Set-Valued Var. Anal (2018) 26:277–304
https://doi.org/10.1007/s11228-017-0461-4

Solving Graph Coloring Problems with the Douglas-Rachford Algorithm

Francisco J. Aragón Artacho[1] · Rubén Campoy[1]

Received: 15 December 2016 / Accepted: 1 November 2017 / Published online: 5 January 2018

Abstract We present the Douglas-Rachford algorithm as a successful heuristic for solving graph coloring problems. Given a set of colors, these types of problems consist in assigning a color to each node of a graph, in such a way that every pair of adjacent nodes are assigned with different colors. We formulate the graph coloring problem as an appropriate feasibility problem that can be effectively solved by the Douglas-Rachford algorithm, despite the non-convexity arising from the combinatorial nature of the problem. Different modifications of the graph coloring problem and applications are also presented. The good performance of the method is shown in various computational experiments.

Keywords Douglas-Rachford algorithm · Graph coloring · Feasibility problem · Non-convex

Mathematics Subject Classification (2010) 47J25 · 90C27 · 47N10

1 Introduction

A *graph* $G = (V, E)$ is a collection of points V that are connected by links $E \subset V \times V$. The points are usually known as *nodes* or *vertices* while the links are called *edges*, *arcs* or *lines*. An *undirected graph* is a graph in which the edges have no orientation; that is, the edges are not ordered pairs of vertices but sets of two vertices.

✉ Rubén Campoy
ruben.campoy@ua.es

Francisco J. Aragón Artacho
francisco.aragon@ua.es

1 University of Alicante, Alicante, Spain

A *proper m-coloring* of an undirected graph G is an assignment of one of m possible colors to each vertex of G such that no two adjacent vertices share the same color. More specifically, given the set of colors $K = \{1, \ldots, m\}$, an *m-coloring* of G is a mapping $c : V \mapsto K$, assigning a color to each vertex. We say that c is *proper* if

$$c(i) \neq c(j) \text{ for all } \{i, j\} \in E.$$

The *graph coloring problem* consists in determining whether it is possible to find a proper m-coloring of the graph G. For a basic reference on graph coloring, see e.g. [25].

Graph coloring has been used in many practical applications such as timetabling and scheduling [27], computer register allocation [15], radio frequency assignment [22], and printed circuit board testing [20]. The graph coloring problem was proved to be NP-complete [26], so it is reasonable to believe that no polynomial-time exact algorithm solving these problems can be found. For this reason, a wide variety of heuristics and approximation algorithms have been developed for solving graph coloring problems. See [28] for a bibliographic survey of algorithms and applications, or the more recent survey [19].

In this paper we show that the Douglas-Rachford algorithm can be successfully used as heuristic for solving a wide variety of graph coloring problems when they are conveniently modeled as feasibility problems. Despite that the convergence of the Douglas-Rachford algorithm is only guaranteed for convex sets, the method has been successfully employed for solving many different nonconvex optimization problems, specially those of combinatorial nature (see, e.g., [3, 4, 9, 17]). The Douglas-Rachford method belongs to the family of so-called projection algorithms, which are traditionally analyzed using nonexpansivity properties when the problem is convex. There are very few results explaining why the algorithm works in nonconvex settings, and even less justifying its good global performance. For example, the global convergence of the algorithm for the case of a sphere and a line was proved in [13] (see also [1, 14]), and global convergence was recently proved in [5] for the case of a halfspace and a potentially nonconvex set. For local convergence results involving nonconvex sets, see e.g. [11, 23, 29].

The good performance of the Douglas-Rachford algorithm for solving the problem consisting in coloring the edges of a complete graph with three colors while avoiding monochromatic triangles was shown by Elser et al. in [17]. As far as we know, this is the only instance of a graph coloring problem whose solution with the Douglas-Rachford algorithm has been studied in the literature. Further, the graph coloring problem considered in [17] is a very specific problem dealing with the coloring of edges of a complete graph. In this paper, we consider any possible graph, and we study node coloring problems instead of link coloring problems.

The paper is structured as follows. Section 2 contains some preliminary results and notions. We show how to model the graph coloring problem as a feasibility problem in Section 3. When available, maximal clique information can be easily added to the model, as explained in Section 3.1. We present two ways of reformulating a 3-SAT problem as a graph coloring problem in Section 3.2. The precoloring and list coloring problems, which are variations of the graph coloring problem, are discussed in Section 4. We also treat in this section a well-known example of the precoloring problem: Sudoku puzzles. In Section 5, we show that the 8-queens puzzle, as well as some generalizations, can be also modeled as modified graph coloring problems. In Section 6, we discuss the Hamiltonian path problem. We report the results of a collection of numerical experiments in Section 7, where we exhibit the good performance of the Douglas-Rachford method for finding a solution of all the graph coloring problems considered along the paper. We finish with various concluding remarks in Section 8.

2 Preliminaries

Let $\mathcal{H}$ be a Hilbert space with inner product $\langle\cdot,\cdot\rangle$ and induced norm $\|\cdot\|$. Given a nonempty subset $C \subseteq \mathcal{H}$ and $x \in \mathcal{H}$, a point $p \in C$ is said to be a *best approximation* to x from C if

$$\|p - x\| = d(x, C) := \inf_{c\in C} \|c - x\|.$$

If a best approximation in C exists for every point in $\mathcal{H}$, then C is called *proximal*. The *projector* onto C is the set-valued mapping $P_C : \mathcal{H} \rightrightarrows C$ given by

$$P_C(x) := \left\{p \in C : \|p - x\| = \inf_{c\in C} \|c - x\|\right\},$$

and the *reflector* is defined as $R_C := 2P_C - I$, where I denotes the identity operator. If every point $x \in \mathcal{H}$ has exactly one best approximation p, then C is called *Chebyshev* and p is referred as the *projection* of x onto C. In this case, both P_C and R_C are single-valued. Recall that a weakly closed nonempty subset of a Hilbert space is convex if and only if it is a Chebyshev set (see, e.g. [2, Theorem 3.2]).

Given $C_1, C_2, \ldots, C_r \subseteq \mathcal{H}$, the *feasibility problem* consists in finding a point belonging to all these sets, that is,

$$\text{Find } x \in \bigcap_{i=1}^{r} C_i.$$

In many practical situations, the projector onto each of these sets can be easily computed, while finding a point in the intersection of the sets might be intricate. In such cases, and when the sets are convex, the Douglas-Rachford method (DR in short) is a useful tool to solve the problem.

Fact 2.1 *Let $A, B \subseteq \mathcal{H}$ be closed and convex sets. Consider the Douglas-Rachford operator defined as*

$$T_{A,B} = \frac{I + R_B R_A}{2}. \tag{1}$$

Given any $x_0 \in \mathcal{H}$, for every $n \geq 0$, define $x_{n+1} = T_{A,B}(x_n)$. Then, the following holds.

(i) *If $A \cap B \neq \emptyset$, then $\{x_n\}$ converges weakly to a point $x^\star$ and $\{P_A(x_n)\}$ is weakly convergent to $P_A(x^\star) \in A \cap B$.*
(ii) *If $A \cap B = \emptyset$, then $\|x_n\| \to +\infty$. Further, if $v := P_{\overline{A-B}}(0) \in A - B$ where $\overline{A - B}$ denotes the closure of the set $A - B$, then $\{P_A(x_n)\}$ converges weakly to some point in $A \cap (v + B)$.*

Proof (i) See [8, Theorem 3.13 and Corollary 3.9] and [33, Theorem 1]. (ii) See [6, Corollary 2.2] and [10, Theorem 4.5]. □

Finitely many sets in a feasibility problem are usually handled by reducing the problem to the two-sets case trough the Pierra's *product space formulation*. To this aim, consider the Hilbert product space $\mathcal{H}^r$ and define the sets

$$C := \prod_{i=1}^{r} C_i \quad \text{and} \quad D := \left\{(x, x, \ldots, x) \in \mathcal{H}^r : x \in \mathcal{H}\right\}. \tag{2}$$

While the set D, sometimes called the *diagonal*, is always a closed subspace, the properties of C are largely inherited. For instance, C is nonempty if $C_1, \ldots, C_r$ are nonempty;

and if $C_1, \ldots, C_r$ are closed and convex, so is C. Thus, the feasibility problem can be reformulated as a two-sets problem, since

$$x \in \bigcap_{i=1}^{r} C_i \Leftrightarrow (x, x, \ldots, x) \in C \cap D.$$

Moreover, knowing the projectors onto $C_1, \ldots, C_r$, the projectors onto C and D can be easily computed. Indeed, for any $\mathbf{x} = (x_1, \ldots, x_r) \in \mathcal{H}^r$, we have

$$P_C(\mathbf{x}) = \prod_{i=1}^{r} P_{C_i}(x_i) \quad \text{and} \quad P_D(\mathbf{x}) = \left(\frac{1}{r}\sum_{i=1}^{r} x_i\right)^r,$$

see [30, Lemma 1.1]. For further details see, for example, [4, Section 3].

Throughout this paper the space $\mathcal{H}$ will be the Euclidean space $\mathbb{R}^{n\times m}$ of $n \times m$ real matrices. Its inner product is given by

$$\langle A, B\rangle := \operatorname{tr}\left(A^T B\right),$$

where A^T is the transpose matrix of A, and $\operatorname{tr}(M)$ is the trace of a square matrix M. The induced norm corresponds to the *Frobenius* norm

$$\|A\|_F = \sqrt{\operatorname{tr}(A^T A)} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2}.$$

Let us state three results that characterize the projectors onto certain subsets of $\mathbb{R}^{n\times m}$, which will be useful later for computing the projector onto different sets.

Fact 2.2 *Let $e_1, \ldots, e_n$ denote the unit vectors of the standard basis of $\mathbb{R}^n$, and consider $C = \{e_1, \ldots, e_n\}$. Then, for any $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$,*

$$P_C(x) = \{e_i : x_i = \max\{x_1, \ldots, x_n\}\}.$$

Proof See, e.g., [4, Remark 5.1]. □

Fact 2.3 *Let $A \in \mathbb{R}^{l\times n}$ be a full row rank matrix. Consider $C = \left\{Z \in \mathbb{R}^{n\times m} : AZ = 0\right\}$. Then, for any $X \in \mathbb{R}^{n\times m}$, one has*

$$P_C(X) = \left(Id_n - A^T\left(AA^T\right)^{-1} A\right) X,$$

where $Id_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix.

Proof See, e.g., [7, Proposition 3.28(iii)] combined with [7, Example 3.27]. □

Fact 2.4 *Let $C = \left\{z \in \{0,1\}^n : \sum_{i=1}^n z_i \geq 1\right\}$. Then, for any $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the projector onto C is described by*

$$P_C(x) = \begin{cases} P_{\{0,1\}^n}(x) \setminus \{0_n\} & \text{if } P_{\{0,1\}^n}(x) \neq \{0_n\}, \\ \{e_i : x_i = \max\{x_1, \ldots, x_n\}\} & \text{otherwise;} \end{cases}$$

where $e_1, \ldots, e_n$ denote the unit vectors of the standard basis of $\mathbb{R}^n$.

Proof First note that $C = \{0, 1\}^n \setminus \{0_n\}$. If $P_{\{0,1\}^n}(x) \neq \{0_n\}$, we trivially have that $P_C(x) = P_{\{0,1\}^n}(x) \setminus \{0_n\}$. Otherwise, we have that $x_i < 0.5$ for all $i = 1, \ldots, n$. Then, any projection in $P_C(x)$ will contain exactly one nonzero value, and the result follows from Fact 2.2. $\square$

To finish this section, let us shortly summarize some basic concepts of graph theory. A *complete graph* is an undirected graph in which every pair of nodes is connected by an edge. A *clique* is a subset of vertices of an undirected graph such that its induced graph is complete. A *maximal clique* is a clique that cannot be extended by adding one more vertex. A *path* is a sequence of edges that connects a sequence of distinct vertices. A path is said to be a *cycle* if there is an edge from the last vertex in the path to the first one.

3 Modeling Graph Coloring Problems as Feasibility Problems

The m-coloring of a graph $G = (V, E)$ with n nodes can be easily modeled as a feasibility problem. To this aim, let $X = (x_{ik}) \in \{0, 1\}^{n \times m}$, where $x_{ik} = 1$ indicates that vertex i receives color k. Then, we have the following constraints:

$$\sum_{k=1}^{m} x_{ik} = 1, \quad \text{for all } i = 1, \ldots, n; \tag{3}$$

$$x_{ik} + x_{jk} \leq 1, \quad \text{for all } \{i, j\} \in E, k = 1, \ldots, m; \tag{4}$$

$$x_{ik} \in \{0, 1\}, \quad \text{for all } i = 1, \ldots, n, k = 1, \ldots, m. \tag{5}$$

Constraint (3) together with (5) determine that each node has exactly one color. Constraint (4) combined with (5) impose the requirement that any two adjacent nodes cannot be assigned with the same color.

The formulation of the constraints has a big effect on the behavior of the Douglas-Rachford scheme when applied to nonconvex constraints. On the one hand, ones needs a formulation where the projectors onto the sets are easy to compute. On the other hand, the formulation chosen often determines whether or not the Douglas-Rachford scheme can successfully solve the problem at hand always, frequently or never [4]. For these two reasons, we have realized that it is convenient to reformulate constraint (4) as follows

$$x_{ik} + x_{jk} - y_{ek} = 0, \quad \text{for all } e = \{i, j\} \in E, k = 1, \ldots, m, \tag{6}$$

where $y_{ek} \in \{0, 1\}$ for all $i, j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m\}$. Although we have considerably increased the number of variables of the feasibility problem by adding lm new variables, where l is the number of edges in the graph, we have empirically observed that the Douglas-Rachford scheme becomes much more successful with this formulation.

Finally, note that, since the labeling of the colors does not have any significant meaning, every permutation of a proper coloring is also a proper coloring. In our numerical tests we observed that this abundance of equivalent solutions significantly decreases the rate of success of the Douglas-Rachford algorithm. To avoid this problem, we may restrict the set of possible colorings to those that assign the first color to the first vertex. Without loss of generality, we can assume that the first vertex is connected to at least another node, which thus cannot be colored with the first color. Hence, we can reduce the number of solutions by forcing one of these nodes to be assigned with the second color. For this reason, we add to the formulation the constraint

$$x_{1,1} = 1 \text{ and } x_{i_0,2} = 1, \quad \text{for some fixed } i_0 \in \{2, \ldots, n\} \text{ such that } \{1, i_0\} \in E. \tag{7}$$

In our experiments, vertex i_0 was chosen as $i_0 := \min\{i \in V : \{1, i\} \in E\}$. Of course, it would be possible to further reduce the number of solutions by following the same strategy with the largest complete subgraph contained in the graph G. As finding such a subgraph is not a trivial task and we did not observe in our numerical tests a clear improvement in the rate of success of the algorithm, we decided to only add constraint (7) into our formulation.

We shall also add the additional constraint that all m colors have to be used, i.e.,

$$\sum_{i=1}^{n} x_{ik} \geq 1, \quad \text{for all } k = 1, \ldots, m. \tag{8}$$

Let $E = \{e_1, \ldots, e_l\}$ be the set of edges, where $e_p \in \{1, \ldots, n\}^2$ for every $p = 1, \ldots, l$. Let $I := \{1, \ldots, n\}$ and $P := \{n+1, \ldots, n+l\}$, and let $K := \{1, \ldots, m\}$ be the set of colors. Then, the m-coloring problem determined by constraints (3), (5), (6), (7) and (8) can be formulated as a feasibility problem with four constraints:

$$\text{Find} \quad Z \in C_1 \cap C_2 \cap C_3 \cap C_4, \tag{9}$$

where $Z = (z_{ik}) \in \mathbb{R}^{(n+l)\times m}$ and

$$C_1 := \left\{ Z \in \mathbb{R}^{(n+l)\times m} : z_{ik} \in \{0, 1\}, \forall (i, k) \in I \times K \text{ and } \sum_{k=1}^{m} z_{ik} = 1, \forall i \in I \right\},$$

$$C_2 := \left\{ Z \in \mathbb{R}^{(n+l)\times m} : z_{ik} + z_{jk} - z_{pk} = 0, \text{ with } e_{p-n} = \{i, j\} \in E, \forall (p, k) \in P \times K \right\},$$

$$C_3 := \left\{ Z \in \{0, 1\}^{(n+l)\times m} : \sum_{i=1}^{n} z_{ik} \geq 1, \forall k \in K \right\},$$

$$C_4 := \left\{ Z \in \mathbb{R}^{(n+l)\times m} : z_{1,1} = 1, z_{i_0,2} = 1 \right\}.$$

Observe that constraint C_2 can be expressed in matrix form as

$$C_2 = \left\{ Z \in \mathbb{R}^{(n+l)\times m} : AZ = 0_{l\times m} \right\}, \tag{10}$$

where $A = (a_{pq}) \in \mathbb{R}^{l\times(n+l)}$ is defined by

$$a_{pq} := \begin{cases} 1 & \text{if } e_p = \{i, j\} \text{ and } q \in \{i, j\}, \\ -1 & \text{if } q = n + p, \\ 0 & \text{elsewhere;} \end{cases}$$

for each $p = 1, \ldots, l$ and $q \in I \cup P$.

The projectors onto C_1 and C_3 can be derived from Fact 2.2 and Fact 2.4, respectively; while the projector onto C_4 is trivially obtained. For any $Z \in \mathbb{R}^{(n+l)\times m}$, these projectors are given, pointwise, by

$$P_{C_1}(Z)[i, :] = \begin{cases} \left\{ e_k^T : z_{ik} = \max\{z_{i1}, z_{i2}, \ldots, z_{im}\} \right\} & \text{if } i \in I, \\ (z_{i1}, z_{i2}, \ldots, z_{im}) & \text{if } i \in P; \end{cases}$$

$$P_{C_3}(Z)[:, k] = \begin{cases} P_{\{0,1\}^{n+l}}(Z[:, k]) \setminus \{0_{n+l}\} & \text{if } P_{\{0,1\}^{n+l}}(Z[:, k]) \neq \{0_{n+l}\}, \\ \{e_i : z_{ik} = \max\{z_{1k}, \ldots, z_{nk}\}\} & \text{otherwise;} \end{cases}$$

$$\left(P_{C_4}(Z)\right)[i, k] = \begin{cases} 1 & \text{if } (i, k) \in \{(1, 1), (i_0, 2)\}, \\ z_{ik} & \text{otherwise;} \end{cases}$$

for each $i \in I \cup P$ and $k \in K$. Since A is full row rank, according to Fact 2.3, the projector onto C_2 is given by

$$P_{C_2}(Z) = \left(\mathrm{Id}_{n+l} - A^T \left(AA^T\right)^{-1} A\right) Z.$$

Finally, observe that the projectors onto C_1 and C_3 may be multivalued. A projection onto these sets $\pi_{C_1}(Z) \in P_{C_1}(Z)$ and $\pi_{C_3}(Z) \in P_{C_3}(Z)$ is given, pointwise, by

$$\left(\pi_{C_1}(Z)\right)[i,k] = \begin{cases} 1 & \text{if } i \in I,\, k = \operatorname{argmax}\{z_{i1}, z_{i2}, \ldots, z_{im}\}, \\ z_{ik} & \text{if } i \in P, \\ 0 & \text{otherwise;} \end{cases}$$

$$\left(\pi_{C_3}(Z)\right)[i,k] = \begin{cases} 1 & \text{if } i = \operatorname{argmax}\{z_{1k}, z_{2k}, \ldots, z_{nk}\}, \\ \min\{1, \max\{0, \operatorname{round}(z_{ik})\}\} & \text{otherwise;} \end{cases}$$

where the lowest index is chosen in argmax and $\operatorname{round}(0.5) = 0$.

3.1 Adding Maximal Clique Information

Let us illustrate with an example the need of adding maximal clique information into our formulation, whenever this information is available.

Example 3.1 Let us consider now the so-called *windmill graph* $\mathrm{Wd}(a, b)$, which is the graph constructed for $a \geq 2$ and $b \geq 2$ by joining b copies of a complete graph with a vertices at a shared vertex. A plot of $\mathrm{Wd}(6, 5)$ is shown in Fig. 1.

Every windmill graph $\mathrm{Wd}(a, b)$ can be easily a-colored (there are $a((a-1)!)^b$ different ways). Despite this abundance of valid colorings, the Douglas-Rachford scheme described in the previous section fails to find a solution rather often, see the results in Fig. 18. This

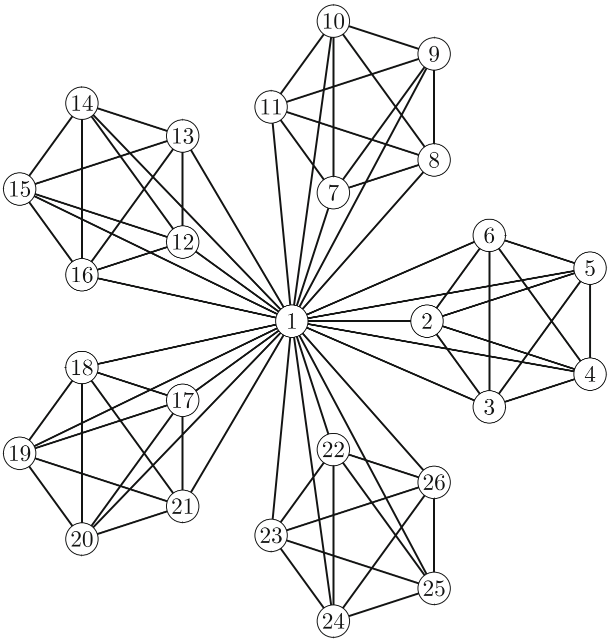


Fig. 1 Plot of the windmill graph Wd(6, 5)

graph has an additional available information that can be used: it has b maximal cliques of size a, and each color can be used at most once within each maximal clique.

Let $Q \subset 2^V$ be a nonempty subset of maximal cliques of the graph $G = (V, E)$ and let $\widehat{E} := E \cup Q$. Let $Q = \{e_{l+1}, \ldots, e_r\}$, with $r \geq l+1$. Thus, $\widehat{E} = \{e_1, \ldots, e_l, e_{l+1}, \ldots, e_r\}$. The maximal clique information can be easily added into constraint C_2 in (10). Indeed, let

$$\widehat{C}_2 := \left\{ Z \in \mathbb{R}^{(n+r)\times m} : \widehat{A}Z = 0_{r\times m} \right\},$$

where $\widehat{A} = (\widehat{a}_{pq}) \in \mathbb{R}^{r\times(n+r)}$ is defined by

$$\widehat{a}_{pq} := \begin{cases} 1 & \text{if } q \in e_p, \\ -1 & \text{if } q = n+p, \\ 0 & \text{elsewhere}; \end{cases}$$

for each $p = 1, \ldots, r$ and $q \in \{1, \ldots, n+r\}$. This is clearly an equivalent formulation of the m-coloring problem, where we have added $(r-l)m$ new variables (now $Z \in \mathbb{R}^{(n+r)\times m}$), which correspond to the (redundant) information that each color can only be used once within each maximal clique. Despite that, this formulation can be advantageous, as shown in Fig. 18. For some particular graphs, adding this information can be crucial, see Table 1, where we compare two reformulations of 3-SAT problems with and without maximal clique information. Explaining these reformulations is the subject of the next section.

3.2 Formulating 3-SAT as 3-Coloring

A *Boolean variable* takes logical values: True (T) or False (F). A *literal* is either a variable or its negation ($\neg$). A *clause* is a disjunction ($\vee$) of literals. A formula in *conjunctive normal form* is a conjunction ($\wedge$) of clauses. Given a formula in conjunctive normal form with 3 literals per clause, the *3-SAT* (*3-satisfiability*) problem consists in determining if there exists an assignment of variables that makes the formula true. Specifically, let $x_1, \ldots, x_n$ be n Boolean variables and consider m clauses $\theta_1, \ldots, \theta_m$, where each clause is the disjunction of 3 literals,

$$\theta_j = t_1^j \vee t_2^j \vee t_3^j, \quad \text{for all } j = 1, 2, \ldots, m;$$

with $t_1^j, t_2^j, t_3^j \in \bigcup_{i=1}^n \{x_i, \neg x_i\}$. Let ϕ be the formula comprising the conjunction of all the clauses:

$$\phi = \theta_1 \wedge \theta_2 \wedge \cdots \wedge \theta_m.$$

Then, the 3-SAT problem consists in determining if there exists an assignment of the variables that makes the formula ϕ true.

Example 3.2 Consider the following 3-SAT problem with 3 variables and 2 clauses:

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3).$$

There are several solutions to ϕ such as (F, T, F), (T, T, F) or (F, F, T), among others.

A 3-SAT problem can be reduced to a 3-coloring problem by using gadgets. A *gadget* is a small graph whose coloring solves some part of the problem. Using a set of gadgets and connecting them in an appropriate manner, the 3-coloring problem of the full graph can be made equivalent to solving the 3-SAT problem. We start by creating $n+1$ gadgets, one for each variable and an additional one for setting the interpretation of the colors:

(a) Create a gadget formed by a complete graph with 3 "color-meaning" nodes named T, F and G, see Fig. 2a. As this gadget is a complete graph, a different color must be

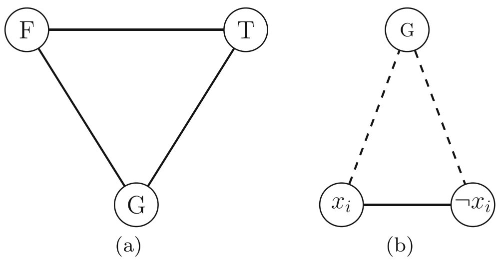


Fig. 2 Gadgets of the variables and colors

assigned to each node. The color assigned to node T will be interpreted as True, the color assigned to F as False, and the remaining color assigned to G (*ground* node) will not have any special interpretation.

(b) For each variable x_i, construct a gadget with 2 connected nodes, one associated to x_i and the other to $\neg x_i$. Link both of them to the node G to create a gadget of the form in Fig. 2b. This gadget forces a logical choice in the value of the variables. Thus, every variable will be assigned to either T or F, and the assignment of every variable and its complement will be consistent.

Next, we present two different formulations[1] of the gadgets corresponding to the clauses.

(c) For the *4-nodes formulation*, take each clause $\theta = t_1 \vee t_2 \vee t_3$ and create the gadget in Fig. 3a with the nodes associated to t_1, t_2, t_3, F, G, and 4 new nodes. The new unlabeled nodes do not have any special meaning, but, by the construction of the gadgets, every 3-coloring of a clause gadget will assign the same color as T to at least one of the literals t_1, t_2 or t_3. Thus, a valid 3-coloring of the gadget will make the corresponding clause to be True.

For the *5-nodes formulation*, the process is similar but introduces five new nodes instead of four: the gadget is shown in Fig. 3b.

(d) Finish building the graph by connecting the clause gadgets together using the edges from the common gadgets from Fig. 2. Full graphs for the four and five node formulations of the 3-SAT problem in Example 3.2 are shown in Fig. 4.

The graph resulting from putting all these gadgets together in the 4-nodes formulation has a total of $3 + 2n + 4m$ nodes and $3 + 3n + 9m$ edges. Observe that the graph has $n + 1$ maximal cliques with 3 nodes, one for each gadget of type (a) and (b). In the 5-nodes formulation, the resulting graph has a total of $3+2n+5m$ nodes and $3+3n+10m$ edges. The number of maximal cliques with 3 nodes has increased up to $n + 1 + 2m$, one for each gadget of type (a) and (b) and two for each gadget of type (c). A 3-coloring of the graph built under one of these two formulations corresponds to a solution of the associated 3-SAT problem. A solution to the 3-SAT problem in Example 3.2 using both formulations is shown in Fig. 4.

The results of testing the performance of the Douglas-Rachford method for solving a sample of 3-SAT problems using both reformulations as 3-coloring problems is shown in Section 7, see also Table 1. With a totally different direct formulation, the Douglas-Rachford method was first shown to be successful for solving 3-SAT problems in [17].

[1]The gadget corresponding to the 5-nodes formulation is well-known and first appeared in [21, Fig. 1]. We inquired of various experts in the field about the origin of the 4-nodes gadget, but none of them knew about it. We found it in the class notes prepared by Keith Schwarz [32, p. 27]. As he independently came up with it and we have not been able to find it elsewhere, we believe K. Schwarz is its originator.

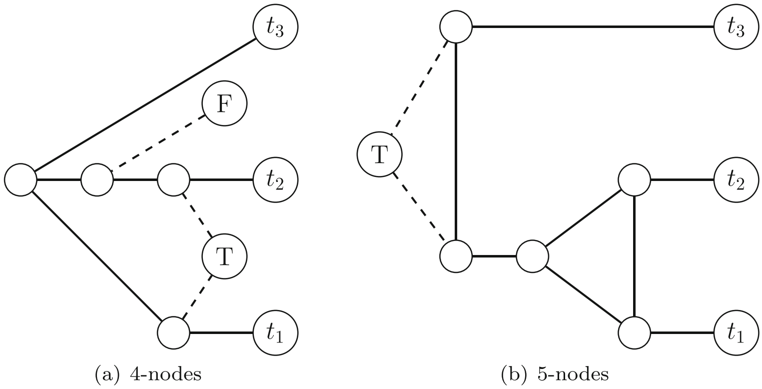
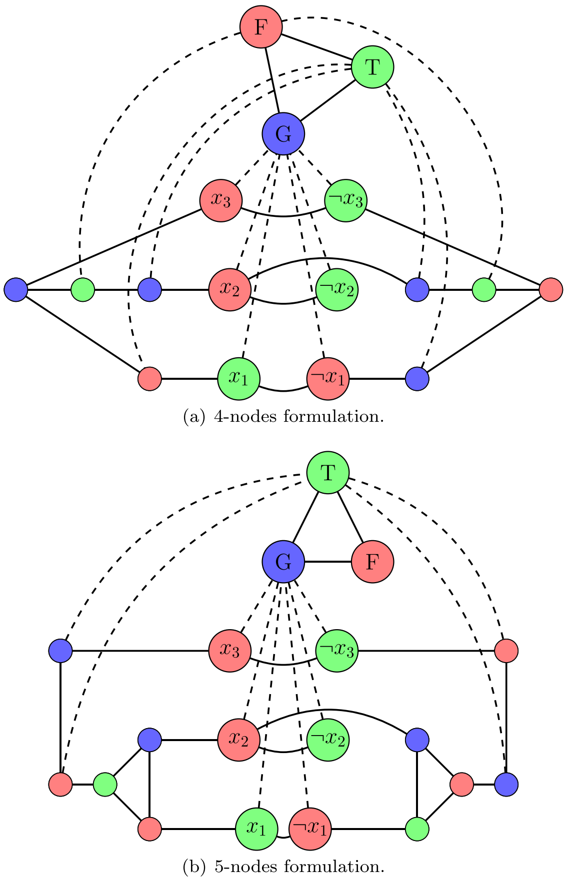


Fig. 3 Gadgets of the clauses

Fig. 4 Two different formulations of the 3-SAT problem in Example 3.2 as a 3-coloring problem. The same solution of the 3-SAT problem is shown for both formulations



4 Precoloring and List Coloring Problems

In many practical graph coloring problems, the set of eligible colors for each of the nodes can be different. This is the case in the *precoloring problem*, a slight modification of the graph coloring problem in which a subset of the vertices has been preassigned to some colors. The task is to color the remaining vertices to obtain a valid coloring of the entire graph. More generally, in the *list coloring problem*, each vertex can only be colored from a list of admissible colors.

The notion of list coloring was independently introduced by Vizing [34], and Erdös, Rubin and Taylor [18]. Given a graph $G = (V, E)$ and a set of m colors $K = \{1, \ldots, m\}$, let $L : V \rightrightarrows K$ be a mapping assigning to each vertex $v \in V$ a list of admissible colors $L(v) \subseteq K$. Thus, the list coloring problem consists in finding a proper coloring of the vertices of the graph G verifying that the color assigned to each vertex belongs to its list of admissible colors; that is, $c(i) \neq c(j)$ for all $\{i, j\} \in E$, and $c(i) \in L(i)$ for all $i \in V$. Note that an ordinary graph coloring problem is a special case of list coloring where $L(i) = K$ for every vertex $i \in V$, and so are the precoloring problems, where the precolored vertices have a list of admissible colors with size one.

List coloring problems can be reduced to standard graph coloring problems. To this aim, one shall add a complete subgraph with m new nodes, each one representing a color in K, and connect each vertex $i \in V$ with the new nodes that represent the colors not belonging to $L(i)$. If we denote by $|A|$ the cardinality of a finite set A, the new graph will have $n + m$ nodes, $l^\star = |E| + \frac{m(m-1)}{2} + nm - \sum_{i=1}^{n} |L(i)|$ edges, and an additional maximal clique of size m. In this way, any valid m-coloring of the extended graph will lead to a solution for the original list coloring problem. An example of such construction with a wheel graph of 5 nodes is shown in Fig. 5.

Note that the new feasibility problem is defined in $\mathbb{R}^{(n+m+l^\star)\times m}$. Constraint C_4 has to be changed, as it no longer makes sense. We have m new nodes, labeled $n+1, \ldots, n+m$, and each of them represents a color. To include this information, we shall replace C_4 by

$$C_4^\star := \left\{ Z \in \mathbb{R}^{(n+m+l^\star)\times m} : z_{n+k,k} = 1, \forall k \in K \right\}.$$

Thereby, the solution set is $C_1 \cap C_2 \cap C_3 \cap C_4^\star$. The projector onto $C_4^\star$ is given by

$$\left(P_{C_4^\star}(Z)\right)[i, k] = \begin{cases} 1 & \text{if } i = k + n, \\ z_{ik} & \text{otherwise.} \end{cases}$$

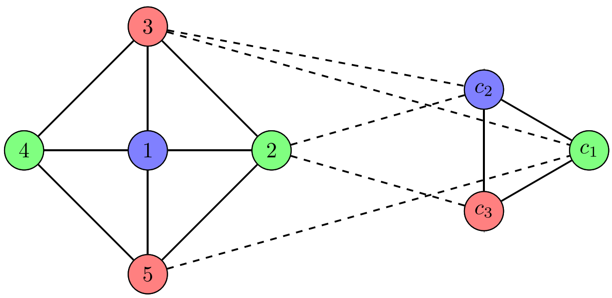


Fig. 5 List coloring reduced to graph coloring of a wheel graph of 5 nodes with admissible colors lists $L(1) = L(4) = \{1, 2, 3\}$, $L(2) = \{1\}$, $L(3) = \{3\}$, and $L(5) = \{2, 3\}$. Nodes c_1, c_2 and c_3 represent colors 1, 2 and 3, respectively

As the increase in the number of nodes and edges may cause the DR algorithm to become slower, another option here would be to directly modify the constraint C_1 to only allow admissible colors, that is, to replace it by the set

$$\overline{C}_1 := \left\{ Z \in \mathbb{R}^{(n+l)\times m} : z_{ik} \in \{0,1\}, \forall (i,k) \in I \times K \text{ and } \sum_{k \in L(i)} z_{ik} = 1, \sum_{k \notin L(i)} z_{ik} = 0, \forall i \in I \right\}.$$

A projection onto $\overline{C}_1$ is given, pointwise, by

$$\left(\pi_{\overline{C}_1}(Z)\right)[i,k] = \begin{cases} 1 & \text{if } i \in I, k = \operatorname{argmax}\{z_{ij}, j \in L(i)\}, \\ z_{ik} & \text{if } i \in P, \\ 0 & \text{otherwise.} \end{cases}$$

In this formulation, constraint C_4 cannot be adapted and it has to be removed from the feasibility problem. Then, the solution set becomes $\overline{C}_1 \cap C_2 \cap C_3$. We shall compare the performance of DR with both formulations in Section 7.

4.1 Formulating Sudokus as 9-Precoloring Problems

It is easy to formulate *Sudoku puzzles* as graph coloring problems. This kind of puzzles consist in a 9×9 grid, divided in nine 3×3 subgrids, with some entries already prefilled. The objective is to fill the remaining cells in such a way that each row, each column and each subgrid contains the digits from 1 to 9 exactly once.

We shall model Sudokus as 9-precoloring problems, with the aim of applying DR. The construction of the graph is very simple and intuitive. Each cell in the grid shall be represented by a node. Then, we link two nodes if their respective associated cells lay in the same row, same column or same subgrid (see Fig. 6). The graph obtained contains 81 nodes and 810 edges, see Fig. 7b. Furthermore, a rich maximal clique information is known. Namely, there are 27 maximal cliques of size 9, one per row, one per column and one per subgrid.

Sudoku puzzles can be directly modeled as integer feasibility programs. Despite that the Douglas-Rachford algorithm fails to solve these integer problems, it can be successfully used for solving the puzzles after reformulating them as binary programs, see [4, Section 6]. We must acknowledge here the fundamental contribution of Veit Elser [17], who first realized the usefulness of this binary reformulation for the success of the DR algorithm. Further, Elser seems to have been the first to see the remarkable potential of the algorithm for solving nonconvex problems.

We associate a color to each of the 9 digits of the puzzle. Since some cells of the Sudoku are prefilled, this is actually a graph precoloring problem. A valid coloring of the graph will lead to a solution of the Sudoku, as shown in the example in Fig. 7.

5 The 8-Queens Puzzle and Generalizations

The *8-queens puzzle* consists in placing eight chess queens on an 8×8 chessboard, so that none of them attack any other. Since a chess queen can be moved any number of squares vertically, horizontally or diagonally, the puzzle's constraints can be formulated as: there is at most one queen at each row, each column and each diagonal. The reformulation of an 8-queens puzzle as a graph coloring problem is similar to the one shown for Sudokus. Each square in the chessboard is represented by a node, and two nodes are linked if their corresponding squares lay on the same column, row or diagonal. The graph has 64 nodes, 728 links and 42 maximal cliques.

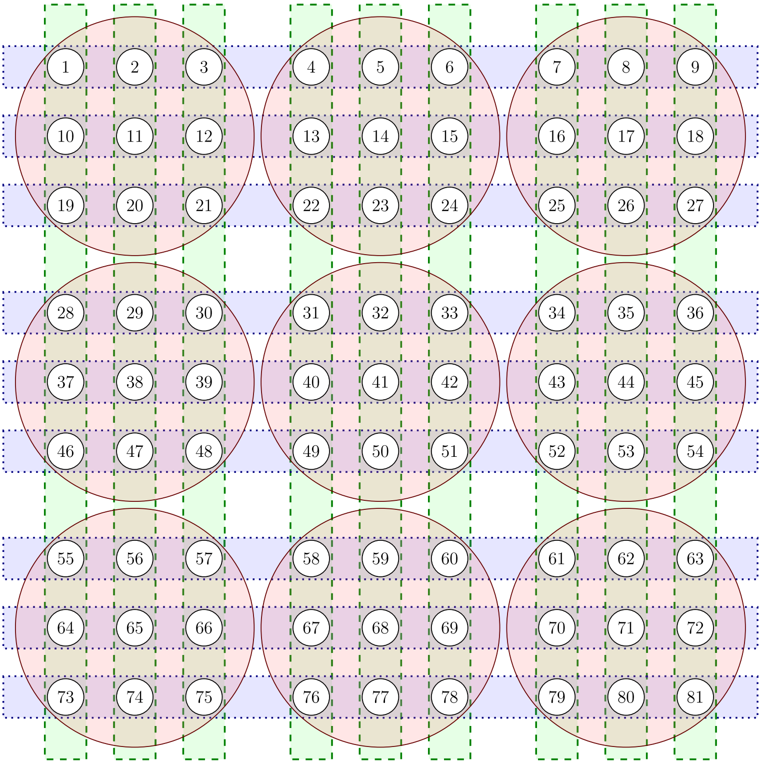
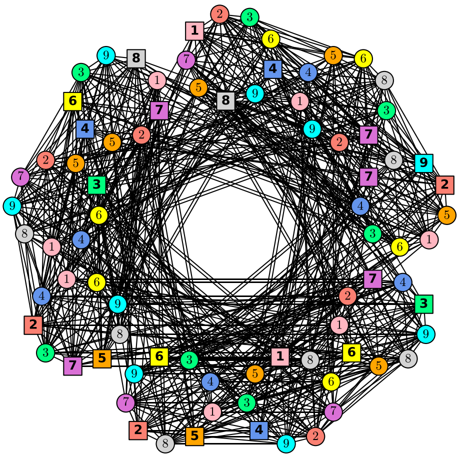


Fig. 6 Graph formulation of a Sudoku, with maximal cliques highlighted

1						7		9
	4				7	2		
8								
	7			1			6	
3								5
	6			4			2	
								8
		5	3				7	
7		2					4	6



(a) Unsolved Sudoku. (b) Graph coloring of Sudoku.

Fig. 7 **a** Unsolved Sudoku puzzle. **b** Graph representation of the Sudoku: each complete subgraph represents a subgrid and the squared nodes correspond to prefilled cells. The valid 9-coloring of the graph leads to a solution of the Sudoku

To solve the 8-queens puzzle, it is not necessary to color all the nodes, but only 8 of them with only one color. Thus, we are dealing with a partial graph coloring problem, in which we add the constraint that the color has to be used exactly 8 times. We must then remove the set C_4 in (9) and replace the sets C_1 and C_3 by

$$\breve{C}_1 := \left\{ Z \in \mathbb{R}^{(n+l)\times m} : z_{ik} \in \{0,1\}, \forall (i,k) \in I \times K \text{ and } \sum_{k=1}^{m} z_{ik} \leq 1, \forall i \in I \right\},$$

$$\breve{C}_3 := \left\{ Z \in \{0,1\}^{(n+l)\times m} : \sum_{i=1}^{n} z_{ik} = q, \forall k \in K \right\},$$

where $n = q = 8$ and $m = 1$ (puzzles with more colors can be considered). Hence, the solution set of the puzzle is $\breve{C}_1 \cap C_2 \cap \breve{C}_3$. A projection onto $\breve{C}_1$ and $\breve{C}_3$ is given by

$$\left(\pi_{\breve{C}_1}(Z)\right)[i,k] = \begin{cases} \min\{1, \max\{0, \operatorname{round}(z_{ik})\}\} & \text{if } i \in I, k = \operatorname{argmax}\{z_{i1}, z_{i2}, \ldots, z_{im}\}, \\ z_{ik} & \text{if } i \in P, \\ 0 & \text{otherwise}; \end{cases}$$

$$\left(\pi_{\breve{C}_3}(Z)\right)[i,k] = \begin{cases} 1 & \text{if } i \in Q_{k,q}, \\ \min\{1, \max\{0, \operatorname{round}(z_{ik})\}\} & \text{if } i \in P, \\ 0 & \text{otherwise}; \end{cases}$$

where, for a given color $k \in K$, we denote by $Q_{k,q} \subset I$ the set of indices corresponding to the q largest values in $\{z_{1k}, z_{2k}, \ldots, z_{nk}\}$ (lowest index is chosen in case of tie).

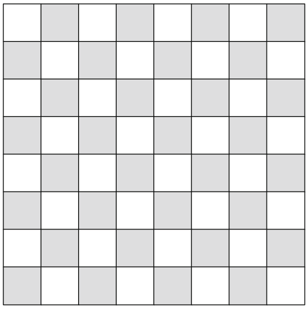
The 8-queens puzzle can be easily posed for any size of the chessboard. The problem has been generalized in many different directions, see [12] for a recent survey. One of these generalizations is the Queens_n^2 puzzle, where one must cover an entire chessboard $n \times n$ with n^2 queens, so that two queens of the same color do not attach each other. This problem is actually the n-coloring problem of the chessboard queens graph, so it can be directly modeled as explained in Section 3 using formulation (9). Different shapes can also be considered: we show in Fig. 8b a chessboard with a hole, and in Fig. 8c a puzzle dedicated to Jonathan Borwein. A solution to these puzzles, obtained with DR, is shown in Fig. 9.

The use of the Douglas-Rachford algorithm for solving the n-queens puzzle is proposed and studied in [31]. One of the main advantages of formulating these puzzles as graph coloring problems is that it is straightforward to model many variations of the problem. For instance, to model the knights puzzle, a similar puzzle played with knights instead of queens, one only needs to change the links of the chessboard graph, see Fig. 8a.

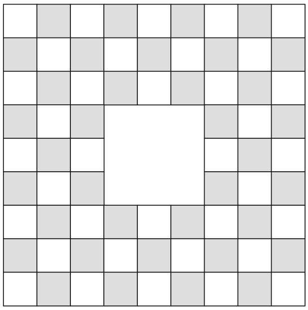
6 The Hamiltonian Path Problem

A *Hamiltonian path* is a path in a graph that visits every vertex exactly once. The Hamiltonian path problem consists in determining whether or not such a path exists. In this section we adapt the graph coloring scheme with the aim of using the Douglas-Rachford algorithm for finding Hamiltonian paths.

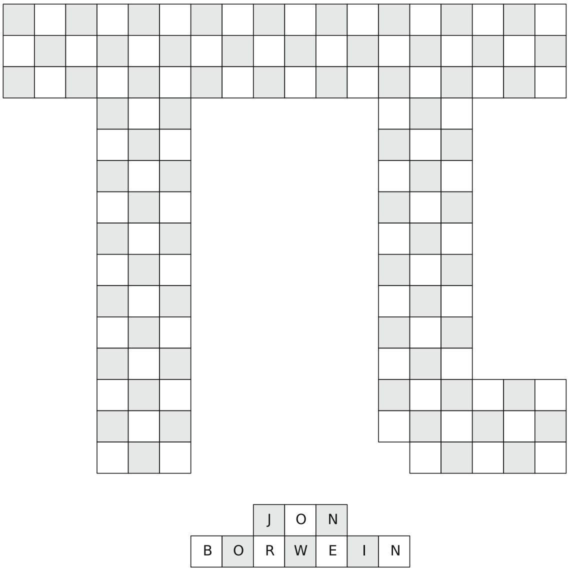
Given a graph G with n nodes, our objective will be to find an n-coloring of the graph, where each color $1, 2, \ldots, n$ will represent a position in the path. In order to ensure that the coloring represents a valid path, we will impose that two nodes assigned with two



(a) Classic chessboard.



(b) Chessboard with a hole.



(c) 'π-zzle'.

Fig. 8 **a** A 16-knights puzzle with 4 colors: a solution will fill the chessboard. **b** A 10-queens puzzle with 3 colors played in a 9×9 chessboard with a hole. **c** Empty 'π-zzle'. The goal of this puzzle is to place on the board 8 times each of the 18 letters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R and W. Ten cells have been prefilled. A solution to these puzzles computed with the Douglas-Rachford algorithm is shown in Fig. 9

consecutive colors must be linked. Constraint C_2 becomes now redundant, as every node must be assigned with a different color, and it is thus no longer necessary to work in $\mathbb{R}^{(n+l)\times n}$, but in $\mathbb{R}^{n\times n}$. Hence, constraint C_1 becomes

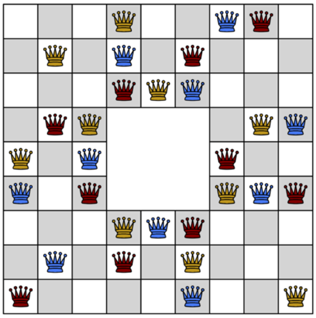
$$\widetilde{C}_1 := \left\{ X \in \mathbb{R}^{n\times n} : x_{ik} \in \{0,1\}, \forall (i,k) \in I \times K \text{ and } \sum_{k=1}^{m} x_{ik} = 1, \forall i \in I \right\},$$

and the set C_3 must be modified and replaced by

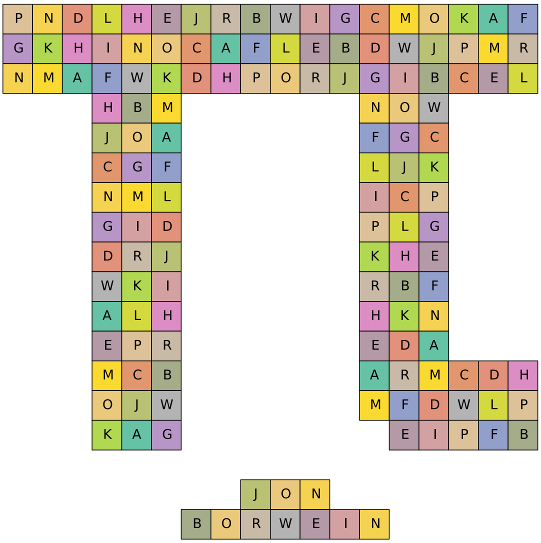
$$\widetilde{C}_3 := \left\{ X \in \{0,1\}^{n\times n} : \forall k = 1, \ldots, n-1, \exists \{i,j\} \in E \text{ s.t. } x_{i,k} x_{j,k+1} = 1 \right\}.$$

We have observed that the rate of success of DR is decreased if C_2 is removed, and that it is better to replace it by the redundant constraint $\widetilde{C}_2 := \mathbb{R}^{n\times n}$, see the experiment shown

(a) Knights on a classic chessboard.



(b) 10-queens puzzle in a chessboard with a hole.



(c) 'π-zzle'.

Fig. 9 Solution to the puzzles in Fig. 8 computed with DR. For 10 random starting points, the average (maximum) time spent for puzzles **a**, **b** and **c** was 0.23, 3.32 and 252.82 seconds (0.35, 11.49 and 424.67 seconds), respectively

in Fig. 22. Note that constraint C_4 forces the path to start on node 1 (a path which may not even exist), so it must be eliminated.

The projector onto $\widetilde{C}_3$ is hard to compute because of the recurrent dependence between all the columns in the matrix X. To overcome this problem, we propose to split the set $\widetilde{C}_3$ into two constraints, one relating each odd column with its following one, and another similar constraint for the even columns. That is, we define the constraints

$$\widetilde{C}_{3,\text{odd}} := \left\{X \in \{0,1\}^{(n+l)\times n} : \forall k = 1,\ldots,\left\lfloor \frac{n}{2} \right\rfloor, \exists\{i,j\} \in E \text{ s.t. } x_{i,2k-1}x_{j,2k} = 1\right\},$$
$$\widetilde{C}_{3,\text{even}} := \left\{X \in \{0,1\}^{(n+l)\times n} : \forall k = 1,\ldots,\left\lfloor \frac{n-1}{2} \right\rfloor, \exists\{i,j\} \in E \text{ s.t. } x_{i,2k}x_{j,2k+1} = 1\right\},$$

which satisfy $\widetilde{C}_3 = \widetilde{C}_{3,\text{odd}} \cap \widetilde{C}_{3,\text{even}}$, where $\lfloor\cdot\rfloor$ denotes the integer part of a number. Therefore, the solution set of the Hamiltonian path problem is $\widetilde{C}_1 \cap \widetilde{C}_2 \cap \widetilde{C}_{3,\text{odd}} \cap \widetilde{C}_{3,\text{even}}$.

To compute a projection onto $\widetilde{C}_{3,\mathrm{odd}}$ and $\widetilde{C}_{3,\mathrm{even}}$, consider the function $h : \mathbb{R} \mapsto \mathbb{R}$ defined by

$$h(x) := \begin{cases} x & \text{if } x \leq 0.5, \\ 1 & \text{if } x > 0.5, \end{cases}$$

and let us denote by

$$(s^0_{k_1,k_2}, s^1_{k_1,k_2}) = \operatorname{argmin}\left\{\left(1 - h(x_{i,k_1})\right)^2 + \left(1 - h(x_{j,k_2})\right)^2, \{i, j\} \in E\right\},$$

where the lowest index is taken in argmin to avoid multivaluedness. Then, a projection onto $\widetilde{C}_{3,\mathrm{odd}}$ and $\widetilde{C}_{3,\mathrm{even}}$ can be obtained as follows

$$\left(\pi_{\widetilde{C}_{3,\mathrm{odd}}}(Z)\right)[i,k] = \begin{cases} 1 & \text{if } i = s^0_{k,k+1}, k < n \text{ and } k \text{ is odd}, \\ 1 & \text{if } i = s^1_{k-1,k} \text{ and } k \text{ is even}, \\ \min\{1, \max\{0, \operatorname{round}(x_{ik})\}\} & \text{otherwise}; \end{cases}$$

$$\left(\pi_{\widetilde{C}_{3,\mathrm{even}}}(Z)\right)[i,k] = \begin{cases} 1 & \text{if } i = s^0_{k,k+1}, k < n \text{ and } k \text{ is even}, \\ 1 & \text{if } i = s^1_{k-1,k}, 1 < k \text{ and } k \text{ is odd}, \\ \min\{1, \max\{0, \operatorname{round}(x_{ik})\}\} & \text{otherwise}. \end{cases}$$

6.1 Hamiltonian Cycles

A *Hamiltonian cycle* is a Hamiltonian path that is also a cycle, that is, there is a link connecting the last node in the path and the first one. The problem of finding such a cycle can be cast as a Hamiltonian path problem as we show next.

Given a graph $G = (V, E)$, select any node $v \in V$ and make a copy of it, i.e., create a new node v' that is connected with all nodes linked to v. Then, create another two new nodes t and s, and link t with v and s with v' (see Fig. 10).

Since t and s have *degree one* (i.e., they are only linked with another node), every admissible Hamiltonian path in the new graph needs to start in one of these nodes and finish in the other. Thus, after removing t and s, we end up with a path going from v to v'. As these nodes were originally the same, we have actually found a Hamiltonian cycle.

Example 6.1 An example of Hamiltonian path/cycle arises in the *knight's tour problem*. The *knight's path problem* consists in finding a sequence of moves of a knight on a chessboard such that it visits exactly once every square. If the final position of such a path is one knight's move away from the starting position of the knight, the path is called a *knight's cycle*. Thus, to find a knight's cycle, one only needs to build the graph corresponding to the knight's movements on a chessboard, and find a Hamiltonian cycle in the graph. A solution for a 12×12 chessboard computed with DR is shown in Fig. 11.

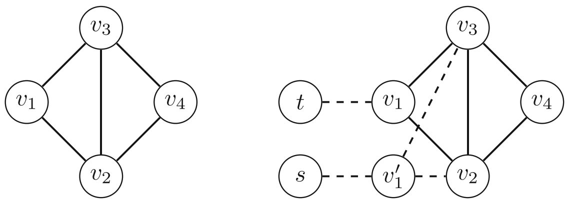


Fig. 10 Hamiltonian cycle reduced to Hamiltonian path

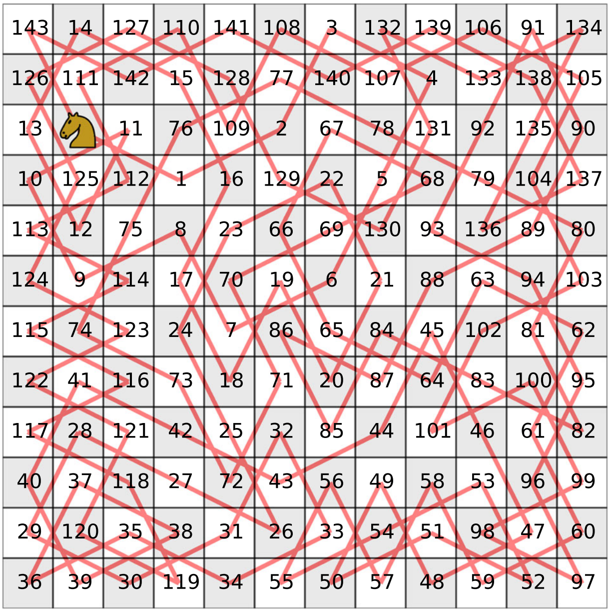


Fig. 11 A knight's cycle on a 12×12 chessboard computed with DR. For 10 random starting points, the method found a solution for every instance, with an average (maximum) time of 1,397 seconds (3,301 seconds, respectively)

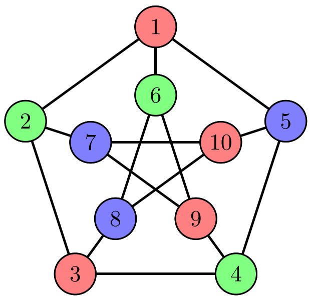
7 Numerical Experiments

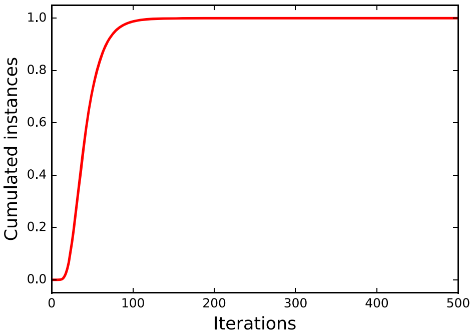
In this section we test the performance of the Douglas-Rachford algorithm for solving a representative sample of the graph coloring problems previously presented. We use the operator $T_{D,C}$ given by (1), where C and D are defined by (2). All codes are written in Python 2.7 and the tests were run on an Intel Core i7-4770 CPU 3.40GHz with 12GB RAM, under Windows 10 (64-bit).

We begin our tests with one of the most well-known graphs: Petersen graph (see Fig. 12). This graph has 10 vertices, 15 edges and can be 3-colored in 120 different ways.

The results of our first experiment are shown in Fig. 13. For 100,000 random starting points and using formulation (9), we report the number of iterations needed by the Douglas-Rachford algorithm until it obtained a solution or it reached 500 iterations. The success rate

Fig. 12 A 3-coloring of Petersen graph





Iterations	Instances	Cumulated
0-99	98,653	98.653%
100-199	1,334	99.987%
200-299	9	99.996%
300-399	2	99.998%
400-499	0	99.998%
Unsolved	2	100%

Fig. 13 Number of iterations spent by DR to find a solution of a 3-coloring of Petersen graph for 100,000 random starting points. On average, each solution was found in 0.00533 seconds. Instances were labeled as "Unsolved" after 500 iterations

was nearly 100% in this experiment: the algorithm was able to find a solution for almost every starting point (with the exception of 2 instances out of 100,000).

In our second experiment, we tested the performance of the Douglas-Rachford algorithm with formulation (9) for finding a valid coloring of complete graphs with 4, 5 and 6 nodes. A complete graph with n vertices has $n(n-1)/2$ edges and can be n-colored in $n!$ different ways. The algorithm was stopped after 500 iterations. DR was able to find a solution for every random starting point for the graphs of 5 and 6 nodes, while it failed in 0.05% of the starting points for the complete graph of 4 nodes. The results are shown in Fig. 14.

We also tested the performance of the Douglas-Rachford algorithm on two wheel graphs of 5 and 6 nodes (see Fig. 15). The results are shown in Fig. 16. A wheel graph with n vertices has $2(n-1)$ edges. If n is even, it can be 4-colored in $4(2^{n-1}-2)$ different ways; if n is odd, it can be 3-colored in 6 different ways.

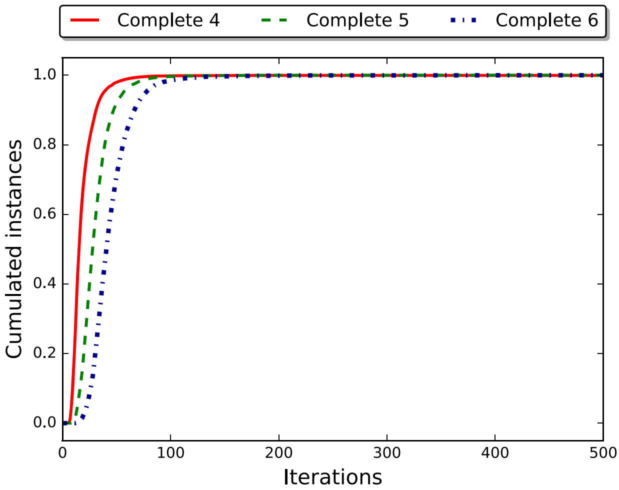
We repeated the same experiment with three cycle graphs (consisting in a given number of vertices connected in a closed chain). A cycle graph with n vertices has n edges. If n is even, it can be 2-colored in 2 different ways; if n is odd, it can be 3-colored in 2^n-2 different ways. The results are shown in Fig. 17.

In our following experiment, whose results are shown in Fig. 18, we compare the performance of the Douglas-Rachford algorithm with and without maximal clique information when it is applied for finding a solution of the windmill graph Wd(6, 5). Observe that, even having increased the number of variables in the feasibility problem, both the rate of success and the rate of convergence (in terms of iterations as well as computing time) are improved.

If $\|Z_k\|$ increases as k increases, the Douglas-Rachford algorithm may serve to detect infeasibility of the corresponding coloring problem, see Fig. 19a–b. In the convex case, according to Fact 2.1(ii), this behavior is assured for infeasible problems. However, due to the lack of convexity, this is not always the case in our context, as shown in Fig. 19c–d. Interestingly, when we removed the extra constraints (7) and (8), which is something that does not change the feasibility of the problems, the algorithm was not able to detect any infeasible problem.

Next, we tested the performance of DR for the 4-nodes and the 5-nodes formulations for the first 50 3-SAT problems with 20 variables and 91 clauses in SATLIB.[2] For each of the formulations, we run the Douglas-Rachford algorithm with and without maximal clique

[2]SATLIB: www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/RND3SAT/uf20-91.tar.gz



	Complete 4		Complete 5		Complete 6	
Iterations	Instances	Cumul.	Instances	Cumul.	Instances	Cumul.
0-99	9,981	99.81%	9,961	99.61%	9,847	98.47%
100-199	14	99.95%	37	99.98%	143	99.9%
200-299	0	99.95%	2	100.0%	6	99.96%
300-499	0	99.95%	0	100.0%	4	100.0%
Unsolved	5	100%	0	100%	0	100%

Fig. 14 Number of iterations spent by DR to find a solution of an n-coloring of a complete graph with n vertices for 10,000 random starting points, with $n = 4, 5, 6$. Each solution was found, on average, in 0.00215 seconds for $n = 4$, 0.00371 seconds for $n = 5$, and 0.00569 seconds for $n = 6$. Instances were labeled as "Unsolved" after 500 iterations

information for 10 random starting points. The results are shown in Table 1. Clearly, the addition of the maximal clique information turns out to be crucial for the success of the Douglas-Rachford algorithm, specially for the 5-nodes formulation.

For an appropriate visualization of the results and comparison of the formulations, we turn to performance profiles (see [16]). We use the modification proposed in [24], since it

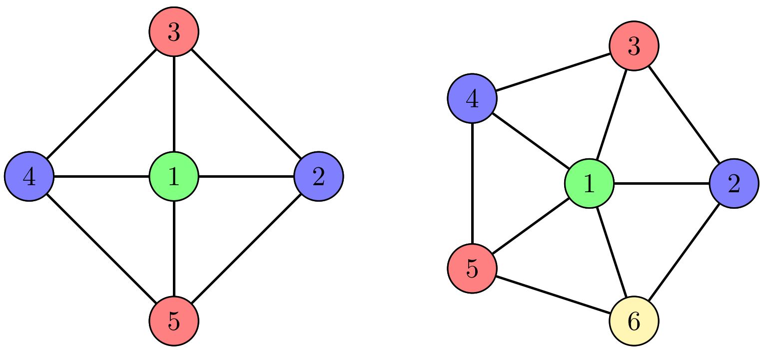
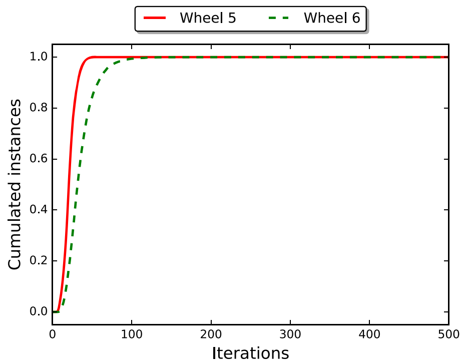


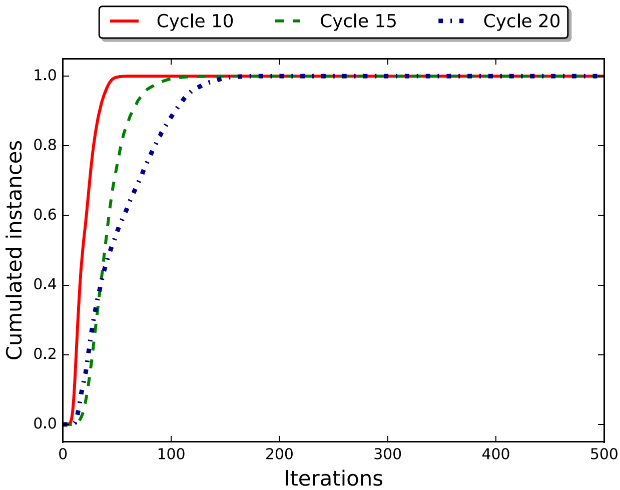
Fig. 15 A 3-coloring and a 4-coloring of two wheel graphs of 5 and 6 nodes



	3-coloring of wheel 5		4-coloring of wheel 6	
Iterations	Instan.	Cumul.	Instan.	Cumul.
0-49	9,988	99.88%	8,332	83.32%
50-99	12	100%	1,600	99.32%
100-149	0	100%	65	99.97%
150-199	0	100%	1	99.98%
200-249	0	100%	1	99.99%
250-499	0	100%	0	99.99%
Unsolved	0	100%	1	100%

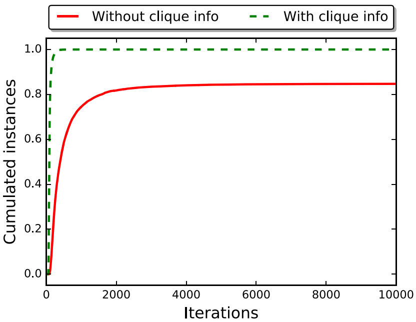
Fig. 16 Number of iterations spent by DR to find a solution of two wheel graphs for 10,000 random starting points. Each solution was found, on average, in 0.00266 seconds for wheel 5, and 0.00455 seconds for wheel 6. Instances were labeled as "Unsolved" after 500 iterations

suits better our experiment, where we have multiple runs for every formulation and problem. Let Φ denote the (finite) set of all formulations. For each formulation $f \in \Phi$, let $t_{f,p}$ be the average time required by DR to solve problem p among all the successful runs, and let us



	2-coloring of cycle 10		3-coloring of cycle 15		2-coloring of cycle 20	
Iterations	Instances	Cumul.	Instances	Cumul.	Instances	Cumul.
0-49	9,967	99.67%	7,276	72.76%	5,438	54.38%
50-99	33	100%	2,640	99.16%	3,349	87.87%
100-149	0	100%	81	99.97%	1,156	99.43%
150-199	0	100%	2	99.99%	57	100%
200-499	0	100%	0	99.99%	0	100%
Unsolved	0	100%	1	100%	0	100%

Fig. 17 Number of iterations spent by DR to find a solution of three cycle graphs for 10,000 random starting points. Each solution was found, on average, in 0.0025 seconds for cycle 10, 0.00561 seconds for cycle 15, and 0.00731 seconds for cycle 20. Instances were labeled as "Unsolved" after 500 iterations

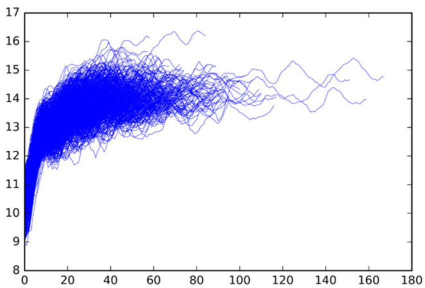


	Without maximal clique information		With maximal clique information	
Iterations	Instances	Cumul.	Instances	Cumul.
0-999	7,448	74.48%	9,999	99.99%
1,000-1,999	728	81.76%	0	99.99%
2,000-2,999	165	83.41%	0	99.99%
3,000-3,999	64	84.05%	0	99.99%
4,000-4,999	29	84.34%	0	99.99%
5,000-5,999	17	84.51%	0	99.99%
6,000-6,999	7	84.58%	0	99.99%
7,000-7,999	7	84.65%	0	99.99%
8,000-8,999	4	84.69%	0	99.99%
9,000-9,999	1	84.7%	0	99.99%
Unsolved	1,530	100%	1	100%

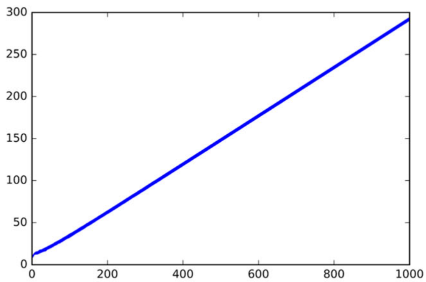
Fig. 18 Comparison of the number of iterations spent by DR to find a solution of the windmill graph Wd(6, 5) for 10,000 random starting points. Complete maximal clique information was used in the right columns. Each solution was found, on average, in 0.13347 seconds without clique information, and 0.02424 seconds with maximal clique information. Instances were labeled as "Unsolved" after 10,000 iterations

denote by $s_{f,p}$ the portion of successful runs for problem p. Compute $t_p^\star := \min_{f\in\Phi} t_{f,p}$ for all $p \in \{1, \ldots, n_p\}$, where n_p is the number of problems in the experiment. Then, for any $\tau \geq 1$, define $R_f(\tau) := \{p \in \{1, \ldots, n_p\}, t_{f,p} \leq \tau t_p^\star\}$; that is, $R_f(\tau)$ is the set of problems for which formulation f is at most τ times slower than the best one. The *performance profile* function of formulation f is given by

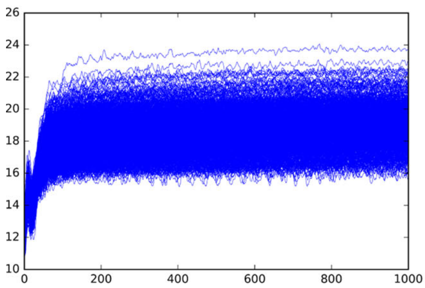
$$\begin{array}{rcl} \pi_f : [1, +\infty) & \longmapsto & [0, 1] \\ \tau & \mapsto & \pi_f(\tau) := \frac{1}{n_p} \sum_{p\in R_f(\tau)} s_{f,p}. \end{array}$$



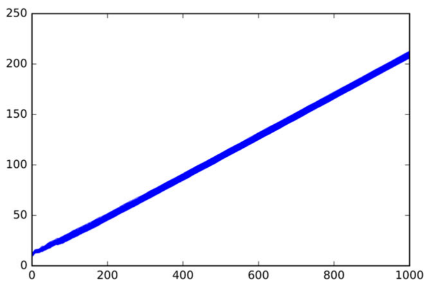
(a) 3-coloring of Petersen graph (feasible).



(b) 2-coloring of Petersen graph (infeasible).



(c) 4-coloring of a 7-complete graph (infeasible).



(d) 3-coloring of a 7-complete graph (infeasible).

Fig. 19 For 1000 random starting points, we represent the iteration k in the horizontal axis and $\|Z_k\|$ in the vertical axis for 1000 iterations of the Douglas-Rachford algorithm

Table 1 Time spent (in seconds) by DR to find a solution of 50 different 3-SAT problems with 20 variables and 91 clauses. For each problem, 10 random starting points were chosen. After 5 minutes without finding a solution, instances where labeled as "Unsolved". Two formulations of the gadgets were considered, with 4 and 5 nodes

	4-nodes without clique info.		4-nodes with clique info.		5-nodes without clique info.		5-nodes with clique info.	
Time	Inst.	Cumul.	Inst.	Cumul.	Inst.	Cumul.	Inst.	Cumul.
0–49	278	55.6%	323	64.6%	0	0.0%	249	49.8%
50–99	60	67.6%	65	77.6%	0	0.0%	69	63.6%
100–149	31	73.8%	24	82.4%	0	0.0%	42	72.0%
150–199	20	77.8%	14	85.2%	0	0.0%	31	78.2%
200–249	16	81.0%	11	87.4%	0	0.0%	21	82.4%
250–299	11	83.2%	10	89.4%	0	0.0%	8	84.0%
Unsolved	84	100%	53	100%	500	100%	80	100%

The value $\pi_f(1)$ indicates the portion of runs for which f was the fastest formulation. When $\tau \to +\infty$, then $\pi_f(\tau)$ shows the portion of successful runs for formulation f. Performance profiles for the 3-SAT experiment from Table 1 are displayed in Fig. 20. It clearly shows that the 4-nodes formulation with clique information is the best one.

In our next numerical experiment, for solving Sudoku puzzles, we compared the performance of DR applied to the Elser's binary feasibility problem formulation [17] (see also [4, Section 6.2]), with the reformulations as a graph coloring ($C_1 \cap C_2 \cap C_3 \cap C_4^{\star}$) and as a graph precoloring ($\overline{C}_1 \cap C_2 \cap C_3$) explained in Section 4. We considered the 95 hard puzzles

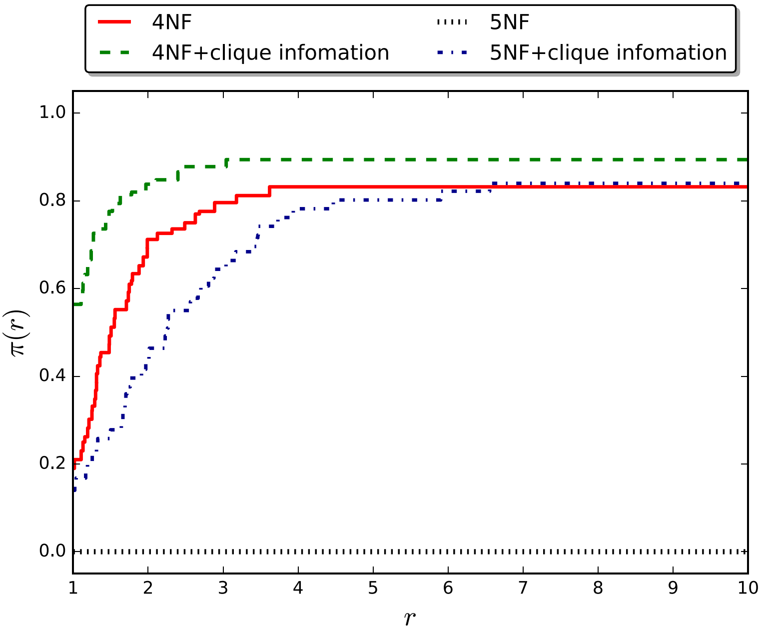
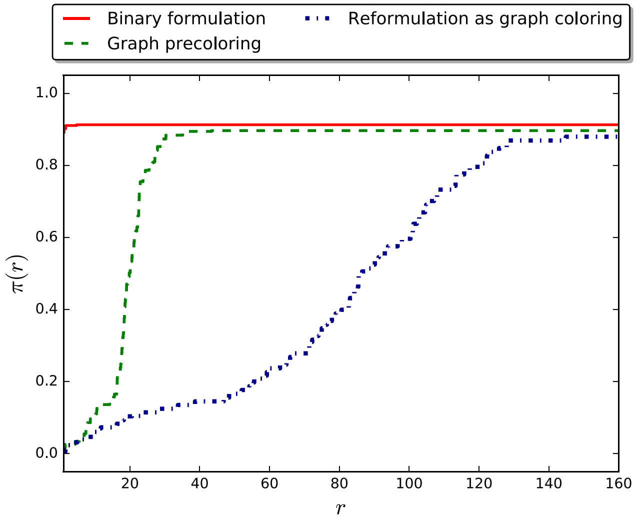


Fig. 20 Performance profile functions for the results in the 3-SAT experiment



	Binary formulation		Graph precoloring		Reformulation as graph coloring	
Time	Inst.	Cumul.	Inst.	Cumul.	Inst.	Cumul.
0-49	1,688	88.8%	1,451	76.4%	261	13.7%
50-99	19	89.8%	173	85.5%	534	41.8%
100-149	15	90.6%	40	87.6%	451	65.6%
150-199	6	90.9%	22	88.7%	267	79.6%
200-249	4	91.2%	12	89.4%	118	85.8%
250-299	2	91.3%	5	89.6%	45	88.2%
Unsolved	166	100%	197	100%	224	100%

Fig. 21 Time spent (in seconds) to find the solution of 95 different Sudoku problems by DR with the graph precoloring, the binary, and the graph coloring formulations. For each problem, 20 starting points were randomly chosen. We stopped the algorithm after a maximum time of 5 minutes, in which case the problem was labeled as "Unsolved". The results are shown in a table and a performance profile

from the library `top95`,[3] which was the one among the libraries tested in [4, Table 2] where DR was most unsuccessful. For each formulation and each puzzle, Douglas-Rachford was run for 20 random starting points. Results and performance profiles are displayed in Fig. 21. As it was expected, the binary formulation was much faster, since this formulation is specifically designed for solving these puzzles. On average, the binary formulation solved a Sudoku in 5.76 seconds, while the graph precoloring formulation needed 33.78 seconds. The worst method was the reformulation as a graph coloring problem, which needed 112.25 seconds on average to solve a Sudoku. Even so, it was surprising to see that the rate of success for these three formulations was very similar, around 90%.

In Table 2 we list the Sudokus for which either the binary or the graph precoloring formulation failed to find a solution for some starting points. It is apparent that the three methods tend to fail on the same Sudokus. The reformulation as graph coloring was clearly the most successful method for Sudoku 19. The graph precoloring formulation had a very bad performance on Sudoku 22, compared to the other two methods. On the other hand, it

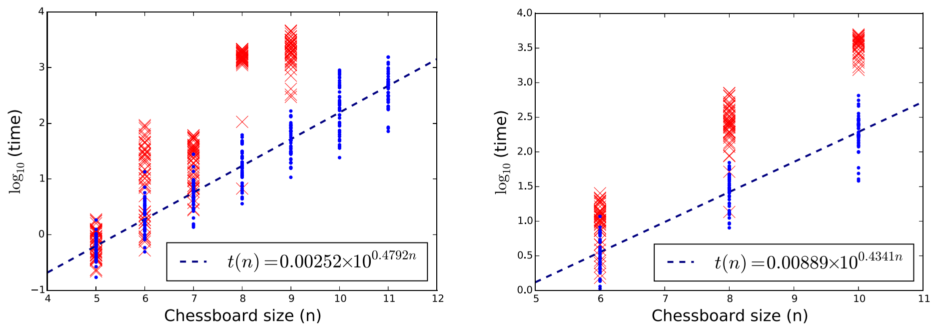
[3] `top95`: http://magictour.free.fr/top95

Table 2 Number of failed runs in either the binary or the graph precoloring formulation. Sudokus not listed here where successfully solved by these two formulations for every starting point (not all the Sudokus where the graph coloring reformulation failed are listed)

Sudoku Number	5	12	13	17	19	22	25	29	38
Binary formulation	0	0	16	19	5	1	20	1	17
Graph precoloring	6	1	18	18	13	19	17	7	15
Reformulation as graph coloring	13	1	16	18	1	9	18	15	12
Sudoku Number	53	59	66	82	83	85	86	90	94
Binary formulation	0	0	14	0	5	18	17	14	19
Graph precoloring	5	3	13	3	5	15	15	8	16
Reformulation as graph coloring	6	1	11	7	4	14	16	14	15

is remarkable that the binary formulation was significantly less successful than the graph precoloring for Sudoku 90, and that it failed to find any solution at all for Sudoku 25. Both the graph precoloring formulation and the reformulation as graph coloring also had troubles with this Sudoku, and were only able to find a solution for 3 and 2 out of the 20 starting points, respectively. This Sudoku is the one shown in Fig. 7.

Finally, in our last experiment, we explored the behavior of DR for solving the knight's tour problem when the size of the chessboard is increased. Results are displayed in Fig. 22, where we analyze both paths and cycles with the two formulations $\widetilde{C}_1 \cap \widetilde{C}_{3,\text{odd}} \cap \widetilde{C}_{3,\text{even}}$ (red crosses) and $\widetilde{C}_1 \cap \widetilde{C}_2 \cap \widetilde{C}_{3,\text{odd}} \cap \widetilde{C}_{3,\text{even}}$ (blue dots). Clearly, the formulation including the redundant constraint $\widetilde{C}_2 = \mathbb{R}^{n\times n}$ is much faster. For this reason, no knight's paths of size 10 or 11 are shown for the formulation without $\widetilde{C}_2$, as the algorithm was stopped before it had enough time to converge. The rate of success of both formulations for paths and cycles was very similar, around 95%. It can be observed an exponential dependence between time and size, which makes DR to be inappropriate for big puzzles. It is remarkable that the line $t(n)$ obtained by linear regression predicts an average time of $t(12) = 1439$ seconds for



(a) Knight's paths. (b) Knight's cycles.

Fig. 22 Time (in $\log_{10}$) required by DR for finding knight's paths and cycles on chessboards of different size. For each size, 50 random starting points were chosen. Blue dots represent instances of the DR method applied with the addition of the redundant constraint $\widetilde{C}_2 = \mathbb{R}^{n\times n}$, while red crosses represent instances where the method was run without $\widetilde{C}_2$. The dotted lines were obtained by linear regression. The algorithm was stopped after a maximum time of 5,000 seconds, in which case the instance is not displayed finding a knight's cycle in a 12×12 chessboard, and this totally fits with the average time of 1397 seconds obtained in the experiment shown in Fig. 11.

8 Conclusion

We showed that the Douglas-Rachford method can be used as a successful heuristic for solving graph coloring problems. A wide collection of examples and variants of these problems were considered along the paper: precoloring and list coloring problems (including Sudoku puzzles), 3-SAT problems, 8-queens puzzles and generalizations, and Hamiltonian path problems (as the knight's tour problem).

A key aspect for the success of the method was to formulate the problems as suitable combinatorial feasibility problems. In this framework, the Douglas-Rachford method had already been proved to be an effective heuristic [3, 4, 17], despite the shortfall of theoretical results that justify its good performance.

We tested the performance of Douglas-Rachford for solving a representative sample of graph coloring problems. It is important to point out that the Douglas-Rachford algorithm is conceptually simple and easy to implement. For simple graphs, the method was able to find a solution for almost every random starting point. For more complex problems, we showed the importance of adding maximal clique information for the success of the method. It is worth mentioning the results in the 3-SAT experiment, where we observed that the use of maximal clique information was decisive.

As expected, in problems where it was possible to successfully apply Douglas-Rachford to the original problem, the method became slower when it was applied to the reduction of the problem to a graph coloring problem. This is the case for Sudoku puzzles, which were solved in our experiments much faster when the method was applied to the formulation of the problem as a binary feasibility problem (on average, 6 and 20 times faster than the graph precoloring formulation and the reformulation as graph coloring, respectively). Nevertheless, it was interesting to observe that the rate of success in finding the solution was high and very similar for the three formulations.

For the knight's tour problem, we showed a clear exponential dependence of the time needed to find a solution with respect to the size of the chessboard. After all, this is not that surprising, due to the NP-completeness of the problem. This shows that the Douglas-Rachford method is probably inadequate for tackling big complex graphs.

In the convex setting, for infeasible problems, the sequence generated by Douglas-Rachford provably tends to infinity (in norm). In our experiments, we obtained some similar results for some particular graphs (see Fig. 19), a behavior that seems to be strongly influenced by the formulation of the feasibility problem.

All the above motivates us to further study in future research why the Douglas-Rachford algorithm can successfully solve this type of nonconvex problems, as well as analyze the detection of infeasibility in nonconvex settings with this algorithm.

Acknowledgements This paper is dedicated to the memory of Jon Borwein, for his enthusiastic comments and suggestions on a very preliminary version of the manuscript. Jon was planning to collaborate with us on this paper after getting back from Canada. Unfortunately, he stayed there forever. We greatly missed his valuable input in the elaboration of this work, and we will surely miss him in the future.

The authors are thankful to the anonymous referees for their careful reading, comments and suggestions that helped to improve the final version of this paper. F.J. Aragón and R. Campoy were partially supported by MINECO of Spain and ERDF of EU, grant MTM2014-59179-C2-1-P. F.J. Aragón was supported by the Ramón y Cajal program by MINECO of Spain and ERDF of EU (RYC-2013-13327) and R. Campoy was supported by MINECO of Spain and ESF of EU (BES-2015-073360) under the program "Ayudas para contratos predoctorales para la formación de doctores 2015".

References

1. Aragón Artacho, F.J., Borwein, J.M.: Global convergence of a non-convex Douglas-Rachford iteration. J. Glob. Optim. **57**(3), 753–769 (2013)
2. Aragón Artacho, F.J., Borwein, J.M., Martín-Márquez, V., Yao, L.: Applications of convex analysis within mathematics. Math. Program. Ser. B **148**(1–2), 49–88 (2014)
3. Aragón Artacho, F.J., Borwein, J.M., Tam, M.K.: Douglas-rachford feasibility methods for matrix completion problems. ANZIAM J. **55**(4), 299–326 (2014)
4. Aragón Artacho, F.J., Borwein, J.M., Tam, M.K.: Recent results on Douglas-Rachford methods for combinatorial optimization problem. J. Optim. Theory. Appl. **163**(1), 1–30 (2014)
5. Aragón Artacho, F.J., Borwein, J.M., Tam, M.K.: Global behavior of the Douglas-Rachford method for a nonconvex feasibility problem. J. Glob. Optim. **65**(2), 309–327 (2016)
6. Baillon, J.B., Bruck, R.E., Reich, S.: On the asymptotic behavior of nonexpansive mappings and semigroups in Banach spaces. Houston J. Math. **4**(1), 1–9 (1978)
7. Bauschke, H.H., Combettes, P.L.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer, New York (2011)
8. Bauschke, H.H., Combettes, P.L., Luke, D.R.: Finding best approximation pairs relative to two closed convex sets in Hilbert spaces. J. Approx. Theory **127**(2), 178–192 (2004)
9. Bauschke, H.H., Koch, V.R.: Projection methods: swiss army knives for solving feasibility and best approximation problems with halfspaces. Contemp. Math. **636**, 1–40 (2015)
10. Bauschke, H.H., Moursi, W.M.: On the Douglas-Rachford algorithm. Math. Program. Ser. A **164**(1–2), 263–284 (2017)
11. Bauschke, H.H., Noll, D.: On the local convergence of the Douglas-Rachford algorithm. Arch. Math. **102**(6), 589–600 (2014)
12. Bell, J., Stevens, B.: A survey of known results and research areas for n-queens. Discrete Math. **309**, 1–31 (2009)
13. Benoist, J.: The Douglas-Rachford algorithm for the case of the sphere and the line. J. Global Optim. **63**(2), 363–380 (2015)
14. Borwein, J.M., Sims, B.Bauschke, H.H., Burachik, R., Combettes, P.L., Elser, V., Luke, D.R., Wolkowicz, H. (eds.): The Douglas-Rachford algorithm in the absence of convexity. Springer, New York (2011)
15. Chaitin, G.J., Auslander, M.A., Chandra, A.K., Cocke, J., Hopkins, M.E., Markstein, P.W.: Register allocation via coloring. Comput. Lang. **6**(1), 47–57 (1981)
16. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. Ser. A **91**(2), 201–213 (2002)
17. Elser, V., Rankenburg, I., Thibault, P.: Searching with iterated maps. Proc. Natl. Acad. Sci. **104**(2), 418–423 (2007)
18. Erdős, P., Rubin, A.L., Taylor, H.: Choosability in graphs. In: Proceeding of the West-Coast Conference on Combinatorics, Graph Theory and Computing, Arcata, California. Congressus Numerantium XXVI, pp. 12–157 (1979)
19. Formanowicz, P., Tanaś, K.: A survey of graph coloring - its types, methods and applications. Foundations of Computing and Decision Sciences **37**(3), 223–238 (2012)
20. Garey, M.R., Johnson, D.S., So, H.C.: An application of graph coloring to printed circuit testing. IEEE Transactions on Circuits and Systems **23**(10), 591–599 (1976)
21. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theoret. Comput. Sci. **1**(3), 237–267 (1976)
22. Hale, W.K.: Frequency assignment: theory and applications. Proc. IEEE **68**(12), 1497–1514 (1980)
23. Hesse, R., Luke, D.R.: Nonconvex notions of regularity and convergence of fundamental algorithms for feasibility problems. SIAM J. Optim. **23**(4), 2397–2419 (2013)
24. Izmailov, A.F., Solodov, M.V., Uskov, E.T.: Globalizing stabilized sequential quadratic programming method by smooth primal-dual exact penalty function. J. Optim. Theor. Appl. **169**(1), 1–31 (2016)
25. Jensen, T.R., Toft, B.: Graph Coloring Problems. Wiley, New York (1995)
26. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)

27. Leighton, F.T.: A graph coloring algorithm for large scheduling problems. J. Res. Nat. Bur. Standard **84**(6), 489–506 (1979)
28. Pardalos, P.M., Mavridou, T., Xue, J.: The graph coloring problem: a bibliographic survey. In: Handbook of Combinatorial Optimization, pp. 1077–1141. Springer, US (1998)
29. Phan, H.M.: Linear convergence of the Douglas-Rachford method for two closed sets. Optimization **65**(2), 369–385 (2016)
30. Pierra, G.: Decomposition through formalization in a product space. Math. Program. **28**, 96–115 (1984)
31. Schaad, J.: Modeling the 8-queens problem and Sudoku using an algorithm based on projections onto nonconvex sets. Master's thesis, Univ. British Columbia (2010)
32. Schwarz, K.: More NP completeness [lecture notes for mathematical foundations of computing, stanford university]. http://web.stanford.edu/class/archive/cs/cs103/cs103.1132/lectures/27/small27.pdf. Accessed 29 august 2017
33. Svaiter, B.F.: On weak convergence of the Douglas-Rachford method. SIAM J. Control Optim. **49**(1), 280–287 (2011)
34. Vizing, V.G.: Coloring the vertices of a graph in prescribed colors. Diskret. Analiz. **29**, 3–10 (1976)