Revisiting Neighborhood Inverse Consistency on Binary CSPs

Robert J. Woodward, Shant Karakashian, Berthe Y. Choueiry, Christian Bessiere

Revisiting Neighborhood Inverse Consistency on Binary CSPs

Robert J. Woodward[1], Shant Karakashian[1], Berthe Y. Choueiry[1], and Christian Bessiere[2]

[1] Constraint Systems Laboratory, University of Nebraska-Lincoln, USA
{rwoodwar|shantk|choueiry}@cse.unl.edu
[2] LIRMM-CNRS, University of Montpellier, France
bessiere@lirmm.fr

Abstract. Our goal is to investigate the definition and application of strong consistency properties on the dual graphs of binary Constraint Satisfaction Problems (CSPs). As a first step in that direction, we study the structure of the dual graph of binary CSPs, and show how it can be arranged in a triangle-shaped grid. We then study, in this context, Relational Neighborhood Inverse Consistency (RNIC), which is a consistency property that we had introduced for non-binary CSPs [17]. We discuss how the structure of the dual graph of binary CSPs affects the consistency level enforced by RNIC. Then, we compare, both theoretically and empirically, RNIC to Neighborhood Inverse Consistency (NIC) and strong Conservative Dual Consistency (sCDC), which are higher-level consistency properties useful for solving difficult problem instances. We show that all three properties are pairwise incomparable.

1 Introduction

Enforcing consistency properties on Constraint Satisfaction Problems (CSPs) allows us to effectively prune the exponential search space of these problems, and constitutes one of the most significant contributions of Constraint Programming (CP). While lower level consistencies, such as Arc Consistency (AC) [15], are often sufficient for solving easy problems, solving difficult problems often requires enforcing higher consistency levels. To facilitate solving difficult CSPs, we propose, as a research goal, to investigate the effectiveness of enforcing higher levels of consistency on the *dual* graphs of binary CSPs.

To this end, we first study the structure of the *dual graph* of binary CSPs and show that it exhibits an interesting triangle-shaped grid that, in general, may affect the 'level' of the consistency property enforced and the operation of the algorithms for enforcing it. Then, we focus our attention on Relational Neighborhood Inverse Consistency (RNIC) [17], a consistency property that we had proposed and evaluated as an extension of Neighborhood Inverse Consistency (NIC) introduced by Freuder and Elfe [8]. We show how the structure of the dual graph of a binary CSP affects the consistency level enforced by RNIC,

characterizing the conditions where RNIC cannot be stronger than another relational consistency property that we had defined in [11] when both properties are enforced on binary CSPs. In order to characterize the effectiveness of RNIC on binary CSPs despite the identified limitation imposed by the structure, we turn our attention back to 'strong' consistency properties defined for binary CSPs, and compare RNIC, both theoretically and empirically, to NIC and strong Conservative Dual Consistency (sCDC) [14], showing that all three properties are incomparable.

This paper is structured as follows. Section 2 reviews background information about CSPs. Section 3 discusses the structure of the dual graph of a binary CSP, mainly, that it is a triangle-shaped grid. Section 4 discusses RNIC on binary CSPs. Section 5 reviews the state of the art in relational consistency. Section 6 discusses experimentally the filtering power of NIC, sCDC, and RNIC on binary CSPs. Finally, Section 7 concludes this paper.

2 Background

A Constraint Satisfaction Problem (CSP) is defined by $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ where $\mathcal{V}$ is a set of variables, $\mathcal{D}$ is a set of domains, and $\mathcal{C}$ is a set of constraints. Each variable $V_i \in \mathcal{V}$ has a finite domain $D_i \in \mathcal{D}$, and is constrained by a subset of the constraints in $\mathcal{C}$. Each constraint $C_i \in \mathcal{C}$ is specified by a relation R_i defined on a subset of the variables, called the scope of the relation and denoted $scope(R_i)$. Given a relation R_i, a tuple $\tau_i \in R_i$ is a vector of allowed values for the variables in the scope of R_i. Solving a CSP corresponds to finding an assignment of a value to each variable such that all the constraints are satisfied. The dual encoding of a CSP, $\mathcal{P}$, is a binary CSP whose variables are the relations of $\mathcal{P}$, their domains are the tuples of those relations, and the constraints enforce *equalities* over the shared variables.

2.1 Graphical representations

A binary CSP is graphically represented by its *constraint graph* where the vertices are the variables of the CSP and the edges represent the relations [6]. Neigh(V_i) denotes the set of variables adjacent to a variable V_i in the constraint graph. The *dual graph* of a CSP is a graph whose vertices represent the relations of the CSP, and whose edges connect two vertices corresponding to relations whose scopes overlap. Neigh(R_i) denotes the set of relations adjacent to a relation R_i in the dual graph. Janssen et al. [10] and Dechter [6] observed that, in the dual graph, an edge between two vertices is *redundant* if there exists an alternate path between the two vertices such that the shared variables appear in every vertex in the path. Redundant edges can be removed without modifying the set of solutions. Janssen et al. introduced an efficient algorithm for computing the *minimal dual graph* [10]. Many minimal graphs may exist, but all are guaranteed to have the same number of edges. Figure 1 shows the constraint, dual graph, and a minimal dual graph of a small CSP.

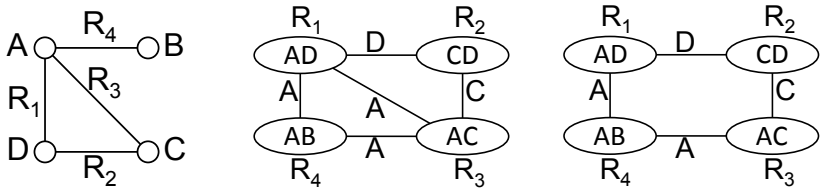

Fig. 1. A constraint graph, dual graph, and minimal dual graph.

2.2 Consistency properties & algorithms

CSPs are in general $\mathcal{NP}$-complete and solved by search. To reduce the severity of the combinatorial explosion, they are usually 'filtered' by enforcing a given local consistency property [2]. One common property is Arc Consistency (AC). A CSP is arc consistent iff for every binary constraint, any value in the domain of one variable can be extended to the domain of the other variable while satisfying the constraint, and vice versa. The more difficult the CSP, the larger is its search space, and the more advantageous it is to enforce consistency properties of higher levels. In fact, Freuder provided a sufficient condition for guaranteeing a backtrack-free search that links the level of consistency to a structural parameter of the CSP [7]. However, enforcing higher-level consistencies may add constraints and modify the structure of the problem. For this reason, we focus, in this paper, on higher-level consistency properties for binary CSPs that do not modify the graphical representations of a problem.

Freuder and Elfe introduced Neighborhood Inverse Consistency (NIC), which ensures that each value in the domain of a variable can be extended to a solution of the subproblem induced by the variable and all the variables in its neighborhood in the constraint graph [8]. NIC is defined on binary CSPs and the algorithm for enforcing it operates on the constraint graph. RNIC ensures that any tuple in any relation can be extended in a consistent assignment to all the relations in its neighborhood in the dual graph. Enforcing NIC (RNIC) does not modify the constraint graph (dual graph). Further, it exploits the structure of the problem to focus the pruning on where a variable (relation) most tightly interacts with the problem. Thus, the topology of the constraint graph (dual graph) of a problem can determine the level of consistency enforced.

As extensions to RNIC, we also proposed wRNIC, triRNIC, and wtriRNIC, which modifies the structure of the dual graph but not the CSP solution set [17]. wRNIC is defined on a minimal dual graph; triRNIC is defined on a triangulated dual graph;[3] and wtriRNIC is defined on a triangulation of a minimal dual graph. We gave a selection strategy, selRNIC, for automatically determin-

[3] Graph triangulation adds an edge (a chord) between two non-adjacent vertices in every cycle of length four or more [9]. While minimizing the number of edges added by the triangulation process is NP-hard, MinFill is an efficient heuristic commonly used for this purpose [12, 6].

ing which RNIC variation to use based on the density of the dual graph.[4] We showed that selRNIC statistically dominates all other RNIC properties. We have also studied m-wise consistency, which we denoted R($*$,m)C, and proposed the first algorithm for enforcing it [11]. R($*$,m)C ensures that every tuple in every relation can be extended in a consistent assignment to every combination of $m-1$ relations in the problem. In this paper, we use the knowledge about the structure of the dual graph of binary CSPs to formally characterize the relationship between RNIC and R($*$,m)C.

Strong consistency properties for binary CSPs that do not affect the topology of the constraint graph have been carefully reviewed, studied, and compared to each others [5, 14]. Such properties include maxRPC [3], SAC [4], CDC [13], and, the strongest of them all, sCDC [14]. Further, Lecoutre et al. show that, on binary CSPs, strong Conservative Dual Consistency (sCDC) is equal SAC+CDC [14].[5] While NIC was shown to be incomparable to SAC [5], its relationship to sCDC has not yet been addressed [13]. In this paper, we complete the comparison of NIC, RNIC, and sCDC, and show that they are, both theoretically and empirically, pairwise incomparable. Thus, our results contribute to the characterization of strong consistency properties for binary CSPs.

When enforcing a relational consistency property, we always terminate the process by filtering the variables' domains by projecting on them the filtered relations. For RNIC, we call the resulting consistency property RNIC+DF (domain filtering). To compare a consistency property p_i defined on the relations of a CSP to another one defined on the variables, we always consider p_i+DF. To compare two consistency properties p and p', we use the following terminology [4]:

- p is *stronger* than p' if, in any CSP where p holds, p' also holds.
- p is *strictly stronger* than p' if p is stronger than p' and there exists at least one CSP in which p' holds but p does not.
- p and p' are *equivalent* when p is stronger than p' and vice versa.
- Finally, p and p' are *incomparable* when there exists at least one CSP in which p holds but p' does not, and vice versa.

In practice, when a consistency property is stronger (respectively, weaker) than another, enforcing the former never yields less (respectively, more) pruning than enforcing the latter on the same problem.

3 Structure of the Dual Graph of Binary CSPs

The structure of the dual graph determines the neighborhoods of its vertices (i.e., CSP relations) and may affect the level of relational consistency that can

[4] The density of a graph $G = (V, E)$ is considered to be $\frac{2|E|}{|V|(|V|-1)}$.

[5] Singleton Arc Consistency (SAC) ensures that a binary CSP remains AC after instantiating any single variable to any value in the variable's domain [4]. Conservative Dual Consistencies (CDC) ensures that for every instantiation of two variables in the scope of some constraint, $\{(V_i, a), (V_j, b)\}$, that b remain in the domain of V_j in the arc-consistent CSP where a is assigned to V_i and vice versa [13].

be enforced on the CSP. We first discuss the case of a binary CSP with a complete constraint graph, showing that the structure of its dual graph can be arranged in a triangle-shaped grid. We show that redundant edges can be removed in a way to maintain the grid structure. We then discuss the case of a binary CSP with a non-complete constraint graph, and show that its dual graph can also be arranged in a triangle-shaped grid but with fewer vertices and a less regular shape than that of a CSP with complete constraint graph. We discuss the effects of the dual-graph structure on RNIC in Section 4.

3.1 Binary CSP with a complete constraint graph

Theorem 1. *The* $\frac{n(n-1)}{2}$ *vertices of the dual graph of a binary CSP of* n *variables whose constraint graph is complete such as the one shown in Figure 2 (i.e., forms a clique of* n *vertices,* K_n*), can be arranged in an* $(n-1) \times (n-1)$ *triangle-shaped grid where:*

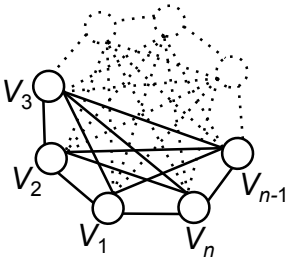

Fig. 2. A complete constraint graph of n vertices.

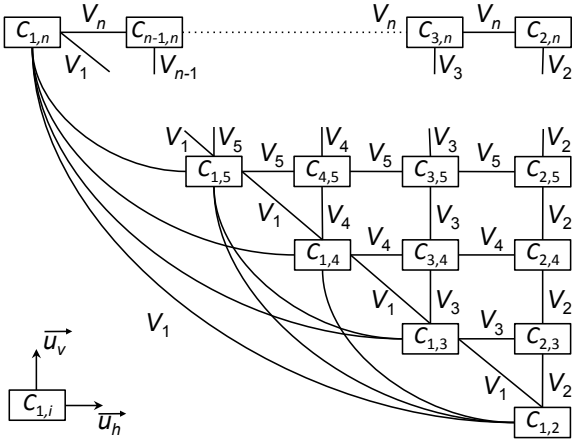

Fig. 3. Dual graph corresponding to the CSP in Figure 2.

1. *The* $n-1$ *vertices on the diagonal of the triangle correspond to the constraints over the variable* V_1*. They are denoted* $C_{1,i}$ *where* $i \in [2, n]$ *and completely connected. The connecting edges are labeled with* V_1*.*
2. *The* $n-1$ *vertices corresponding to the constraints over variable* $V_{i \geq 2}$ *are located along the path in the grid shown in Figure 4 and specified as follows:*
   - *Considering the coordinate system defined by the horizontal and vertical unit vectors* $\boldsymbol{u_h}$*,* $\boldsymbol{u_v}$ *and centered on* $C_{1,i}$*,*
   - $i-2$ *vertices are lined up along the horizontal axis* $\boldsymbol{u_h}$*, and*
   - $n-i$ *vertices are lined up along the vertical axis* $\boldsymbol{v_h}$*.*

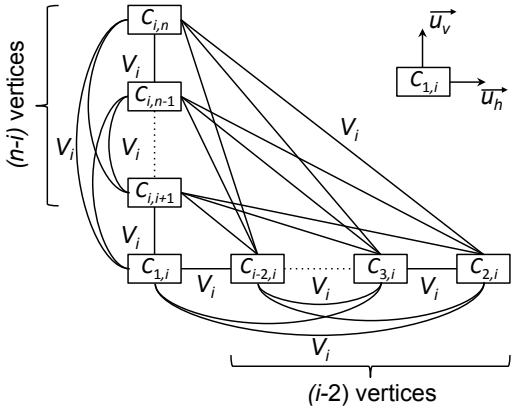

Fig. 4. The path for the constraints over variables $V_{i\geq 2}$ of the grid of Figure 3.

– *Those $n-1$ vertices are completely connected, and the connecting edges are labeled with V_i. (For the sake of clarity, Figure 3 does not show all the edges of the dual graph: only all the edges labeled V_1 are shown on the diagonal of the grid.)*

Proof: See Appendix A. □

Corollary 1. *After the removal of redundant edges, the dual graph of a binary CSP of n variables whose constraint graph is complete can be arranged in a $(n-1) \times (n-1)$ triangle-shaped grid, where every CSP variable annotates the edges of a chain of length $n-2$.*

Proof: See Appendix B. □

Because redundancy removal is not unique, not all minimal dual graphs necessarily yield a triangle-shaped grid as we show using a counter-example. One possible minimal dual graph for the complete constraint graph of five vertices of Figure 5 is shown in Figure 6. In this example, there is a cycle of size six in the dual graph, indicated by the bold lines in Figure 6. Thus, the dual graph is not a grid. Further, the variable V_2 does not annotate a chain, but a star, as indicated by the dotted lines in the dual graph.

3.2 Binary CSP with a non-complete constraint graph

In a binary CSP with a non-complete constraint graph, the dual graph can be thought of as the complete binary constraint graph with some missing vertices. Because, in the dual graph of any complete constraint graph, all the vertices corresponding to the constraints that apply to a given CSP variable are completely connected, it is always possible, even in the case of a CSP with a non-complete constraint graph, to form, in its corresponding dual graph, a chain connecting vertices related to the same variable. However, the length of such a chain may

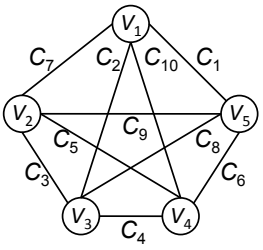

Fig. 5. A complete graph with five variables.

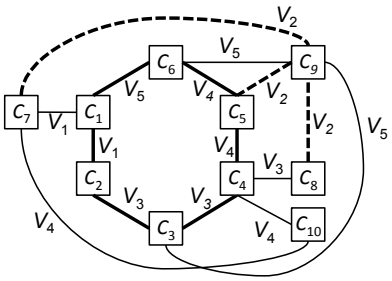

Fig. 6. A minimal dual graph of Figure 5, which does not form a grid.

be less than $n-2$. Thus, the triangle-shaped grid can be preserved. For example, consider the binary CSP with $n = 5$ variables given in Figure 7. A minimal dual

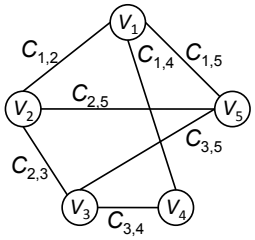

Fig. 7. A constraint graph with five variables.

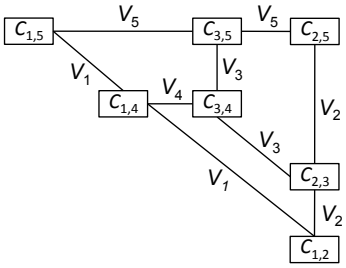

Fig. 8. The minimal dual graph of Figure 7.

graph for that binary CSP is given in Figure 8, which was constructed from the dual graph for the complete CSP by removing the vertices corresponding to the constraints that are not in the CSP. Again, because the minimal dual graph is not unique, there exists minimal dual graphs that do not favor the chains, and thus, are not triangle-shaped grids.

4 RNIC on Binary CSPs

Knowing the structure of the dual graph of a binary CSP, we first prove limitations of the filtering power of RNIC and wRNIC (RNIC enforced on a minimal dual graph). Then we theoretically compare sCDC, RNIC, and NIC, showing that they are pairwise incomparable.

4.1 Effects of the dual-graph's structure on RNIC

Theorem 2. *RNIC, R(∗,2)C, and R(∗,m)C are equivalent on any dual graph that is tree structured or is a cycle of length* $\geq maximum(4, m+1)$.

Proof: By straightforward generalization of Theorem 5 in [17]. □

Any redundancy-free dual graph of an arbitrary binary CSP can contain only one or more of the following configurations:

1. A cycle of length four, on a grid-shaped dual graph
2. A cycle of length larger than four as shown in Figure 6.
3. A triangle along the diagonal.

In the first two cases above, enforcing RNIC on the minimal dual graph (wRNIC) is equivalent to R(∗,2)C by Theorem 2. On the third case, wRNIC is equivalent to R(∗,3)C.

Theorem 3. *On a binary CSP, wRNIC is never strictly stronger than R(∗,3)C.*

Proof: See Appendix C. □

Using an algorithm for enforcing RNIC to enforce R(∗,3)C is wasteful of resources. Indeed, the former executes more consistency-checking operations than needed to enforce R(∗,3)C given that the neighborhoods considered by the former are supersets of those considered by the latter.

4.2 Comparing sCDC, RNIC and NIC

Theorem 4. *On binary CSPs, sCDC and RNIC+DF are incomparable.*

Proof: In Figure 9, the CSP is RNIC+DF but not sCDC. sCDC empties all

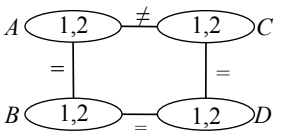

Fig. 9. RNIC+DF but not sCDC.

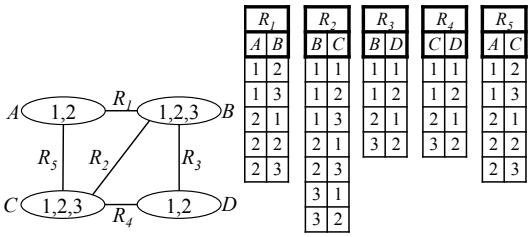

Fig. 10. sCDC but not RNIC+DF.

variables domains. In Figure 10, borrowed from Debruyne and Bessière [5], the CSP is sCDC but not RNIC+DF. RNIC removes $\{(2,3),(3,2)\}$ from R_2, $\{(1,2),(1,3)\}$ from R_1, and $\{(1,2),(1,3)\}$ from R_5. Therefore, RNIC+DF removes the value 1 from A. □

Theorem 5. *On binary CSPs, sCDC and NIC are incomparable.*

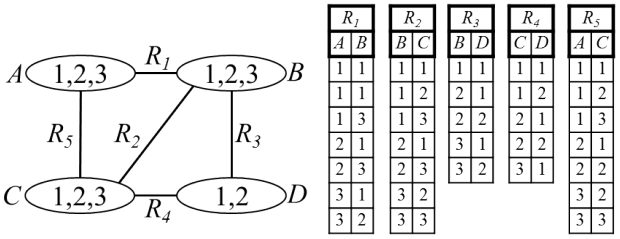

Fig. 11. sCDC but not NIC.

Proof: In Figure 9, the CSP is NIC but not sCDC. In Figure 11, borrowed from Debruyne and Bessière [5], the CSP is sCDC but not NIC. NIC removes the value 1 from A. □

Theorem 6. *On binary CSPs, NIC and RNIC+DF are incomparable.*

Proof: In Figure 12, the CSP is NIC but not RNIC+DF. RNIC removes the

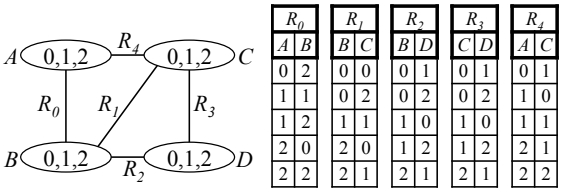

Fig. 12. NIC but not RNIC+DF.

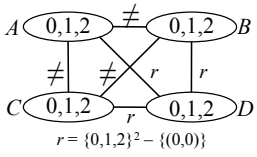

Fig. 13. RNIC+DF but not NIC.

tuples in $\{(0,2),(2,2)\}$ from R_0, $\{(0,0),(1,2)\}$ from R_1, $\{(0,2)\}$ from R_2, $\{(0,2)\}$ from R_3, and $\{(0,1),(2,1)\}$ from R_4. Therefore, RNIC+DF removes the value 0 from A. In Figure 13, the CSP is RNIC+DF but not NIC. NIC removes the value 0 from D. □

5 Related Work

Most of the related work is discussed in Section 2.2. To the best of our knowledge, no prior work has investigated the structure of the dual graph of binary CSPs.

Bacchus et al. studied the application of NIC to the dual encoding of a CSP, which they denoted *nic(dual)* [1]. While this property is identical to RNIC, the paper did not go beyond stating that *nic(dual)* is strictly stronger than *ac(dual)*.

Despite its pruning power and light space overhead, NIC received relatively little attention in the literature, likely because of the prohibitive cost of the algorithm for enforcing it. Freuder and Elfe tested their algorithm in a preprocessing step to backtrack search for solving binary instances whose constraint density did not exceed 4.25% [8]. Debruyne and Bessière showed that NIC is ineffective on sparse graphs and too costly on dense graphs [5].

6 Experimental Results

We study the impact of enforcing consistency on binary CSPs in a pre-processing step, prior to search. We then find the first solutions of the CSPs with backtrack search using a dynamic domain/weighted-degree variable (dom/wdeg) ordering and MAC [16] for full lookahead. We consider the four consistency properties: AC, sCDC, NIC and selRNIC. We measure the number of visited during search and the total CPU time (i.e., pre-processing and search). In practice, out of the five RNIC consistency algorithms, *the performance of only selRNIC matters* because it enforces the algorithm chosen by a selection policy [17].

We ran our experiments on benchmarks from the CSP Solver Competition.[6] We limited the processing time per instance to 90 minutes. We tested 571 instances of binary CSPs, with density in [1%,100%] with an average of 18%. We report the following measures:

- *#Instances*: We report two numbers for each benchmark. The first number is the number of instances completed by all four algorithms. The second number is the total number of instances in the benchmark.
- *CPU Time (msec)*: The average CPU time in milliseconds for pre-processing and search, computed over only the instances completed by all algorithms. The number of additional instances solved above the number completed by all is given in parenthesis.
- *BT-Free*: The number of instances that were solved backtrack-free.
- *#NV*: the average number of nodes visited for search, computed over only the instances that were completed by all algorithms.

In the results tables, we highlight in boldface the best values. When one of the algorithms does not complete any instances within the time threshold, no averages can be computed. To obtain averages over these instances, we compute the averages over only the algorithms that 'completed,' and mark in gray the box corresponding to the ignored algorithm. Table 1 shows the CPU times for the tested benchmarks, and Table 2 shows the number of instances solved backtrack-free and the number of nodes visited. We split the results into three sections based on average CPU time: those where NIC performs well, those where sCDC1 performs well, and those where selRNIC performs well.

While NIC may be too costly to use in general [5], there are difficult problems that benefit from higher level consistency. On instances where NIC performs the quickest in terms of CPU time, its search has orders of magnitude lower nodes visited than the other algorithms. Interestingly, NIC performs well on the rand-2-23/24 benchmarks, where the density is 100% (there is a constraint between every pair of variables). This result is interesting, because NIC is solving the instance during pre-processing. Therefore, it is solving every instance backtrack-free. Note, that on rand-2-24, despite taking a large amount of CPU time, NIC solves 7 additional instances that no other tested algorithm solved. The average density for the other benchmarks where NIC performs well is 14%.

[6] All constraints are normalized `http://www.cril.univ-artois.fr/CPAI09/`.

Table 1. CPU time: Search (MAC+dom/wdeg) with pre-processing by AC3.1, sCDC1, NIC, and selRNIC.

Benchmark	#Instances	AC3.1	sCDC1	NIC	selRNIC
		CPU Time (msec)			
		NIC Quickest			
bqwh-15-106	100/100	3,505	3,860	**1,470**	3,608
bqwh-18-141	100/100	68,629	82,772	**38,877**	77,981
graphColoring-sgb-queen	12/50	680,140	(+3) -	**(+9) 57,545**	634,029
graphColoring-sgb-games	3/4	41,317	33,307	**(+1) 860**	41,747
rand-2-23	10/10	1,467,246	1,460,089	**987,312**	1,171,444
rand-2-24	3/10	**567,620**	677,253	**(+7)** 3,456,437	677,883
		sCDC1 Quickest			
driver	2/7	(+5) 70,990	**(+5) 17,070**	358,790	(+4) 185,220
ehi-85	87/100	(+13) 27,304	**(+13) 573**	513,459	(+13) 75,847
ehi-90	89/100	(+11) 34,687	**(+11) 605**	713,045	(+11) 90,891
frb35-17	10/10	41,249	**38,927**	179,763	73,119
		selRNIC Quickest			
composed-25-1-25	10/10	226	335	1,457	**114**
composed-25-1-2	10/10	233	283	1,450	**88**
composed-25-1-40	9/10	(+1) 288	(+1) 357	120,544	**(+1) 137**
composed-25-1-80	10/10	223	417	(+1) -	**190**
composed-75-1-25	10/10	2,701	1,444	363,785	**305**
composed-75-1-2	10/10	2,349	1,733	48,249	**292**
composed-75-1-40	7/10	(+3) 1,924	(+3) 1,647	631,040	**(+3) 286**
composed-75-1-80	10/10	1,484	1,473	(+1) -	**397**

On instances where sCDC1 performs the quickest in terms of CPU time, it is able to filter the instances very quickly. The frb35-17 benchmark has an average density of 44%. This large density explains why NIC is performing poorly on these instances. An interesting benchmark is ehi-85/90 whose instances are all unsatisfiable. Interestingly, sCDC1 detects unsatisfiability at pre-processing by a domain wipe-out of the very first variable that it checks. Thus, its speed. selRNIC and (to a lesser extent) NIC, detect unsatisfiability at pre-processing, but cost more effort than sCDC1. Note that AC3.1 is too weak to uncover inconsistency at pre-precessing.

Interestingly, selRNIC automatically selected RNIC, and not wRNIC, on all tested benchmarks, except rand-2-23/24 where it selected wRNIC, thus not hindering itself as predicted by Theorem 3. selRNIC performs well on the composed benchmarks, where all of the algorithms, except AC3.1, were able to detect unsatisfiability at pre-processing. For the composed-25 benchmarks, the average density of the CSP is 50% (the average dual graph density is 12%), and for the

Table 2. Number of nodes visited (#NV): Search (MAC+dom/wdeg) with preprocessing by AC3.1, sCDC1, NIC, and selRNIC.

Benchmark	#Instances	AC3.1	sCDC1	NIC	selRNIC	AC3.1	sCDC1	NIC	selRNIC
		BT-Free				#NV			
		NIC Quickest							
bqwh-15-106	100/100	0	3	**8**	5	1,807	1,881	**739**	1,310
bqwh-18-141	100/100	0	0	**1**	0	25,283	25,998	**12,490**	22,518
graphColoring-sgb-queen	12/50	1	-	**16**	1	91,853	-	**15,798**	91,853
graphColoring-sgb-games	3/4	1	1	**4**	1	14,368	14,368	**40**	14,368
rand-2-23	10/10	0	0	**10**	0	471,111	471,111	**12**	471,111
rand-2-24	3/10	0	0	**10**	0	222,085	222,085	**24**	222,085
		sCDC1 Quickest							
driver	2/7	**1**	**2**	**1**	**1**	3,893	**409**	3,763	3,763
ehi-85	87/100	0	**100**	87	**100**	1,425	**0**	**0**	**0**
ehi-90	89/100	0	**100**	89	**100**	1,298	**0**	**0**	**0**
frb35-17	10/10	**0**	**0**	**0**	**0**	24,491	24,491	24,491	**24,346**
		selRNIC Quickest							
composed-25-1-25	10/10	0	**10**	**10**	**10**	153	**0**	**0**	**0**
composed-25-1-2	10/10	0	**10**	**10**	**10**	162	**0**	**0**	**0**
composed-25-1-40	9/10	0	**10**	9	**10**	172	**0**	**0**	**0**
composed-25-1-80	10/10	0	**10**	-	**10**	112	**0**	-	**0**
composed-75-1-25	10/10	0	**10**	**10**	**10**	345	**0**	**0**	**0**
composed-75-1-2	10/10	0	**10**	**10**	**10**	346	**0**	**0**	**0**
composed-75-1-40	7/10	0	**10**	7	**10**	335	**0**	**0**	**0**
composed-75-1-80	10/10	0	**10**	-	**10**	199	**0**	-	**0**

composed-75 benchmarks, the average density of the CSP is 20% (the average dual graph density is 5%). The large densities of the composed-75 benchmark explain the poor performance of NIC.

7 Conclusions & Future Work

An important contribution of this paper is the understanding of the structure of the dual graph on binary CSPs, which should impact the development of future consistency algorithms that operate on the dual graph of binary CSPs. We also theoretically showed that NIC, sCDC, and RNIC are incomparable. Despite previous work showing that NIC may be too costly to use in general [5], our experimental results show that there are instances that benefit from higher level consistency.

The algorithm we use to remove redundant edges from a dual graph generates triangle-shaped grids for binary CSPs [10]. However, there may also be non-grid

shaped minimal dual graphs. We propose to investigate why this algorithm favors the triangle-shaped grids. We also propose to develop a portfolio-based algorithm that measure the structure of a constraint graph and of its dual graph to select the appropriate consistency property to enforce.

Acknowledgments

Experiments were conducted on the equipment of the Holland Computing Center at the University of Nebraska-Lincoln. Robert Woodward was partially supported by a National Science Foundation (NSF) Graduate Research Fellowship grant number 1041000. This research is supported by NSF Grant No. RI-111795.

A Proof of Theorem 1

(By induction of number of variables.)

Base Step: Stated for $n = 3$.

For $n = 3$, the constraint graph is shown in Figure 14 and the corresponding dual graph in Figure 15. The dual graph is obviously a triangle.

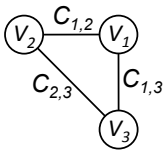

Fig. 14. A complete constraint graph with 3 variables.

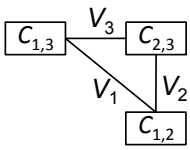

Fig. 15. The dual graph of a complete constraint graph with 3 variables.

- The two vertices corresponding to the constraints over the variable V_1 form the diagonal.
- The two vertices corresponding to the constraints over V_2 start at $C_{1,2}$ and have 0 vertices along the horizontal axis, and one vertex along the vertical axis. Also, the two vertices corresponding to the constraints over V_3 start at $C_{1,3}$ have 0 vertices along the horizontal axis, and one vertex along the vertical axis.

Inductive Step: Assume that the theorem holds for a CSP with k variables (inductive hypothesis). Show the theorem holds for a CSP with $k + 1$ variables (inductive step).

Consider the complete constraint graph of a CSP with k variables, which is the clique K_k. By the inductive hypothesis, the dual graph can be arranged in the triangle-shaped grid. Now, add the variable V_{k+1} to the CSP. In order to connect V_{k+1} to all k variables, k constraints are added to the constraint graph of the CSP, as shown in Figure 16. Namely, these k constraints are $C_{i,k+1}, \forall i \leq k$. Place the dual variables as follows, going from right to left in Figure 17:

- $C_{i,k+1}, i \in [2, k-1]$ is placed above $C_{i,k}$,

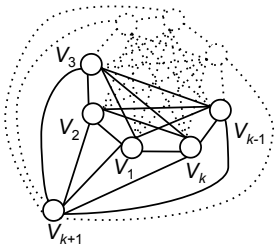

Fig. 16. A complete graph with $k+1$ variables.

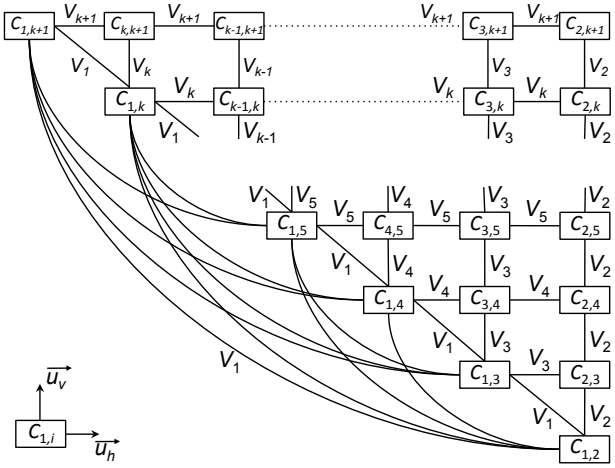

Fig. 17. The dual graph of Figure 16.

- $C_{k,k+1}$ is placed above $C_{1,k}$, and
- $C_{1,k+1}$ is placed to the left of $C_{k,k+1}$.

This arrangement yields a dual graph that is a triangle-shaped grid because:

- The vertices corresponding to the constraints over the variable V_1 are located on the diagonal of the triangle because $C_{k+1,1}$ is to the left of $C_{k+1,k}$,
- The coordinate system centered on $C_{1,i\in[2,k]}$ increases by one vertical unit for vertex $C_{k+1,i}$ and labeled with variable V_i.
- The coordinate system centered on $C_{1,k+1}$ has $(k+1)-2 = k-1$ vertices on the horizontal axis and 0 vertices in the vertical axis. The k vertices on the top row of the triangle form a clique whose edges are labeled with V_{k+1} (shown partially, for readability).

Consequently, this new dual graph of a complete constraint graph of $k+1$ variables has the topology of a triangle-shaped grid. □

B Proof of Corollary 1

Let us consider the $n-1$ vertices corresponding to the constraints that apply on variable V_i and the coordinate system defined by the horizontal and vertical unit vectors $\boldsymbol{u_h}$, $\boldsymbol{u_v}$ and centered on $C_{1,i}$. All edges between the $i-2$ horizontal vertices and the $n-i$ vertical vertices that link two non-consecutive vertices are redundant and can be removed, leaving a path linking the $n-1$ vertices along the horizontal and vertical axis. As for V_1, a similar operation can be applied to the vertices along the diagonal of the triangle. □

C Proof of Theorem 3

(By contradiction) Assume that wRNIC is strictly stronger than R(∗,3)C, that is, enforcing the former can result in more filtering more than enforcing the latter. To filter more, wRNIC has to consider simultaneously four constraints. Therefore, there must be a configuration of the minimal dual graph where a given constraint, C_1, has three adjacent constraints C_2, C_3, and C_4, and where C_1 is not an articulation point (otherwise, wRNIC would have the same filtering power as R(∗,3)C). The only redundancy-free configuration is the one shown in Figure 18. We show that this configuration is not possible.

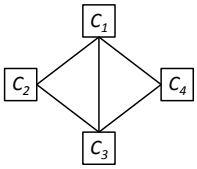

Fig. 18. A redundancy-free configuration of four binary constraints.

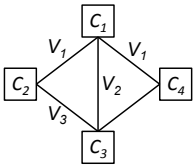

Fig. 19. One possible labeling of the edges incident to C_1.

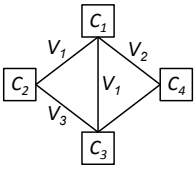

Fig. 20. The other possible labeling of the edges incident to C_1.

1. Given the topology of the graph shown in Figure 18, the three edges incident to C_1 cannot have the same labeling, for example variable V_1, because C_1 becomes a unary constraint. There cannot be three different labeling, for example variables V_1, V_2, and V_3, otherwise C_1 becomes a ternary constraint. Thus, they must be labeled with two variables, V_1 and V_2, as shown in Figures 19 and 20.
2. In Figure 19, the edge between C_2 and C_3 cannot be labeled V_1 (otherwise, C_2 becomes a unary constraint); cannot be labeled V_2 (otherwise, the scopes of C_2 and C_1 become equal, and we assume that the CSP is normalized); therefore, it must be labeled V_3. The edge between C_3 and C_4 cannot be labeled V_1 or V_4 (otherwise, C_3 becomes a ternary constraint); cannot be labeled V_2 (otherwise, the scopes of C_1 and C_3 become equal); cannot be labeled V_3 (otherwise, the scopes of C_2 and C_4 become equal). Therefore, no possible labeling for the edge between C_3 and C_4 exists, and this configuration is impossible.
3. In Figure 20, the edge between C_2 and C_3 cannot be labeled V_1 (otherwise, C_2 would be a unary constraint); cannot be labeled V_2 (otherwise, the scopes of C_1 and C_2 become equal); cannot be labeled V_3 (otherwise, the scopes of C_2 and C_3 become equal). Therefore, no possible labeling for the edge between C_2 and C_3 exist, and this configuration is impossible.

Consequently, no redundancy-free dual graph of a binary CSP can have a configuration of its vertices for enforcing R(∗,4)C. □

References

1. Bacchus, F., Chen, X., Beek, P.V., Walsh, T.: Binary vs. Non-Binary Constraints. Artificial Intelligence 140, 1–37 (2002)
2. Bessiere, C.: Handbook of Constraint Programming, chap. Constraint Propagation. Elsevier (2006)
3. Debruyne, R., Bessière, C.: From Restricted Path Consistency to Max-Restricted Path Consistency. In: Principles and Practice of Constraint Programming (CP 97). Lecture Notes in Computer Science, vol. 1330, pp. 312–326. Springer (1997)
4. Debruyne, R., Bessière, C.: Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. In: Proceedings of the 15 th International Joint Conference on Artificial Intelligence. pp. 412–417 (1997)
5. Debruyne, R., Bessière, C.: Domain Filtering Consistencies. Journal of Artificial Intelligence Research 14, 205–230 (2001)
6. Dechter, R.: Constraint Processing. Morgan Kaufmann (2003)
7. Freuder, E.C.: A Sufficient Condition for Backtrack-Free Search. JACM 29 (1), 24–32 (1982)
8. Freuder, E.C., Elfe, C.D.: Neighborhood Inverse Consistency Preprocessing. In: Proceedings of AAAI-96. pp. 202–208. Portland, Oregon (1996)
9. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Elsevier (2004), annals of Discrete Mathematics, Vol 75
10. Janssen, P., Jégou, P., Nougier, B., Vilarem, M.C.: A Filtering Process for General Constraint-Satisfaction Problems: Achieving Pairwise-Consistency Using an Associated Binary Representation. In: IEEE Workshop on Tools for AI. pp. 420–427 (1989)
11. Karakashian, S., Woodward, R., Reeson, C., Choueiry, B.Y., Bessiere, C.: A First Practical Algorithm for High Levels of Relational Consistency. In: 24th AAAI Conference on Artificial Intelligence (AAAI 10). pp. 101–107 (2010)
12. Kjærulff, U.: Triagulation of Graphs - Algorithms Giving Small Total State Space. Research Report R-90-09, Aalborg University, Denmark (1990)
13. Lecoutre, C., Cardon, S., Vion, J.: Conservative Dual Consistency. In: Proceedings of AAAI-2007. pp. 237–242 (2007)
14. Lecoutre, C., Cardon, S., Vion, J.: Second-Order Consistencies. Journal of Artificial Intelligence Research 40, 175–219 (2011)
15. Mackworth, A.K.: Consistency in Networks of Relations. Artificial Intelligence 8, 99–118 (1977)
16. Sabin, D., Freuder, E.C.: Contradicting Conventional Wisdom in Constraint Satisfaction. In: Proceedings of the 11 th European Conference on Artificial Intelligence. pp. 125–129. Amsterdam, The Netherlands (1994)
17. Woodward, R., Karakashian, S., Choueiry, B.Y., Bessiere, C.: Solving Difficult CSPs with Relational Neighborhood Inverse Consistency. In: 25th AAAI Conference on Artificial Intelligence (AAAI 11). pp. 112–119 (2011)