An Adaptation of Path Consistency for Boolean Satisfiability

Pavel Surynek[*]
Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00 Praha 1, 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 extended abstract. The concept of path-consistency known from the constraint programming paradigm (CP) is adapted for SAT. The adaptation consists in increasing inference strength of the consistency on the literal encoding model of SAT.

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

1 Introduction and Motivation

A method how to increase the inference strength of *path-consistency* [11], [12] will be described. It combines the standard path-consistency on the *literal encoding* model [14] of the given *Boolean satisfiability* (SAT) instance [4] 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. Unfortunately, the problem of checking the existence of a path according to augmented requirements turned out to be *NP*-complete [13]. 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) need to be established first to make reasoning about path-consistency in context of SAT possible.

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

[*] This work is supported by The Czech Science Foundation under the contract number 201/09/P318 and by The Ministry of Education, Youth and Sports, Czech Republic under the contract number MSM 0021620838.

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 [6]. The key concept of *path-consistency* [12] that is addressed in this extended abstract 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*) [10] 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 [4], [8]. 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.

The standard definition of *path-consistency* in CSP will be recalled before the augmented version is 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 [11], [12]. 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 [9] and PC-6 [1], [1]. They differ in representation of auxiliary data structures and 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.

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* [14], 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.

3 Standard Path-consistency in SAT

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: $X = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$; 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 $i_1, i_2 \in \{1,2,\ldots,N\} \wedge i_1 \neq i_2$ 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 [14].

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.

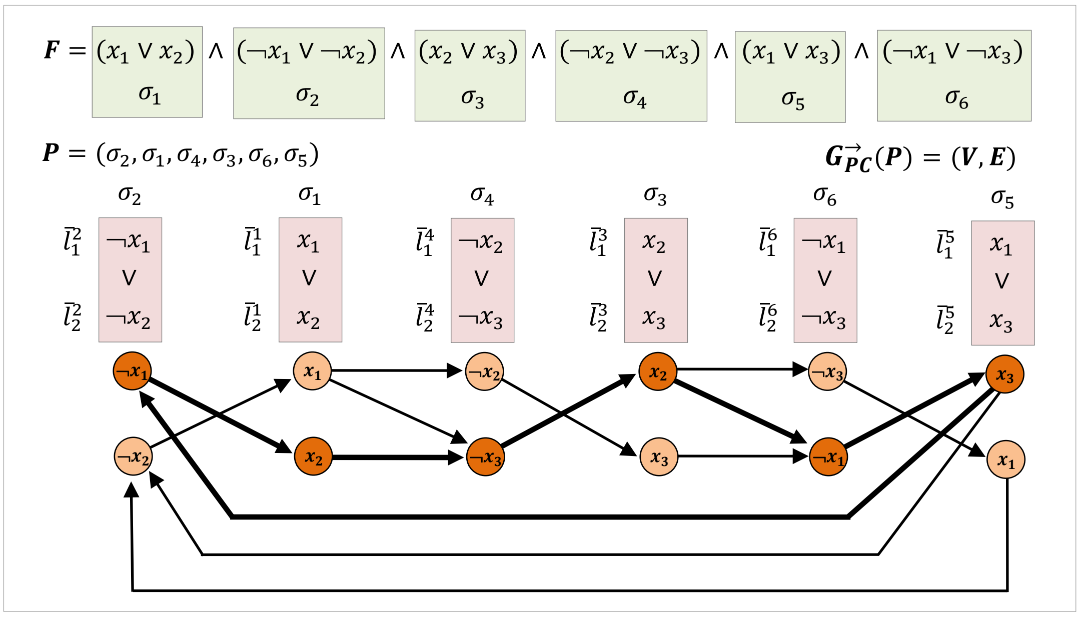
$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$
σ_1 σ_2 σ_3 σ_4 σ_5 σ_6
$P = (\sigma_2, \sigma_1, \sigma_4, \sigma_3, \sigma_6, \sigma_5)$
$G_{\vec{PC}}(P) = (V, E)$
$\bar{l}_1^2$ $\neg x_1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

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).

More precisely, let $P = (\sigma_{i_0}, \sigma_{i_1}, \ldots, \sigma_{i_K})$ with $i_j \in \{1,2,\ldots,N\}$ for $j = 0,1,\ldots,K$ 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 $(\bar{l}_{j_1}^{i_j}, \bar{l}_{j_2}^{i_{(j+1) \bmod K}}) \in R^{c(\sigma_{i_j}, \sigma_{i_{(j+1) \bmod K}})}$ then a directed edge $(\bar{l}_{j_1}^{i_j}, \bar{l}_{j_2}^{i_{(j+1) \bmod K}})$ is included into 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 $G_{\vec{PC}}(P)$ and there exists a path from the vertex $\bar{l}_{j_1}^{i_0}$ to the vertex $\bar{l}_{j_2}^{i_K}$ in $G_{\vec{PC}}(P)$. The graph $G_{\vec{PC}}(P)$ 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 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 $\{\vec{l}_{j_1}^{i_j}, \vec{l}_{j_2}^{i_k}\}$ 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 $j = 1,2,\ldots,N$ and $j_1, j_2 = 1,2,\ldots,K_j \wedge j_1 \neq j_2$. 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 A Modified Version of Path-consistency

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.

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 $G_{PC}^{\rightarrow}(P) = (V, E)$ 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 subsets $L_1, L_2, \ldots, L_M$ called *layers* (that is, $\bigcup_{i=1}^{M} L_i = V$ and $L_i \cap L_j$ $\forall i, j \in \{1,2,\ldots,M\} \wedge i \neq j$). 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 process of determining the maximum size of the intersection is out of scope of this work (see [13] for further details). Nevertheless, it can be briefly stated that it can be done over the corresponding auxiliary constraint graph $G_{CSP}(P)$ by decomposing its vertices 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 [13].

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 [13]. 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 [3]. This is known to be an *NP*-complete problem. Hence, it is not tractable to find a path that satisfies defined requirements exactly. Moreover, initial experiments showed that 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 [5] 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. 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 $L_1, L_2, \ldots, L_M$ be layers of $G_{\vec{PC}}(P)$; let a function $\chi: V \longrightarrow \mathbb{N}$ defines requirements on maximum sizes of intersections as follows: $\chi(\vec{l}_j^{i_k})$ is the maximum size of the intersection of the constructed path with a set of vertices $\{\vec{l}_m^{i_l} | \vec{l}_m^{i_l} \in L_n \wedge l \leq j\}$ with $\vec{l}_j^{i_k} \in L_n$ and $n \in \{1,2,\ldots,M\}$ imposed by constraints. 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.

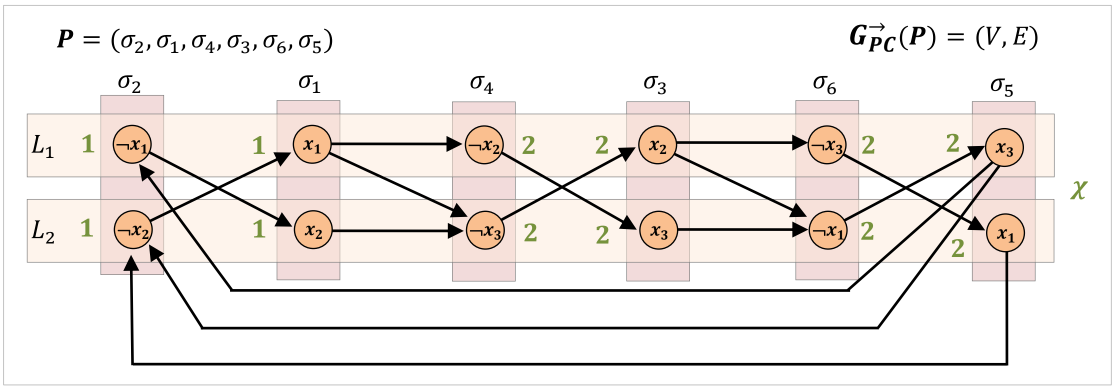

Figure 2. *An illustration of modified path-consistency in the CSP model of a SAT problem.* The maximum size of the intersection of the constructed path with vertices preceding the given vertex (including) in its layer is calculated using constraints for each vertex - these maximum sizes are denoted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

The function χ can calculated from the decomposition of the graph $G_{CSP}(Q)$ into vertex disjoint stable subsets where Q is a prefix of the path P. This is almost the same approach as in the case of the initial augmentation. A more detailed discussion on this issue can be found in [13].

Contrary to the initial augmentation of path-consistency, a simple relaxation of requirements on paths being constructed in modified path-consistency defined by a modified algorithm for finding single source shortest paths in a graph [5] now provides stronger inference than the standard path-consistency [13]. The algorithm maintains an M-tuple of integers in each vertex of the graph $G_{\vec{PC}}(P)$ which expresses the minimum size of the intersection of paths with individual layers ending in the given vertex. In other words, every path ending in the given vertex must visit individual layers the number of times given by the M-tuple. Paths whose number of visits exceeds the maximum allowed size of the intersection with layers are ruled out.

5 Future Work and Conclusion

The approach for adapting path-consistency for Boolean satisfiability (SAT) used throughout this work has three conceptual stages. First, an appropriate CSP model of SAT is chosen – literal encoding scheme has been used. Then some stronger requirement than it is defined by the standard path-consistency is imposed on paths being constructed. Finally, an algorithm that finds paths according to the defined stronger requirement but in a relaxed manner is constructed. The eventual relaxation has been compelled by the fact that finding paths that conform to stronger requirements exactly turned out to be intractable.

The particular problem is thus to balance the strength of requirements defined in the second stage and weakness of the relaxation in the last stage. The goal is to obtain efficient consistency with the inference strength above the standard path-consistency. This goal has been partially achieved in this work. Nevertheless, the room for improvements of presented ideas is still great. Namely, some effort should be devoted to finding strongly inferring but tractable relaxations of defined requirements on paths. An experimental evaluation of the proposed concept should be made as well.

References

[1] A. Chmeiss, P. Jégou: 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.

[2] A. Chmeiss, P. Jégou: 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.

[3] V. Chvátal: *Tough Graphs and Hamiltonian Circuits.* Discrete Mathematics 306, Volume 10-11, pp. 910-917, Elsevier, 2006.

[4] S. A. Cook: 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.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein: *Introduction to Algorithms* (Second edition), MIT Press and McGraw-Hill, 2001.

[6] R. Dechter: Decomposing a Relation into a Tree of Binary Relations. *Journal of Computer and System Sciences (JCSS)*, Volume 41(1), pp. 2-24, Elsevier, 1990.

[7] R. Dechter: Constraint Processing. Morgan Kaufmann Publishers, 2003.

[8] M. R. Garey and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP Completeness.* W. H. Freeman & Co., 1979, ISBN: 978-0716710455.

[9] C. C. Han, C. H. Lee: Comments on Mohr and Henderson's Path Consistency Algorithm. *Artificial Intelligence*, Volume 36(1), pp. 125-130, Elsevier, 1988.

[10] P. Jackson, D. Sheridan. 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.

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

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

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

[14] T. Walsh: SAT vs. CSP. Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming, 441-456, LNCS 1894, Springer Verlag, 2000.