{i,j} + \psi(u)_{i,j-1}$
28:                         else
29:                             $\psi(v)_{i,j} \leftarrow \psi(v)_{i,j} + \psi(u)_{i,j}$
30:         $\omega \leftarrow TRUE$
31:         for $j = 0,1, \dots, \chi(v)$ do
32:             if $\psi(v)_{l^v,j} > 0$ then
33:                 $\omega \leftarrow FALSE$
34:         if $\omega$ then
35:             $\psi(v) \leftarrow 0^{M\times(K+1)}$
36: if $\psi(d_K) \neq 0^{M\times(K+1)}$ then
37:     return $TRUE$
38: else
39:     return $FALSE$
```

Proposition 2. Let $d_0 \in D(\sigma_{i_0})$ and $d_K \in D(\sigma_{i_K})$. If the modified path consistency algorithm does not find the path conforming to requirements imposed by the algorithm then there is no path π connecting d_0 and d_K in $G_{\vec{PC}}(P)$ such that $\pi \cap \{v_0^l, v_1^l, \dots, v_k^l\} \leq \chi(v_k^l)$ for each $l \in \{1,2,\dots,M\}$ and for each $k \in \{0,1,\dots,K^l\}$. ■

The worst case time complexity of the algorithm is $O(|P|^2|\mathbb{D}|^3)$, the worst case space complexity is $O(|P|^2|\mathbb{D}|^2)$.

At this moment, a question may arise whether the modified path-consistency is actually stronger than the standard path-consistency. The answer is that it really is. The particular instance, where the modified path-consistency can infer that there is no consistent pair of values while the standard path-consistency is unable to infer this, is an encoding of the *pigeon hole principle* [1].

6 Preliminary Experimental Evaluation

We have performed a preliminary experimental evaluation of the above suggestions. The evaluation was targeted on determining the quality of layer decomposition and maximum sizes of intersections with layers determined by proposed algorithms (Algorithm 1 and Algorithm 2).

Table 1. *Maximum intersection sizes with the first layer of the layer decomposition*. The intersection sizes are calculated in the graph interpretation of several satisfiability instances from SATLib. The determining of layers and maximum size of intersections was done using Algorithm 1 and Algorithm 2 respectively.

SAT instance	Maximum **intersection** with $L_1 = [v_0^1, v_1^1, \dots, v_7^1]$							
	$\chi(v_0^1)$	$\chi(v_1^1)$	$\chi(v_2^1)$	$\chi(v_3^1)$	$\chi(v_4^1)$	$\chi(v_5^1)$	$\chi(v_6^1)$	$\chi(v_7^1)$
ais12.cnf	1	1	1	1	1	1	1	1
hanoi4.cnf	1	2	2	3	3	3	4	4
huge.cnf	1	1	2	2	2	2	3	3
jnh1.cnf	1	2	2	3	4	4	4	5
par16-1.cnf	1	1	1	2	2	2	2	2
par16-1-c.cnf	1	2	2	3	3	4	4	5
pret150_75.cnf	1	1	2	2	3	3	4	4
s3-3-3-8.cnf	1	1	2	3	3	4	4	5
ssa7552-160.cnf	1	1	2	3	4	4	5	6
sw100-5.cnf	1	1	2	2	2	2	3	3
Urq8_5.cnf	1	1	2	2	3	3	4	4
uuf250-0100.cnf	1	1	2	2	3	3	4	4

Several instances from the Satisfiability Library (SATLib) [11] and from [1] were encoded using literal encoding and graph interpretations were constructed for them with respect to the path of variables P consisting of 8 clauses starting with the first clause of the instance. The remaining clauses for the path P were selected so that they are the most constrained with already selected clauses for P. This way of selection prefers choosing of highly constrained path for subsequent construction of layers and calculation of maximum sizes of intersections. In order to yet more increase constraining of the generated graph interpretation of $G_{\vec{PC}}(P)$ the constraint model

(X, C, D) is enriched by additional constraint inferred by the *singleton unit propagation* [8] (that is, each literal is assigned the value $TRUE$ and unit propagation is performed; assignments to literals enforced by the propagation together with assignment to the original literal form additional constraints).

To provide reproducibility of results all the results presented in this paper, the complete source code, and input data are provided at the web: http://ktiml.mff.cuni.cz/~surynek/research/csclp2010. Results regarding maximum sizes of intersections are shown in Table 1. Parts of the auxiliary constraint graph restricted on the first layer of several instances are shown in Fig 3. Although the experimental results are incomplete at the current stage of the development, they indicate that information captured by constraints over non-neighboring variables in the path of variables is relatively strongly reflected in the maximum sizes of intersections with layers.

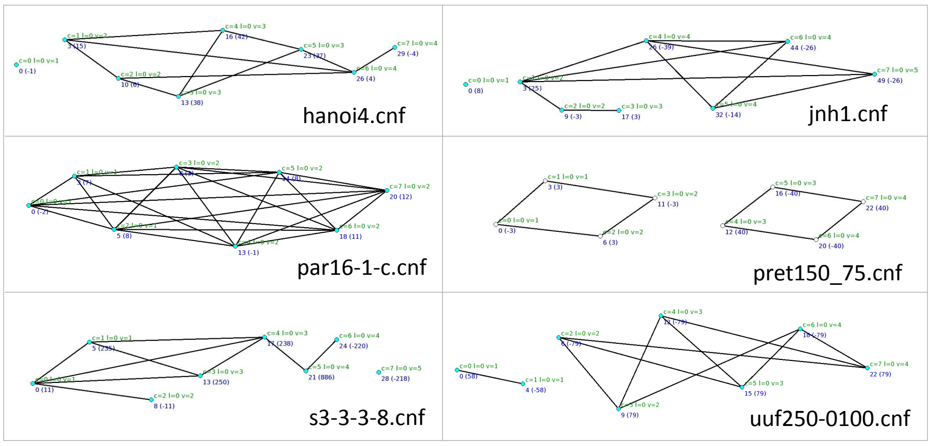


Fig 3. *An illustration of first layers of the auxiliary constraint graph of several satisfiability instances.* Several sparse and dense graphs are shown together with calculated maximum sizes of intersection of the path being constructed with the layer.

7 Conclusion and Future Work

A new consistency for Boolean satisfiability has been proposed in this paper. The new type of consistency augments the standard path-consistency by exploiting global properties of the input instance. Particularly, stronger requirements are imposed on the path being checked to exist compared to the situation in the standard path-consistency – namely, the size of the intersection of the path with certain sets called layers is restricted. Preliminary experimental evaluation has shown that it is possible to relatively successfully derive the restriction on the size of the intersection with layers on several well known benchmark SAT instances.

For future work we plan to further reduce the relaxation in the calculation of the maximum sizes of intersections with layers. The simple augmentation can be done in the alternative where the new vertex is considered to be selected into the intersection. The number of vertices in the layer not ruled out by constraints when the new vertex is included should be taken into account. Indeed, the future work is also to evaluate inference strength of the modified path-consistency checking algorithm itself (Algorithm 3). We also would like to evaluate possible applications of modified path-consistency as a preprocessing tool for SAT solving.

References

1. Aloul, F. A., Ramani, A., Markov, I. L., Sakallah, K. A.: Solving Difficult SAT Instances in the Presence of Symmetry. Proceedings of the 39th Design Automation Conference (DAC 2002), 731-736, USA, ACM Press, 2002, http://www.aloul.net/benchmarks.html, [October 2010] .
2. Chmeiss, A., Jégou, P.: Two New Constraint Propagation Algorithms Requiring Small Space Complexity. Proceedings of the 8th International Conference on Tools with Artificial Intelligence (ICTAI 1996), pp. 286-289, IEEE Computer Society, 1996.
3. Chmeiss, A., Jégou, P.: Efficient Constraint Propagation with Good Space Complexity. Proceedings of the Second International Conference on Principles and Practice of Constraint Programming (CP 1996), pp. 533-534, LNCS 1118, Springer, 1996.
4. Chvátal, V.: Tough Graphs and Hamiltonian Circuits. Discrete Mathematics 306, Volume 10-11, pp. 910-917, Elsevier, 2006.
5. Cook, S. A.: The Complexity of Theorem Proving Procedures. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971), pp. 151-158, ACM Press, 1971.
6. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.: Introduction to Algorithms (Second edition), MIT Press and McGraw-Hill, 2001.
7. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers, 2003.
8. Dowling, W., Gallier, J.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. Journal of Logic Programming, Volume 1 (3), 267-284, Elsevier Science Publishers, 1984.
9. Garey, M. R., Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP Completeness. W. H. Freeman & Co., 1979, ISBN: 978-0716710455.
10. Han, C. C., Lee, C. H.: Comments on Mohr and Henderson's Path Consistency Algorithm. Artificial Intelligence, Volume 36(1), pp. 125-130, Elsevier, 1988.
11. Holger, H. H., Stützle, T.: SATLIB: An Online Resource for Research on SAT. Proceedings of Theory and Applications of Satisfiability Testing, 4th International Conference (SAT 2000), pp.283-292, IOS Press, 2000, http://www.satlib.org, [October 2010].
12. Jackson, P., Sheridan, D.. Clause Form Conversions for Boolean Circuits. Theory and Applications of Satisfiability Testing, 7th International Conference (SAT 2004), Revised Selected Papers, pp. 183–198, Lecture Notes in Computer Science 3542, Springer 2005.
13. Mohr, R., Henderson, T. C.: Arc and Path Consistency Revisited. Artificial Intelligence, Volume 28 (2), 225-233, Elsevier Science Publishers, 1986.
14. Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences, Volume 7, pp. 95-132, Elsevier, 1974.
15. Rossi, F., Dhar, V., Petrie, C.: On the Equivalence of Constraint Satisfactions Problems. Proceedings of the 9th European Conference on Artificial Intelligence (ECAI 1990), pp. 550-556, 1990.
16. Surynek, P.: Making Path Consistency Stronger for SAT. Proceedings of the Annual ERCIM Workshop on Constraint Solving and Constraint Logic Programming (CSCLP 2008), ISTC-CNR, 2008.
17. Walsh, T.: SAT vs. CSP. Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming, 441-456, LNCS 1894, Springer Verlag, 2000.