Exploiting Global Properties in Path-Consistency Applied on SAT

Pavel SURYNEK[a]
[a] *Charles University in Prague*
Faculty of Mathematics and Physics
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. Performed experiments show that the new consistency outperforms the standard path-consistency in terms of the inference strength as well.

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

Introduction and Motivation

A method how to increase the inference strength of path-consistency [14, 15] is described in this paper. It combines the standard path-consistency on the literal encoding model [19] of the given Boolean satisfiability (SAT) instance [5] with global properties calculated from the instance. The existence of a path in a certain 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. Unfortunately, the problem of checking the existence of a path according to augmented requirements turned out to be NP-complete [17]. Hence, various relaxations that still preserve the inference strength of augmented variants above the level of the standard path-consistency were proposed. An evaluation of the benefit of the new path-

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.

consistency has been done through a simple SAT preprocessing tool which is based on it. The result is that preprocessing can reduce the number of decisions of the SAT solver as well as the overall runtime significantly on a set of representative instances.

1. 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 have 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 [16]. The key concept of *path-consistency* [15] 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*) [13] 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, 10]. 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 $v(x_0) = d_0 \wedge v(x_K) = d_K$ 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 [14, 15]. 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 [11] 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* [19], 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 $\mathbb{D} = \bigcup_{i=1}^{n} (\bigcup_{j=1}^{K_i} \{\overline{l_j^i}\})$ 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 $D(\sigma_i) = \bigcup_{j=1}^{K_i} \{\overline{l_j^i}\}$; 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 [19].

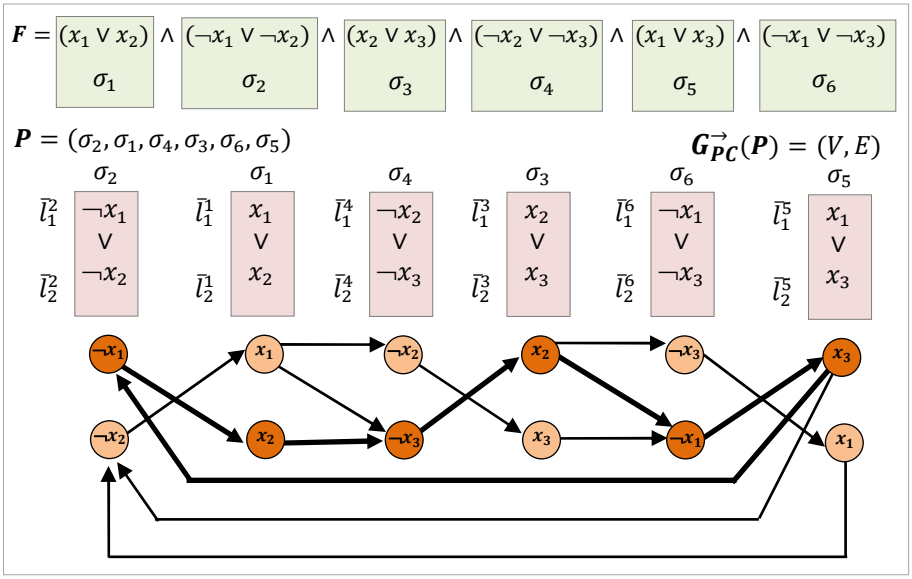

Figure 1. *An illustration of path-consistency in the CSP model of a SAT problem.* The SAT problem represented by a formula F shown here is a representation of the requirement of selecting an odd number of variables from every of the following sets to be true: $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_2, x_3\}$. 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 $G_{PC}^{\rightarrow}(P) = (V, E)$, in which path-consistency can be interpreted as the existence of paths, is defined as follows: let $V = \bigcup_{j=0}^{K} D(\sigma_{i_j})$ be a set of vertices and if $(\bar{l}_{j_1}^{i_j}, \bar{l}_{j_2}^{i_{(j+1) \bmod K}}) \in R^{c(\sigma_{i_j}, \sigma_{i_{(j+1) \bmod K}})}$ then a directed edge $(\bar{l}_{j_1}^{i_j}, \bar{l}_{j_2}^{i_{(j+1) \bmod K}})$ is included into the set of edges E. A pair of values $\bar{l}_{j_1}^{i_0} \in D(\sigma_{i_0})$ and $\bar{l}_{j_2}^{i_K} \in D(\sigma_{i_K})$ is path-consistent with respect to the path P if there is an edge $(\bar{l}_{j_1}^{i_K}, \bar{l}_{j_1}^{i_0})$ in $G_{PC}^{\rightarrow}(P)$ and there exists a path from the vertex $\bar{l}_{j_1}^{i_0}$ to the vertex $\bar{l}_{j_2}^{i_K}$ in $G_{PC}^{\rightarrow}(P)$. The graph $G_{PC}^{\rightarrow}(P)$ will be called a *graph interpretation* of path-consistency – see Figure 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 Figure 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 $\{\vec{l}^{i_j}_{j_1}, \vec{l}^{i_k}_{j_2}\}$ is added to E if $(\vec{l}^{i_j}_{j_1}, \vec{l}^{i_k}_{j_2}) \in R^{c(\sigma_{i_j}, \sigma_{i_k})}$; and all the edges $\{\vec{l}^{i_j}_{j_1}, \vec{l}^{i_j}_{j_2}\}$ 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 $G^{\rightarrow}_{PC}(P) = (V, E)$ 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 $L_1, L_2, \dots, L_M$ called *layers* (that is, $\bigcup_{i=1}^{M} \hat{L}_i = V$ and $\hat{L}_i \cap \hat{L}_j \; \forall i, j \in \{1,2,\dots,M\} \wedge i \neq j$; where denotes the union of the sequence $\hat{A}$, that is $\hat{A} = \bigcup_{i=1}^{n} \{a_i\}$ for $A = [a_1, a_2, \dots, a_n]$). 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 [17]. 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 [17].

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 [17]. 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 $G_{PC}^{\rightarrow}(P)$ 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 $G_{PC}^{\rightarrow}(P)$; 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 Figure 2 for illustration).

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 $G_{PC}^{\rightarrow}(P)$. A pseudo-code for the above calculation is given in [18].

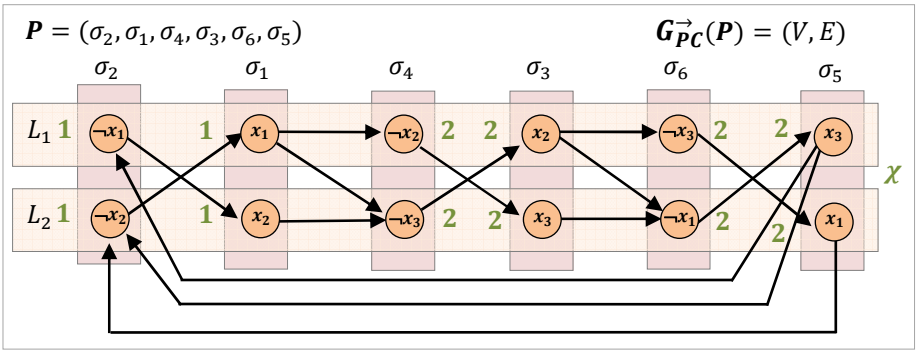

Figure 2. *An illustration of modified path-consistency in the CSP model of a SAT problem.* The maximum size of the intersection of the constructed path with vertices preceding the given vertex (including) in its layer is calculated using constraints for each vertex - these maximum sizes are denoted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

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. A Note on Modified Path-Consistency Enforcing Algorithms

Several essential steps have to be done 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).

The detailed description of how to perform mentioned essential steps is given in [18]. Let us briefly note that layer decomposition is made greedily while the most constrained parts of the graph interpretation are preferably included into the currently constructed layer. To estimate maximum sizes of intersections with layers, vertices in the given layer are ordered with correspondence to the selected path. Then vertices of the layer are processed in the given ordering. It is checked for every vertex whether it can increase the size of the intersection.

Having maximum sizes of intersections, it is possible to check the modified path-consistency according to the computation described in the previous section. The pseudo-code of consistency checking algorithm is given in [18]. Let us finally note that worst-case time complexity of all the algorithms necessary to carry out consistency checking is polynomial.

6. Experimental Evaluation

We have performed an experimental evaluation of the modified path-consistency on a set of Boolean satisfiability instances from the Satisfiability Library (SATLib) [12] and from [1]. The evaluation is aimed on the comparison of preprocessing abilities of the standard path-consistency and the modified version. Results are reported in Table 1 (types of instances with non-trivial behavior of consistencies are reported only).

We have measured the number of binary clauses inferred by the standard path-consistency and by the modified path-consistency. The runtime needed for this computation has been measured as well. It is possible to conclude that modified path consistency typically infers substantially more binary clauses than the standard version. This is especially true for instances with relatively clustered graph interpretations (*fpga*, *hole*, and *chnl* instances). Next, we have evaluated whether the application of consistencies as a preprocessing technique can reduce the number of decisions made by the SAT solver. The Minisat2 solver [9] has been used for this test. It is possible to observe that modified-path consistency can reduce the number of decisions of the solver significantly while it makes greater reduction of decisions than the standard path consistency in most cases.

Table 1. *Comparison of the standard path-consistency (PC) and modified path-consistency (mPC)*. The number of newly inferred clauses by both tested consistencies and runtime[*] are reported. The comparison of preprocessing abilities is shown in terms of the number of decisions made by the Minisat2 SAT solver of instances augmented by inferred clauses.

SAT instance	Instance characteristics		Inferred clauses				Minisat2 decisions		
	Variables	Clauses	PC	mPC	Runtime PC (*sec.*)	Runtime mPC (*sec.*)	Original	PC	mPC
ais6	61	581	0	37	0	0.22	27	27	2
ais12	265	5666	0	216	1.94	14.58	117	117	1
anomaly	48	261	94	103	0.09	0.14	5	1	1
hole6	42	133	0	42	0.01	0.04	1777	1777	1
hole7	56	204	0	56	0.02	0.14	10123	10123	1
hole8	72	297	0	72	0.04	0.32	40554	40554	1
hole9	90	415	0	90	0.07	0.64	202160	202160	1
par8-1-c	64	254	5	10	0.04	0.08	12	3	2
par8-2-c	68	270	0	2	0.04	0.08	17	17	5
par8-3-c	75	298	0	1	0.05	0.09	43	43	9
par8-4-c	67	266	0	2	0.04	0.08	13	13	6
par16-1-c	317	1264	0	4	0.26	0.39	1729	1729	2
par16-2-c	349	1292	0	4	0.27	0.4	5993	5993	2
par16-3-c	334	1332	0	4	0.27	0.39	4280	4280	2
par16-4-c	324	1292	0	4	0.26	0.39	338	338	2
chnl10_11	220	1122	0	220	0.23	2.38	N/A	N/A	1
chnl10_12	240	1344	0	240	0.25	2.6	N/A	N/A	1
chnl10_13	260	1586	0	260	0.27	2.82	N/A	N/A	1
chnl11_12	264	1476	0	264	0.36	4.18	N/A	N/A	1
chnl11_13	286	1742	0	286	0.39	4.54	N/A	N/A	1
chnl11_20	440	4220	0	440	0.63	7.05	N/A	N/A	1
fpga10_8_sat	120	448	0	80	0.14	1.23	264	264	1
fpga10_9_sat	135	549	0	90	0.19	1.67	250	250	1
fpga12_8_sat	144	560	0	96	0.25	2.6	390	390	1
fpga12_9_sat	162	684	0	108	0.31	3.26	383	383	1
fpga12_11_sat	198	968	0	132	0.5	5.19	421	421	1
fpga12_12_sat	216	1128	0	144	0.63	6.5	403	403	1
fpga13_10_sat	195	905	0	130	0.49	5.43	499	499	1
fpga13_12_sat	234	1242	0	156	0.76	8.31	335	335	1
fpga10_12_uns_rcr	240	1344	0	240	0.25	2.61	N/A	N/A	1
fpga10_13_uns_rcr	260	1586	0	260	0.28	2.82	N/A	N/A	1
fpga10_15_uns_rcr	300	2130	0	300	0.32	3.27	N/A	N/A	1
fpga10_20_uns_rcr	400	3840	0	400	0.45	4.37	N/A	N/A	1
fpga11_11_uns_rcr	264	1476	0	264	0.36	4.18	N/A	N/A	1
fpga11_12_uns_rcr	286	1742	0	286	0.39	4.54	N/A	N/A	1
fpga11_13_uns_rcr	308	2030	0	308	0.43	4.93	N/A	N/A	1
fpga11_15_uns_rcr	330	2340	0	330	0.46	5.26	N/A	N/A	1
huge	459	7054	359	399	8.52	13.66	41	2	2
jnh1	100	850	37	37	1.1	14.8	48	2	1
jnh2	100	850	157	157	1.04	11.62	12	1	1
jnh3	100	850	15	16	1.07	13.26	92	1	1
jnh4	100	850	4	6	1.17	11.57	26	6	7
logistics.a	828	6718	344	371	1.71	7.3	1731	1	1
medium	116	953	73	83	0.58	0.92	11	2	2
s3-3-3-8	912	8356	3	10	0.56	1.27	10871	11932	17

7. Conclusions 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 –

[*] All the experiments were run on an dual AMD Opteron 1600 MHz, with 1GB RAM under Mandriva Linux 10.1, 32-bit edition; gcc 3.4.3 with optimization level –o3 was used for compilation.

namely, the size of the intersection of the path with certain sets called layers is restricted. The experimental evaluation has shown that it is possible to use the modified path-consistency as a preprocessing tool for SAT solving.

For future work we plan to enhance the modified path-consistency in terms of inference strength as well as in terms of efficiency of its propagation algorithms. Currently, the biggest limitation of the proposed concept is absence of a fine tuned implementation of propagation algorithms for modified path-consistency which precludes making experimental evaluation on large satisfiability instances. Our future work will be mainly targeted on overcoming this limitation.

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, [March 2011] .

[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 proposi-tional Horn formulae. Journal of Logic Programming, Volume 1 (3), 267-284, Elsevier Science Publishers, 1984.

[9] Eén, N., Sörensson, N.: MiniSat — A SAT Solver with Conflict-Clause Minimization. Poster, 8th International Conference on Theory and Applications of Satisfiability Testing (SAT 2005), 2005.

[10] 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.

[11] 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.

[12] Holger, H. H., Stützle, T.: SATLIB: An Online Resource for Research on SAT. Proceed-ings of Theory and Applications of Satisfiability Testing, 4th International Conference (SAT 2000), pp.283-292, IOS Press, 2000, http://www.satlib.org, [March 2011].

[13] 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.

[14] Mohr, R., Henderson, T. C.: Arc and Path Consistency Revisited. Artificial Intelligence, Volume 28 (2), 225-233, Elsevier Science Publishers, 1986.

[15] Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences, Volume 7, pp. 95-132, Elsevier, 1974.

[16] 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.

[17] Surynek, P.: Making Path Consistency Stronger for SAT. Proceedings of the Annual ER-CIM Workshop on Constraint Solving and Constraint Logic Programming (CSCLP 2008), ISTC-CNR, 2008.

[18] Surynek, P.: An Adaptation of Path Consistency for Boolean Satisfiability: a Theoretical View of the Concept, Proceedings of the Annual ERCIM Workshop on Constraint Solving and Constraint Logic Programming, 2010 (CSCLP 2010), pp. 16-30, Berlin, Germany, Fraunhofer FIRST, 2010.

[19] 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.