An enhanced formulation for solving graph coloring problems with the Douglas–Rachford algorithm

Francisco J. Aragón Artacho*   Rubén Campoy†   Veit Elser‡

August 6, 2018

Abstract

We study the behavior of the Douglas–Rachford algorithm on the graph vertex-coloring problem. Given a graph and a number of colors, the goal is to find a coloring of the vertices so that all adjacent vertex pairs have different colors. In spite of the combinatorial nature of this problem, the Douglas–Rachford algorithm was recently shown to be a successful heuristic for solving a wide variety of graph coloring instances, when the problem was cast as a feasibility problem on binary indicator variables. In this work we consider a different formulation, based on semidefinite programming. The much improved performance of the Douglas–Rachford algorithm, with this new approach, is demonstrated through various numerical experiments.

Keywords: Douglas–Rachford algorithm, graph coloring, feasibility problem, nonconvex constraints

MSC2010: 47J25, 90C27, 47N10

1 Introduction

Let $G = (V, E)$ be a *graph* with n vertices $V = \{1, \ldots, n\}$ that are connected by edges $E \subset V \times V$. A *proper m-coloring* of the graph G is a mapping $c : V \mapsto K := \{1, \ldots, m\}$, assigning one of the m possible colors to each vertex, such that no two adjacent vertices share the same color, that is,

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

The *graph coloring problem* consists in determining whether it is possible to find a proper m-coloring of the graph G. Many problems arising from different fields can be formulated as graph coloring problems. Some applications include time–tabling and scheduling [28], computer register allocation [14], radio frequency assignment [20], and printed circuit board testing [19]. Since the graph coloring problem is proved to be NP-complete [27], most commonly used solvers rely on heuristics. For a basic reference on graph coloring, algorithms and applications, see, e.g., [24, 29], or the surveys [18, 30].

The aim of this paper is to study the behavior of the Douglas–Rachford (DR) algorithm when it is applied to the Karger–Motwani–Sudan (KMS) [26] semi-definite programming formulation of the graph coloring problem. The KMS formulation is reviewed in Section 3. The DR algorithm belongs to the family of so-called projection algorithms, which arise in convex

---

*Department of Mathematics, University of Alicante, Spain. e-mail: francisco.aragon@ua.es

†Department of Mathematics, University of Alicante, Spain. e-mail: ruben.campoy@ua.es

‡Department of Physics, Cornell University, Ithaca, NY. e-mail: ve10@cornell.edu

optimization. Although the convergence of the DR algorithm is only guaranteed in the convex setting, it has been successfully applied to many nonconvex problems, including those of combinatorial type (see, e.g., [4, 5, 10, 16]). However, the theory in nonconvex settings is very limited, with results on global behavior limited to special sets [6, 12]. In most applications the constraint sets satisfy some type of regularity property, and local convergence can be proved, see, e.g. [11, 21, 33].

This is not the first time that the DR scheme has been used to solve graph coloring problems. It was first employed by Elser et al. [16] for edge-colorings, in particular, colorings that avoid monochromatic triangles. In a more recent paper by Aragón and Campoy [7], the DR algorithm was applied as a heuristic for vertex coloring. In that work, the feasibility problem was expressed in terms of binary indicator variables, the same variables that would be used in an integer programming formulation. This formulation was easily adapted to solve (using DR) variants and generalizations of the graph coloring problem, including list coloring, partial coloring, and finding Hamiltonian cycles.

Our numerical experiments indicate that the KMS formulation appears to be superior to the indicator variable formulation, when using the DR heuristic. While we do not have an interpretation of this result, it is empirically supported on a wide spectrum of problem instances.

The remainder of the paper is organized as follows. In Section 2 we recall some preliminary notions and results. The KMS formulation is reviewed in Section 3. In Section 4 we give details on the DR implementation. Finally, in Section 5 we collect various numerical experiments where we show the good performance of the DR algorithm for the KMS formulation.

2   Preliminaries

Let $\mathbb{E}$ be a finite-dimensional Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and induced norm $\| \cdot \|$. Throughout this paper the space $\mathbb{E}$ will be the Euclidean space $\mathbb{R}^{n \times m}$ of the $n \times m$ real matrices with inner product $\langle A, B \rangle = \operatorname{tr}\left(A^T B\right)$. The induced norm corresponds to the *Frobenius* norm,

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

We denote by $\mathcal{S}^n$ the set of symmetric matrices in $\mathbb{R}^{n \times n}$, while $\mathcal{S}_+^n$ is the set of positive semidefinite matrices.

The *projector* onto a nonempty closed set $C \subseteq \mathbb{E}$ is the set-valued mapping $P_C : \mathbb{E} \rightrightarrows C$ given by

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

and the *reflector* is defined as $R_C := 2P_C - \operatorname{Id}$, where Id denotes the identity operator. Any element $\pi_C(x) \in P_C(x)$ is said to be a *best approximation* or a *projection* of x onto C. If C is also convex, then there exists a unique projection. In fact, a closed set in a Hilbert space is convex if and only if its projector is everywhere single-valued (see, e.g., [3, Theorem 3.2]).

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

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

The DR algorithm is a powerful tool for solving feasibility problems, whenever the individual projectors onto the sets can be easily computed. The DR scheme is defined for problems involving two sets. Next we recall its main properties in the convex setting.

Fact 2.1. *Let $A, B \subseteq \mathbb{E}$ be closed and convex sets and let $\lambda \in {]0,2[}$. Consider the DR operator defined by*

$$T_{A,B,\lambda} := \left(1 - \frac{\lambda}{2}\right) \mathrm{Id} + \frac{\lambda}{2} R_B R_A. \tag{2}$$

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

(i) *If $A \cap B \neq \emptyset$, then $\{x_n\}$ converges to a point $x^\star$ with $P_A(x^\star) \in A \cap B$.*
(ii) *If $A \cap B = \emptyset$, then $\|x_n\| \to +\infty$.*

Proof. (i) See, e.g., [9, Theorem 26.11] or [13, Corollary 5.2.4]. (ii) See [8, Corollary 2.2]. $\square$

Although there is no guarantee of convergence when DR is applied to general (not necessarily convex) closed sets, it can still be applied as a heuristic. In this framework, observe that the DR operator may be multivalued due to the fact that the projection onto nonconvex sets is not necessarily unique. Therefore, the equality in (2) must be replaced by an inclusion, and the iteration takes the form

$$x_{n+1} \in T_{A,B,\lambda}(x_n) := \{x_n + \lambda(b_n - a_n) \in \mathbb{E} : a_n \in P_A(x_n), b_n \in P_B(2a_n - x_n)\}. \tag{3}$$

Finally, we recall that any feasibility problem (1) can be reduced to a two-set problem through Pierra's *product space reformulation* [34]. This permits us to employ the scheme (2) for solving feasibility problems involving more than two sets. The reformulation relies on the equivalence

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

where the constraint sets C and D are defined as

$$C := C_1 \times C_2 \times \ldots \times C_r \quad \text{and} \quad D := \{(x, x, \ldots, x) \in \mathbb{E}^r : x \in \mathbb{E}\}.$$

Moreover, the projectors onto C and D are readily computable as long as the projectors onto $C_1, C_2, \ldots, C_r$ are. Indeed, for any $\mathbf{x} = (x_1, x_2, \ldots, x_r) \in \mathbb{E}^r$, we have

$$P_C(\mathbf{x}) = P_{C_1}(x_1) \times P_{C_2}(x_2) \times \ldots \times P_{C_r}(x_r),$$

$$P_D(\mathbf{x}) = \left(\frac{1}{r} \sum_{i=1}^{r} x_i, \frac{1}{r} \sum_{i=1}^{r} x_i, \ldots, \frac{1}{r} \sum_{i=1}^{r} x_i\right);$$

see [34, Lemma 1.1]. For further details see, e.g., [5, Section 3].

3 Coloring graphs with vertices of the regular simplex

Karger, Motwani and Sudan [26] proposed using the geometry of the regular simplex to formulate the graph vertex-coloring problem. This geometrical encoding of the problem, which respects all its symmetries, is well suited to projection based algorithms.

Suppose that we have a proper m-coloring of the graph $G = (V, E)$ given by $c : V \mapsto K$. The m-coloring c can be represented by a matrix as follows. Let $u_1, u_2, \ldots, u_m \in \mathbb{R}^{m-1}$ be the vertices of a standard centered regular $(m-1)$-simplex. Then, $\{u_1, u_2, \ldots, u_m\}$ are m unit vectors whose pairwise dot products are equal to $\frac{-1}{m-1}$, since

$$\langle u_1 + \cdots + u_m, u_1 + \cdots + u_m \rangle = 0.$$

Each of the vertices of the $(m-1)$-simplex shall represent one of the m colors. Hence, the m-coloring of the graph G can be recovered from the matrix

$$U_c := [u_{c(1)}, u_{c(2)}, \ldots, u_{c(n)}] \in \mathbb{R}^{(m-1) \times n}, \tag{4}$$

whose rows are vertices of the $(m-1)$-simplex (possibly repeated). Finally, let us construct the *Gram matrix* associated with W_c, namely,

$$W_c := U_c^T U_c \in \mathbb{R}^{n \times n}. \tag{5}$$

One can easily check that W_c satisfies the following properties (see, e.g. [22, Theorem 7.2.10]):

(P1) $W_c \in \mathcal{S}_+^n$,

(P2) $\operatorname{rank}(W_c) \leq m - 1$,

(P3) $W_c \in \left\{ 1, \frac{-1}{m-1} \right\}^{n \times n}$ and some of the entries of $W_c = [w_{ij}]$ are determined as follows:

$$w_{ii} = 1, \ \forall i \in V, \quad \text{and} \quad w_{ij} = \frac{-1}{m-1}, \ \forall \{i, j\} \in E.$$

Therefore, every valid m-coloring of the graph G leads to a matrix having properties (P1)–(P3). In fact, this is an equivalence, as we shall show after the next illustrative example.

Example 3.1. *Consider a graph $G = (V, E)$ where the set of vertices is $V = \{1, 2, 3, 4, 5\}$, and the set of edges is $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 5\}\}$. A proper 3-coloring of G is shown in Figure 1(a). We identify each of the colors with one of the vertices $u_1, u_2, u_3 \in \mathbb{R}^2$ of a standard centered regular 2-simplex (see Figure 1(b)), where*

$$u_1 = (1, 0)^T, \quad u_2 = \frac{1}{2}\left(-1, \sqrt{3}\right)^T \quad \text{and} \quad u_3 = \frac{1}{2}\left(-1, -\sqrt{3}\right)^T.$$

Then the matrix representation of c given in (5) becomes

$$W_c = U_c^T U_c = \begin{pmatrix} \boxed{1} & \boxed{-0.5} & \boxed{-0.5} & 1 & -0.5 \\ \boxed{-0.5} & \boxed{1} & \boxed{-0.5} & \boxed{-0.5} & 1 \\ \boxed{-0.5} & \boxed{-0.5} & \boxed{1} & -0.5 & \boxed{-0.5} \\ 1 & \boxed{-0.5} & -0.5 & \boxed{1} & -0.5 \\ -0.5 & 1 & \boxed{-0.5} & -0.5 & \boxed{1} \end{pmatrix}, \quad \text{with } U_c = [u_1, u_2, u_3, u_1, u_2].$$

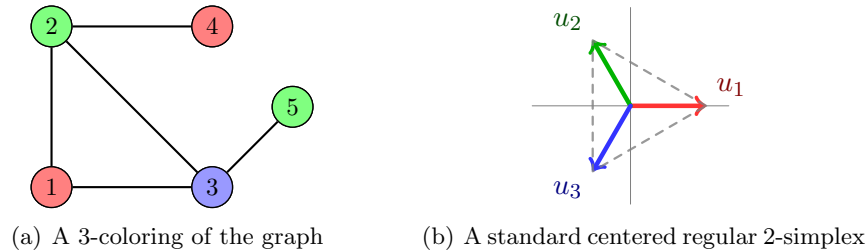*The boxed entries in W_c correspond to those determined by (P3).*

(a) A 3-coloring of the graph   (b) A standard centered regular 2-simplex

Figure 1: Graphical representation of Example 3.1

Proposition 3.1. *Let $G = (V, E)$ be a graph with n nodes and let K be a set of m colors. Consider a matrix $X \in \mathbb{R}^{n \times n}$ that verifies properties (P1)–(P3). Then, there exists a proper m-coloring $c : V \mapsto K$ such that*

$$X = U_c^T U_c,$$

where U_c is given by (4).

Proof. Consider the spectral decomposition $X = Q\Lambda Q^T$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues. Since X is positive definite and has rank not greater than $m-1$, we can assume without loss of generality that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{m-1} \geq 0 = \lambda_m = \cdots = \lambda_n$. Then, we can express

$$X = Q\Lambda Q^T = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} \widehat{\Lambda} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Q_{11}^T & Q_{21}^T \\ Q_{12}^T & Q_{22}^T \end{pmatrix} = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} \widehat{\Lambda} \begin{pmatrix} Q_{11}^T & Q_{21}^T \end{pmatrix},$$

with $\widehat{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_{m-1})$. Hence, we can factorize $X = Y^T Y$, with $Y = \widehat{\Lambda}^{\frac{1}{2}} \begin{pmatrix} Q_{11}^T & Q_{21}^T \end{pmatrix}$. Let $y_1, \ldots, y_n \in \mathbb{R}^{m-1}$ be the columns of Y, i.e., $Y = [y_1|y_2|\cdots|y_n]$. Observe that $y_1, \ldots, y_n$ are unit vectors because X has ones on the diagonal, and thus

$$\langle y_i, y_j \rangle = \begin{cases} 1, & \text{if } y_i = y_j; \\ \frac{-1}{m-1}, & \text{if } y_i \neq y_j; \end{cases} \tag{6}$$

for all $i, j = 1, \ldots, n$. Let us show now that there are at most m distinct vectors among them. To this aim, suppose that $y_{i_1}, y_{i_2}, \ldots, y_{i_{m+1}}$ are $m+1$ different vectors. Consider

$$\widetilde{X} := \begin{bmatrix} y_{i_1}^T \\ y_{i_2}^T \\ \vdots \\ y_{i_{m+1}}^T \end{bmatrix} [y_{i_1}|y_{i_2}|\cdots|y_{i_{m+1}}] = \begin{pmatrix} 1 & \frac{-1}{m-1} & \cdots & \frac{-1}{m-1} \\ \frac{-1}{m-1} & 1 & \cdots & \frac{-1}{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-1}{m-1} & \frac{-1}{m-1} & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{(m+1)\times(m+1)}.$$

It holds that $\text{rank}(\widetilde{X}) \leq \text{rank}(X) \leq m - 1$, since $\widetilde{X}$ is a submatrix of X, but this is a contradiction with the fact that

$$\det(\widetilde{X}) = \left(1 - \frac{m}{m-1}\right)\left(1 + \frac{1}{m-1}\right)^m \neq 0.$$

Therefore, it must hold that $\cup_{j=1}^n \{y_j\} = \{u_1, \ldots, u_r\}$, where $u_1, \ldots, u_r$ are $r \leq m$ distinct vertices of a regular $(m-1)$-simplex (a rotation of the standard simplex). Finally, define $c : V \mapsto K$ by $c(i) = \{k \in \{1, \ldots, r\} : y_i = u_k\}$, so that we trivially get $Y = U_c$, where U_c is as in (4). According to (P3), together with (6), we have that c is a proper m-coloring of G, as claimed. $\square$

In view of Proposition 3.1, finding a proper m-coloring of a graph with n vertices is equivalent to finding an $n \times n$ matrix verifying properties (P1), (P2) and (P3). In this work, the latter will be tackled by solving the following feasibility problem:

$$\text{Find } X \in C_1 \cap C_2 \subseteq \mathbb{R}^{n\times n}, \tag{7}$$

where the constraint sets are defined by

$$C_1 := \left\{ X \in \left\{1, \frac{-1}{m-1}\right\}^{n\times n} : x_{ii} = 1, \forall i \in V \text{ and } x_{ij} = \frac{-1}{m-1}, \forall \{i,j\} \in E \right\}, \tag{8a}$$

$$C_2 := \left\{ X \in \mathcal{S}_+^n : \text{rank}(X) \leq m-1 \right\}. \tag{8b}$$

Remark 3.1. *One advantage of the feasibility problem* (7) *is the avoidance of equivalent colorings in the following sense. Suppose that $c : V \mapsto K$ is a proper m-coloring of a graph G, and let W_c be its associated matrix given by* (5). *For any permutation of the colors, $\sigma : K \mapsto K$, we have that $\sigma \circ c$ is also a proper m-coloring of G, so there exist many equivalent valid colorings. However, observe that $W_{\sigma \circ c} = W_c$, and thus all of them lead to a unique solution of* (7).

5

3.1 Modeling precoloring problems

The *precoloring problem* consists in obtaining a proper coloring of a graph in which the color of some nodes are predefined. Let $\widetilde{V} \subseteq V$ be the subset of precolored nodes and denote by $p_i \in K$ the preassigned color to node $i \in \widetilde{V}$. The task then is to find a coloring $c : V \mapsto K$ such that

$$c(i) \neq c(j), \text{ for all } \{i,j\} \in E \quad and \quad c(i) = p_i, \text{ for all } i \in \widetilde{V}. \tag{9}$$

Notice that any coloring satisfying (9) also verifies

$$c(i) \neq c(j), \text{ for all } \{i,j\} \in E \quad and \quad c(i) = c(j) \Leftrightarrow p_i = p_j, \text{ for all } i,j \in \widetilde{V}. \tag{10}$$

In fact, both conditions can be shown to be equivalent in the following sense. Suppose that $c : V \mapsto K$ is a coloring verifying (10). Then, for any permutation of the colors $\sigma : K \mapsto K$ such that $\sigma(c(i)) = p_i$ for all $i \in \widetilde{V}$, one can easily check that $\sigma \circ c$ is a proper coloring for which (9) holds.

Therefore, we shall focus on finding colorings fulfilling condition (10). The matrix W_c constructed from c as in (5), shall verify now (P1), (P2) and

(P3') $W_c \in \left\{1, \frac{-1}{m-1}\right\}^{n \times n}$ and some of the entries of $W_c = [w_{ij}]$ are determined as follows:

$$w_{ij} = 1, \; \forall \{i,j\} \in \widehat{I} \quad and \quad w_{ij} = \frac{-1}{m-1}, \; \forall \{i,j\} \in \widehat{E};$$

where $\widehat{I} := \{\{i,i\} : i \in V\} \cup \left\{\{i,j\} \subseteq \widetilde{V} : p_i = p_j\right\}$ and $\widehat{E} := E \cup \left\{\{i,j\} \subseteq \widetilde{V} : p_i \neq p_j\right\}$.

The new modified property (P3') can be incorporated into the formulation of the feasibility problem (7) by replacing the constraint C_1 by

$$\widehat{C}_1 := \left\{ X \in \left\{1, \frac{-1}{m-1}\right\}^{n \times n} : x_{ij} = 1, \forall \{i,j\} \in \widehat{I} \text{ and } x_{ij} = \frac{-1}{m-1}, \forall \{i,j\} \in \widehat{E}\right\}. \tag{11}$$

Example 3.2 (Example 3.1 revisited). *Consider the graph in Example 3.1 and suppose that node 2 is precolored red (R), and nodes 4 and 5 are precolored blue (B). The precoloring problem is shown in Figure 2(a). Following the notation established above, we have*

$$\widehat{I} = \{\{i,i\} : i \in V\} \cup \{\{4,5\}\} \quad and \quad \widehat{E} = E \cup \{\{2,4\}, \{2,5\}\} = E \cup \{\{2,5\}\}.$$

The unique solution to the feasibility problem $\widehat{C}_1 \cap C_2$ is the matrix

$$W_c = \begin{pmatrix} \boxed{1} & \boxed{-0.5} & \boxed{-0.5} & 1 & 1 \\ \boxed{-0.5} & \boxed{1} & \boxed{-0.5} & \boxed{-0.5} & \boxed{\boxed{-0.5}} \\ \boxed{-0.5} & \boxed{-0.5} & \boxed{1} & -0.5 & \boxed{-0.5} \\ 1 & \boxed{-0.5} & -0.5 & \boxed{1} & \boxed{1} \\ 1 & \boxed{\boxed{-0.5}} & \boxed{-0.5} & \boxed{1} & \boxed{1} \end{pmatrix},$$

where the boxed entries in W_c correspond to those determined by (P3'). The entries whose values are fixed by (P3') but not by (P3) are marked with a double-box.

Suppose that we obtain the factorization $W_c = U_c^T U_c$, with $U_c = [u_1, u_2, u_3, u_1, u_1]$. The 3-coloring determined by U_c is represented in Figure 2(b). Then, in order to make this coloring consistent with the precoloring of the vertices, we need to suitably permute the set of colors. Precisely, we require $\sigma(G) = R$ and $\sigma(R) = B$. It must therefore be $\sigma(B) = G$. The permuted 3-coloring consistent with the precoloring, given by $U_{\sigma \circ c} = [u_3, u_1, u_2, u_3, u_3]$, is shown in Figure 2(c).

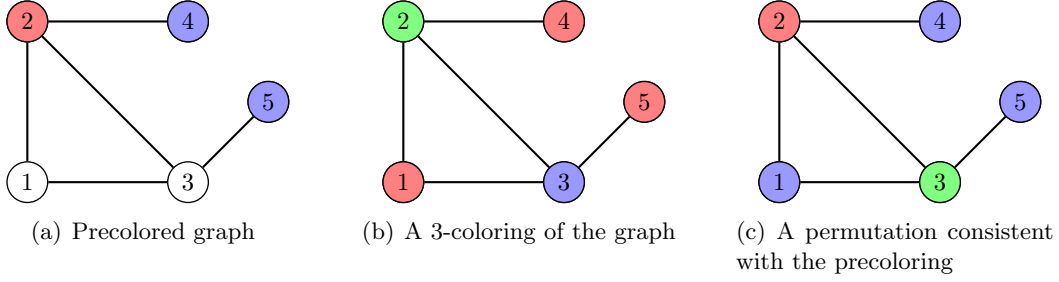(a) Precolored graph (b) A 3-coloring of the graph (c) A permutation consistent with the precoloring

Figure 2: Graphical representation of Example 3.2

4 Implementation of the Douglas–Rachford algorithm

In order to apply the DR algorithm to feasibility problems, and (7) in particular, it must be possible to efficiently compute the projections onto the two constraint sets, in our case (8). This is indeed the case, as shown in the following results.

Proposition 4.1 (Projection onto C_1)**.** *Consider any $X = (x_{ij}) \in \mathbb{R}^{n \times n}$. A projection of X onto the set C_1 defined in (8a) is given componentwise by*

$$
\left(\pi_{C_1}(X)\right)[i,j] = \begin{cases} 1, & \text{if } x_{ij} > \frac{m-2}{2(m-1)} \text{ and } \{i,j\} \notin E, \text{ or } i = j; \\ \frac{-1}{m-1}, & \text{if } x_{ij} \leq \frac{m-2}{2(m-1)} \text{ and } i \neq j, \text{ or } \{i,j\} \in E. \end{cases} \tag{12}
$$

A projection of X onto the set $\widehat{C}_1$ in (11) is given componentwise by

$$
\left(\pi_{\widehat{C}_1}(X)\right)[i,j] = \begin{cases} 1, & \text{if } x_{ij} > \frac{m-2}{2(m-1)} \text{ and } \{i,j\} \notin \widehat{E}, \text{ or } \{i,j\} \in \widehat{I}; \\ \frac{-1}{m-1}, & \text{if } x_{ij} \leq \frac{m-2}{2(m-1)} \text{ and } \{i,j\} \notin \widehat{I}, \text{ or } \{i,j\} \in \widehat{E}. \end{cases} \tag{13}
$$

Proof. Clearly, the projector of X onto C_1 can be computed componentwise. Taking into account the constraints in (8a), the projection of an entry x_{ij} is 1 if $i = j$, and is $\frac{-1}{m-1}$ if $\{i,j\} \in E$. Otherwise, it is equal to $P_{\{1, \frac{-1}{m-1}\}}(x_{ij})$. As the middle point between these two values is $\frac{m-2}{2(m-1)}$, then (12) follows. The proof of (13) is analogous. $\square$

Proposition 4.2 (Projection onto C_2)**.** *Let $X \in \mathcal{S}^n$ and consider its spectral decomposition $X = Q\Lambda Q^T$, with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. A projection of X onto the set C_2 defined in (8b) is given componentwise by*

$$
\pi_{C_2}(X) = Q\Lambda^+_{m-1}Q^T, \tag{14}
$$

where $\Lambda^+_{m-1} = \mathrm{diag}\left(\max\{0, \lambda_1\}, \ldots, \max\{0, \lambda_{m-1}\}, 0, \ldots, 0\right)$.

Proof. See, e.g., [35, Proposition 3.11]. $\square$

Remark 4.1. *According to Propositions 4.1 and 4.2, computing a projection onto C_1 is a simple rounding operation, while a projection onto C_2 requires the computation of the spectral decomposition of an $n \times n$ matrix. From a computational point of view, the former is not a problem but the later may be time-consuming, especially for big problems. However, observe that we do not need to compute the whole spectrum in (14), but only the $m - 1$ largest eigenvalues and their associated eigenvectors. In large-scale problems, m is usually much smaller than n and hence π_{C_2} can be computed reasonably fast.*

Constraint non-convexity manifests itself in the equality case of the conditionals in Proposition 4.1, and the case of degenerate eigenvalues in Proposition 4.2. Neither of these can be acted upon in practice, given the finite precision of the computations.

Remark 4.2. *In order to find $\pi_{C_2}(X)$, Proposition 4.2 requires the matrix X to be symmetric. Observe that, according to (12) and by definition of C_2, we get that*

$$\pi_{C_1}(X), \pi_{C_2}(X) \in \mathcal{S}^n, \quad \text{for all } X \in \mathcal{S}^n.$$

Hence, since $\mathcal{S}^n$ is a subspace, the iterates generated by DR (3) will remain symmetric (with due attention to numerical precision), as long as the initial point is chosen in $\mathcal{S}^n$.

There are several options for implementing the DR algorithm. The simplest choice would be to directly apply DR in the original space $\mathbb{R}^{n \times n}$, since the feasibility problem to be solved (7) only involves two constraint sets. Then, we can iterate by using either $T_{C_1, C_2, \lambda}$ or $T_{C_2, C_1, \lambda}$. On the other hand, although the product space reformulation is typically employed for feasibility problems involving more than two sets, it can still be applied to two sets. In this way, we obtain two additional implementations by either using the operator $T_{D, C, \lambda}$ or $T_{C, D, \lambda}$. The purpose of the next section is to numerically compare these different implementations. In our numerical tests we observed that the numerical behavior of $T_{D, C, \lambda}$ and $T_{C, D, \lambda}$ is similar; thus, to simplify, we only show the results for the operator $T_{D, C, \lambda}$, whose *shadow* $P_D \circ T_{D, C, \lambda}$ is easier to track, as it can be identified with a sequence in the original space $\mathbb{R}^{n \times n}$.

5 Numerical experiments

In this section we run various numerical experiments to test the performance of the DR algorithm for solving different graph coloring problems. We compare the formulation discussed in Section 3 with the one recently proposed in [7]. To distinguish them, we shall refer to the model proposed in [7] as the *binary formulation*, and to the new one developed in Section 3 as the *rank formulation*.

In each of the next five subsections we run illustrative experiments on different families of graphs: the Queens_n^2 puzzles, random colorable graphs, the windmill graphs, Sudokus, and the DIMACS benchmark instances. Each of these families is employed for a different purpose. We start with a difficult coloring problem, the Queens_n^2 puzzle, where we show the effect that the parameter λ has in the different implementations. We also use these puzzles to draw attention to something that is usually overlooked: finite machine precision. Next, to test how the method scales, we run an experiment on random colorable graphs with controlled asymptotic complexity. The windmill graphs and the Sudoku puzzles are used to show that the rank formulation is superior to the binary formulation, even when we allow maximal clique information. We finish this experimental section by testing the algorithm on the DIMACS benchmark instances, a widely used collection of diverse graph types.

Unless otherwise stated, the stopping criterion used for the implementation $T_{A, B, \lambda}$ was

$$\text{Error}_{A,B}(x_k) := \|P_B(P_A(x_k)) - P_A(x_k)\| \leq 10^{-10}, \tag{15}$$

where x_k is the current iterate, in which case the solution attempt was labeled as successful. All codes were written in Python 2.7 and the tests were run on an Intel Core i7-4770 CPU 3.40GHz with 32GB RAM, under Windows 10 (64-bit).

5.1 Queens_n^2 puzzles

A well-known and challenging graph coloring problem is the Queens_n^2 puzzle. This puzzle consists in covering the entire $n \times n$ chessboard with queens of different colors, so that two queens of the same color do not attack each other. The puzzle is equivalent to finding a proper coloring of a particular graph, which has a vertex at each cell of the chessboard and edges between all pairs of vertices (cells) that lie on the same column, row or diagonal. In Table 1

n	2	3	4	5	6	7	8	9	10
$\chi(n)$	4	5	5	5	7	7	9	10	11

Table 1: Chromatic number $\chi(n)$ of the Queens_n^2 graph [31]

we give the chromatic number of the graph for the first nine puzzles. The smallest open case for which the chromatic number is currently unknown is $n = 27$, see [31].

In our first experiment, we analyze how both the implementation and the choice in the relaxation parameter λ affects the behavior of DR for solving this type of puzzle. For each $n \in \{3, 4, \ldots, 10\}$ and each $\lambda \in \{0.25, 0.5, \ldots, 1.75\}$, we ran three different implementations of DR (namely, $T_{C_1,C_2,\lambda}$, $T_{C_2,C_1,\lambda}$ and $T_{D,C,\lambda}$) from 10 random starting points. The results are shown in Figure 3, where the markers correspond to the median among the solved instances. We also show the percentage of instances solved for each value of λ, among all the problems and repetitions. According to these results, it seems that the value of the parameter λ that suits best each of the formulations $T_{C_1,C_2,\lambda}$, $T_{C_2,C_1,\lambda}$ and $T_{D,C,\lambda}$, is $\lambda = 0.75$, $\lambda = 0.5$ and $\lambda = 1$, respectively. To corroborate this conclusion, we visualize the results using performance profiles, which are constructed as follows (see [15] and the modification proposed in [23]).

Performance profile   Let Φ denote the set of formulations, and let $\mathcal{P}$ be a set of N problems. For each formulation $f \in \Phi$, let $t_{f,p}$ be the averaged time required by DR to solve problem $p \in \mathcal{P}$ among all the successful runs, and let $s_{f,p}$ denote the fraction of successful runs for problem p. Compute $t_p^\star := \min_{f \in \Phi} t_{f,p}$ for all $p \in \mathcal{P}$. Then, for any $\tau \geq 1$, define $R_f(\tau) := \{p \in \mathcal{P}, t_{f,p} \leq \tau t_p^\star\}$; i.e., $R_f(\tau)$ 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
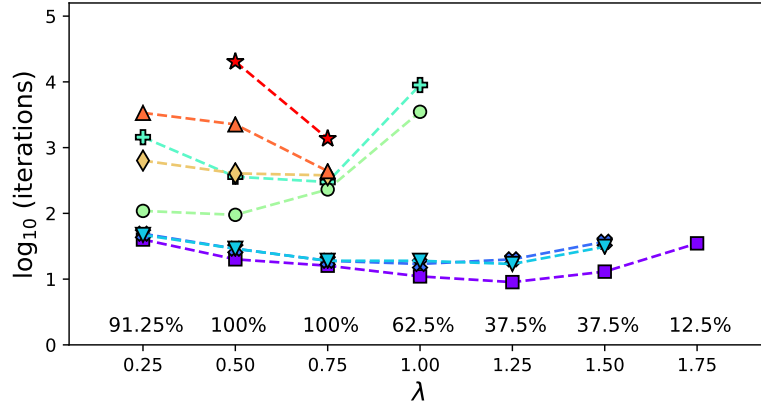
$$\rho_f : \quad [1, +\infty) \quad \longmapsto \quad [0, 1]$$
$$\tau \quad \mapsto \quad \rho_f(\tau) := \tfrac{1}{N} \sum_{p \in R_f(\tau)} s_{f,p}.$$

The value $\rho_f(1)$ indicates the portion of runs for which f was the fastest formulation. When $\tau \to +\infty$, then $\rho_f(\tau)$ gives the fraction of successful runs for formulation f.

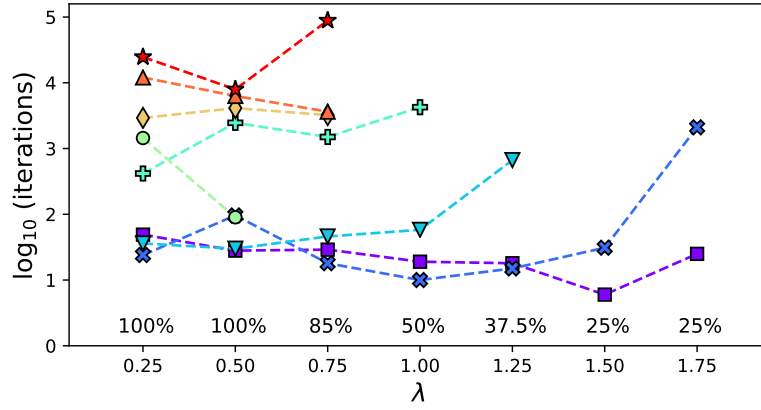We show in Figure 4 the performance profiles for each value of λ and each of the three implementations $T_{C_1,C_2,\lambda}$, $T_{C_2,C_1,\lambda}$ and $T_{D,C,\lambda}$. This corroborates our previous choice of best parameters λ for each implementation. Finally, we compare $T_{C_1,C_2,0.75}$, $T_{C_2,C_1,0.5}$ and $T_{D,C,1}$ in Figure 5, where we can clearly observe that the first implementation dominates the others.

Remark 5.1 (On the machine precision). *Our numerical tests show no systematic effect of the machine precision on the average number of iterations per solution, provided the precision is above a modest threshold of about 6 decimal digits. This is consistent with the chaotic dynamics displayed by DR when solving hard problems.*
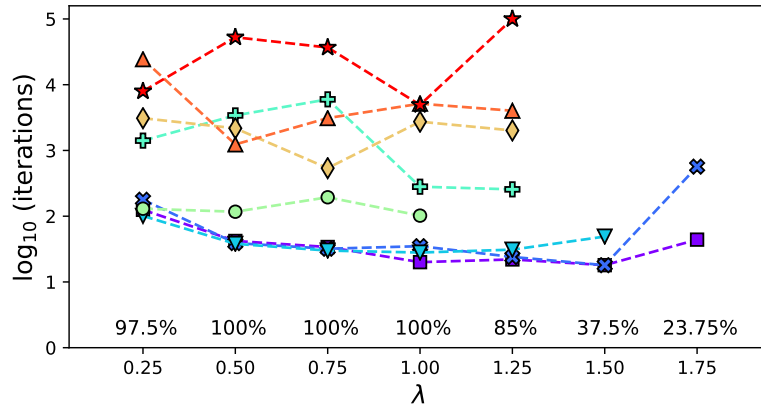
The behavior explained in Remark 5.1 is demonstrated in the next experiment, where $T_{D,C,1}$ was implemented for solving the Queens_6^2 and the Queens_7^2 puzzles. For each problem, the algorithm was run from the same starting point using different values of the machine precision. The stopping criterion (15) was decreased to 10^{-5} to accommodate the reduced precision. We believe this is still adequate to recover a unique, discrete coloring from the Gram matrix. The results of repeating this experiment for 10 different random starting points are shown in Figure 6. In Figure 7 we plot the value of $\mathrm{Error}_{D,C}$ in (15) with respect to the number of iterations for up to 15 digits of precision for one particular random starting point. While these results indicate a high sensitivity to the numerical precision, there is no evidence of a systematic effect. For these experiments, we employed the `mpmath` library [25], which drastically increases the time needed to compute the iterations of the DR algorithm.

(a) DR implemented with $T_{C_1,C_2,\lambda}$

(b) DR implemented with $T_{C_2,C_1,\lambda}$

(c) DR implemented with $T_{D,C,\lambda}$

Figure 3: Results of the Queens_n^2 experiment for three implementations of DR. Each marker corresponds to the median of the solved instances among 10 random starting points. At the bottom of each graph, we show the percentage of solved instances for each value of λ. Instances were considered as unsolved after 100,000 iterations
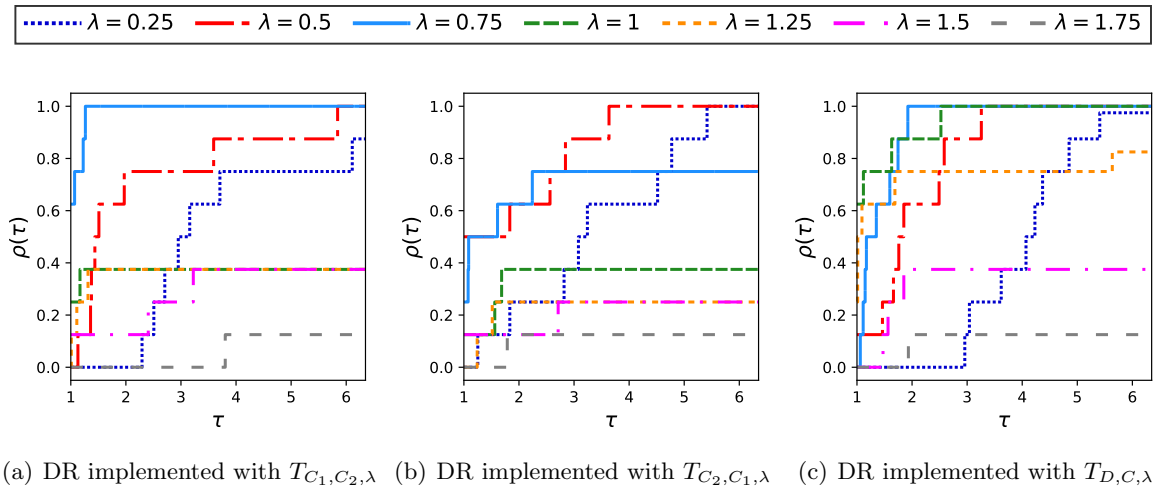
10

(a) DR implemented with $T_{C_1,C_2,\lambda}$ (b) DR implemented with $T_{C_2,C_1,\lambda}$ (c) DR implemented with $T_{D,C,\lambda}$

Figure 4: Performance profiles of the Queens_n^2 experiment for three implementations of DR

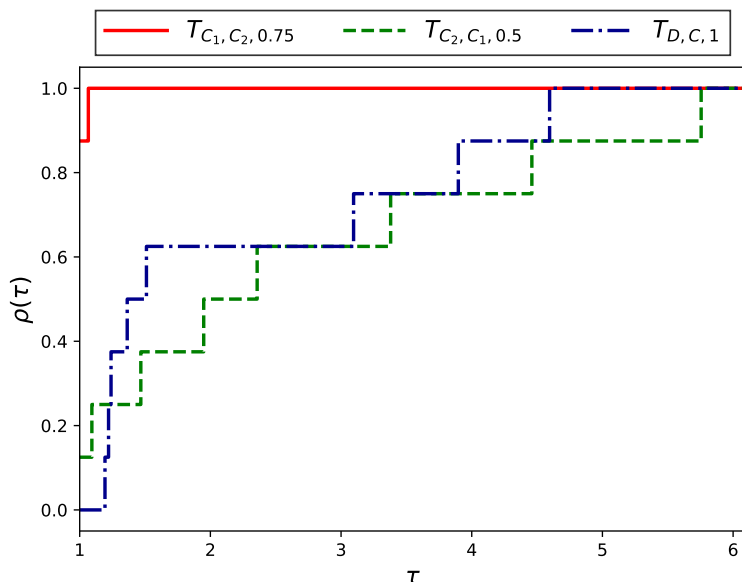
Figure 5: Performance profiles of the Queens_n^2 experiment comparing the implementations $T_{C_1,C_2,0.75}$, $T_{C_2,C_1,0.5}$ and $T_{D,C,1}$

5.2 Random colorable graphs

The hardness of finding a proper coloring of a graph depends on many factors, the single most significant of which is the number of valid colorings. Random colorable graphs are easily constructed, but to be able to draw some consequences from the experiments we run on them, we must generate them in such a way that their complexity is controlled.

We consider the Erdös–Renyi model [17], $\mathbb{G}(\alpha, n)$, which is the ensemble of all graphs with n vertices and $l = \lfloor \alpha n \rfloor$ edges, where $\lfloor \cdot \rfloor$ denotes the integer part, endowed with the uniform measure. Hence, α represents the averaged number of edges per node. The probability that a random graph with this distribution is m-colorable depends on the magnitude of the parameter α. Precisely, the expected number of proper colorings decreases as α increases. There is an *asymptotic threshold* in the colorable-uncolorable transition denoted $\alpha_s(m)$ (see [1, Theorem 1.1]). This means that the probability an m-coloring exists tends to one as n increases, provided that $\alpha < \alpha_s(m)$, and conversely, it converges to zero for $\alpha > \alpha_s(m)$.
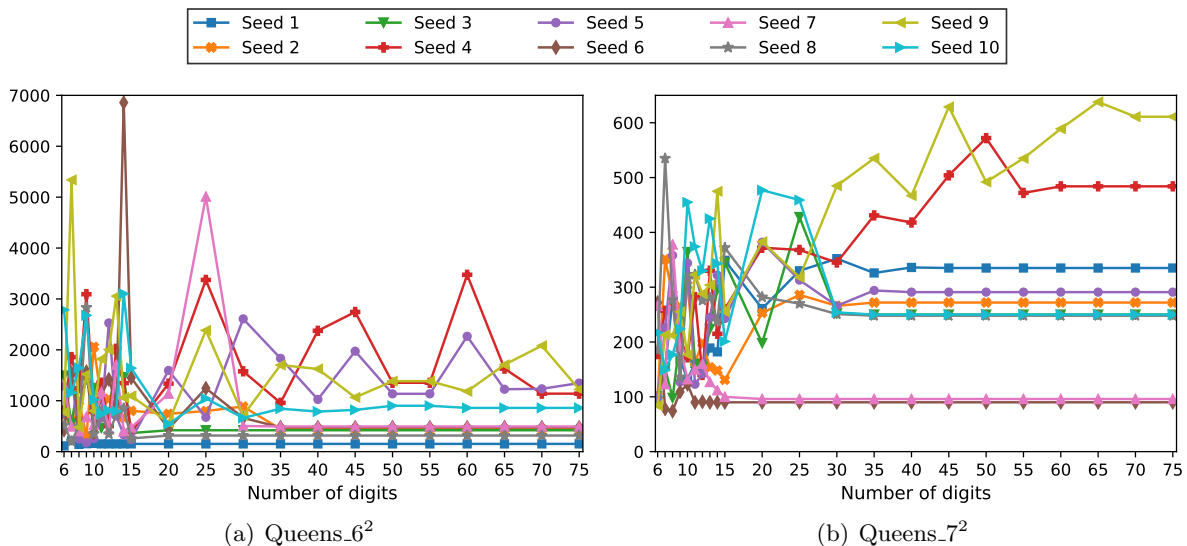
Figure 6: Comparison of the number of number of iterations and the number of digits used in the machine precision for 10 random starting points, when $T_{D,C,1}$ was employed to solve the Queens_6^2 and the Queens_7^2 puzzles. For every starting point and every value of the machine precision, the algorithm found a solution to the puzzle
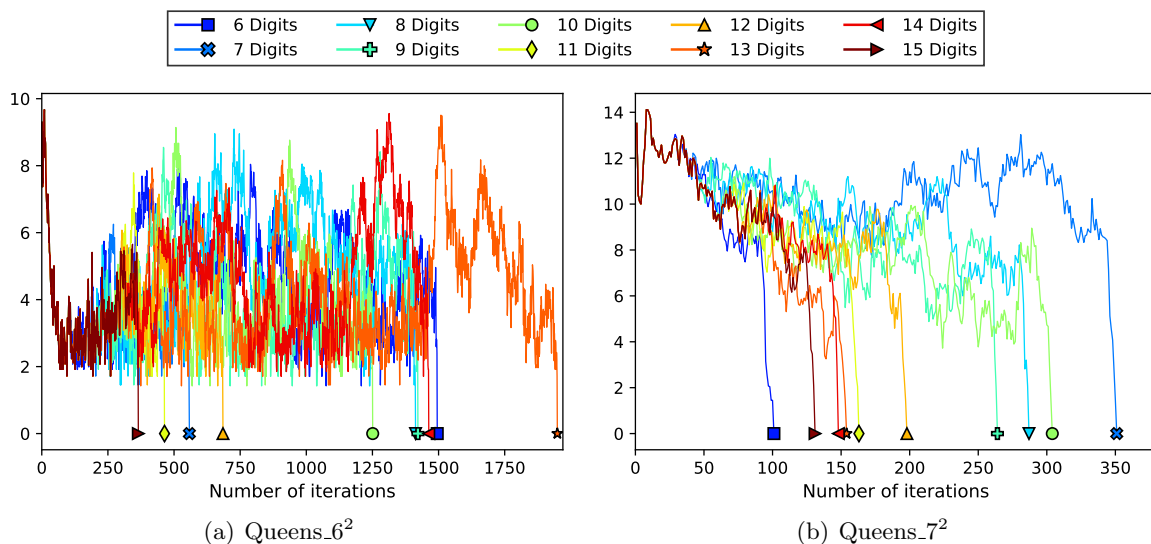

Figure 7: Comparison of the value of Error in (15) and the number of iterations for different number of decimal digits used in the machine precision, when $T_{D,C,1}$ was employed to solve the Queens_6^2 and the Queens_7^2 puzzles

The asymptotic threshold is known to be upper-bounded by $\alpha_s(m) \leq \bar{\alpha}_s(m) := \frac{\log m}{\log \frac{m}{m-1}}$ (see, e.g., [2, Section 2]).

With the number of vertices n and the number of colors m fixed, random graphs sampled from $\mathbb{G}(\bar{\alpha}_s(m), n)$ are at the m-colorability transition and expected to be hard instances, when solvable. In order to avoid non-colorable graphs, the sampling can be modified to ensure the existence of a coloring as follows. First, a partition of V into m groups with approximately equal size is chosen, e.g. consider the equivalence classes defined by the congruence modulo m of the integer vertex labels. Then, $\lfloor \bar{\alpha}_s(m)n \rfloor$ edges are randomly generated from the uniform distribution over the set of all edges connecting two nodes in different groups. Algorithm 1

contains the discussed routine that generates such graphs.

Algorithm 1: Generate an m-colorable random graph with low expected number of m-colorings

---

Input: $V = \{1, \ldots, n\}$, $m \geq 2$

Set $\bar{\alpha}_s(m) := \frac{\log m}{\log \frac{m}{m-1}}$, $E := \emptyset$ and $l = 0$;

while $l < \bar{\alpha}_s(m)n$ **do**

    Generate randomly $e := \{i, j\} \in V \times V$;

    if $e \notin E$ and $(i - j) \not\equiv 0 \pmod{m}$ **then**

        $E = E \cup \{e\}$;

        $l = l + 1$;

Output: $G = (V, E)$

The goal of our next experiment is to show how the DR algorithm complexity grows with respect to the number of vertices in the graph. We make use of colorable random graphs with low expected number of colors so that we have control of the complexity of our instances. For each $m \in \{8, 9, 10\}$ and for each $n \in \{50, 75, \cdots, 200\}$, we generated 5 random graphs using Algorithm 1. Then, for each graph, the DR algorithm was run from 5 different starting points (this makes a total of 25 runs per each pair (m, n)). Based on the results in the Queens_n^2 experiment, we implemented DR with $T_{C_1, C_2, \lambda}$. To confirm our previous choice of the best parameter $\lambda = 0.75$, we repeated the experiment for each $\lambda \in \{0.25, 0.5, \ldots, 1.75\}$. The results shown in Figure 8 confirm that the best choice for general purposes is $\lambda = 0.75$. In Figure 9 we plot the number of iterations needed by $T_{C_1, C_2, 0.75}$ with respect to the size of the graph, for each m. We observe that the number of colors does not have a noticeable effect on the performance of the algorithm. As expected, we come upon an exponential dependence between size and iterations, which is consistent with the NP-hardness of the problem.
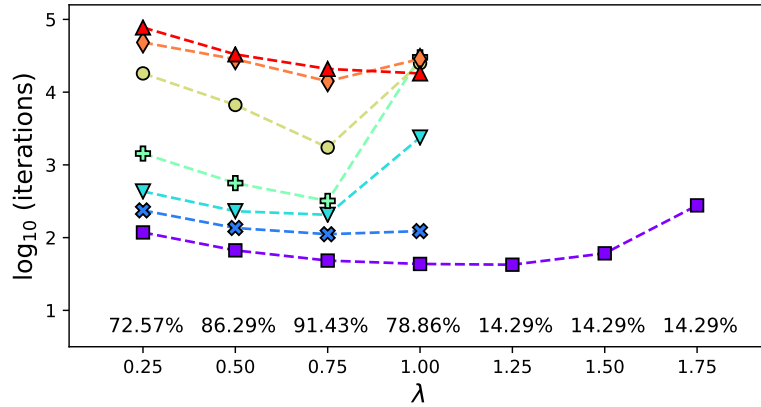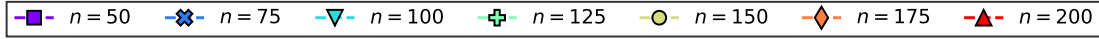
5.3 Windmill graphs

We turn our attention to a very simple graph for which the binary formulation has trouble finding solutions: the so-called *windmill graph* $\mathrm{Wd}(a, b)$, which is constructed for $a \geq 2$ and $b \geq 2$ by joining b copies of a complete graph with a vertices at a shared vertex. Its chromatic number is a, and the graph can be easily colored (there are $a((a-1)!)^b$ different ways to do it). Despite this abundance of valid colorings, all of them are equivalent under a permutation of the colors. Thus, by Remark 3.1, there exists a unique solution to the rank feasibility problem. This is not the case, however, for the binary formulation.
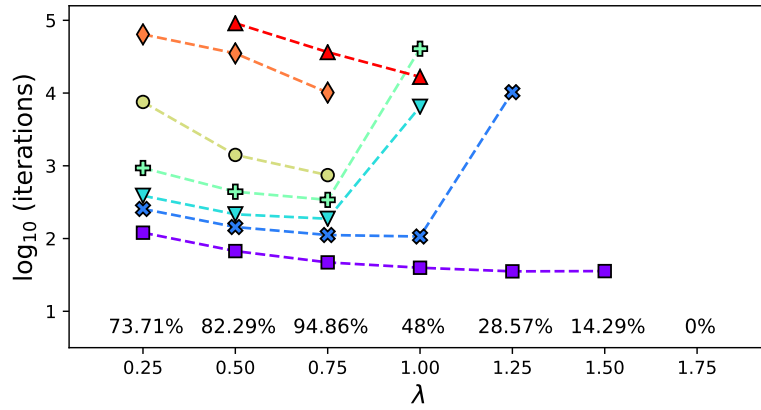
In our experiments with the binary formulation, the DR algorithm fails to find colorings of windmill graphs rather often. We tentatively attribute this to the high multiplicity of solutions in the binary formulation. In [7] we addressed this problem by augmenting the model with information about the maximal cliques of the graph. A *clique* is a subset of nodes whose induced subgraph is complete; that is, a subset where all the vertices are connected to each other. Further, a clique is *maximal* if it is not contained in a larger clique. The set of maximal cliques would normally be unknown (and difficult to find) for general graphs, so a formulation that does not require this information would generally be preferred.

In our next experiment, whose results are shown in Table 2, we compare the binary formulation with and without maximal clique information, and the rank formulation for coloring sixteen windmill graphs of different parameters.
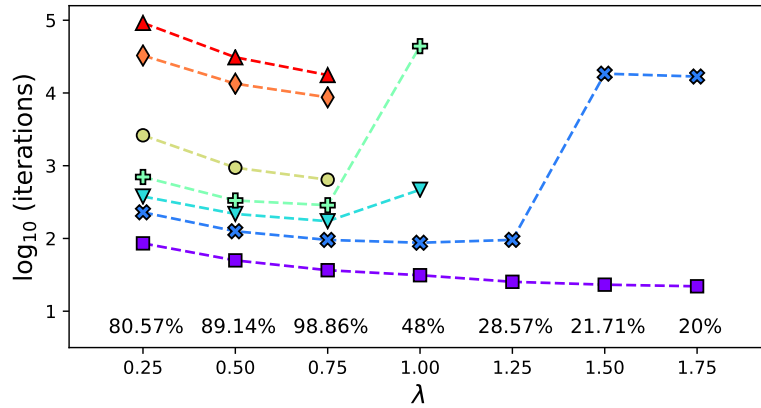
We observe that the addition of maximal clique information is crucial for the success of the binary formulation. Without adding it, the DR algorithm was not able to find any solutions for even modestly large values of a. On the other hand, the superior performance of the

(a) 8 colors

(b) 9 colors

(c) 10 colors

Figure 8: Results of the experiment on m-colorable random graphs for $m = 8, 9, 10$, for the implementation $T_{C_1, C_2, \lambda}$ of DR. Each marker corresponds to the median of the solved instances among 10 random starting points. At the bottom of each graph we show the percentage of solved instances for each value of λ. Instances were considered as unsolved after 100,000 iterations
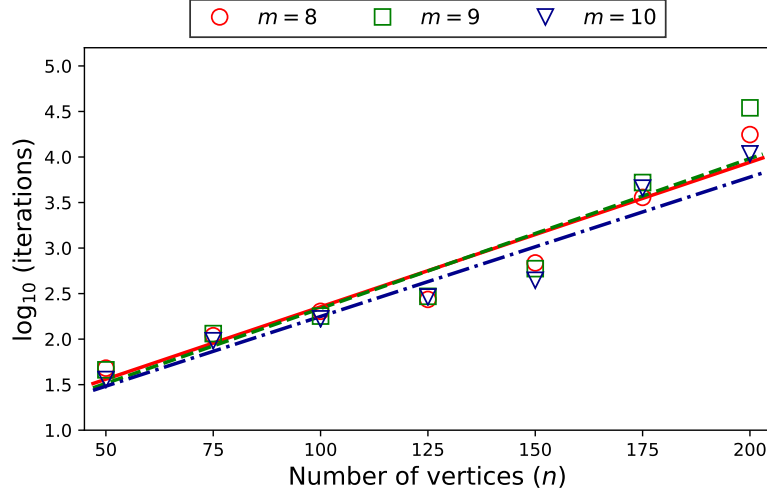
14

Figure 9: Results of the experiment on m-colorable random graphs for $m = 8, 9, 10$, for DR implemented with $T_{C_1, C_2, 0.75}$. Each marker corresponds to the median of the solved instances among 10 random starting points, and the lines were obtained by linear regression among all the solved instances

Wd(a,b)		Binary formulation			Binary formulation with clique info.			Rank formulation		
a	b	Success	Time	Iter.	Success	Time	Iter.	Success	Time	Iter.
5	5	10/10	0.05	226	10/10	0.02	63	10/10	0.01	20
	10	10/10	0.13	375	10/10	0.04	93	10/10	0.02	33
	15	9/10	0.23	503	10/10	0.06	135	10/10	0.03	43
	20	9/10	0.3	521	10/10	0.1	170	10/10	0.05	51
10	5	1/10	1.12	1886	10/10	0.12	200	10/10	0.03	33
	10	0/10	-	-	10/10	0.27	242	10/10	0.08	54
	15	0/10	-	-	10/10	3.47	1729	10/10	0.24	86
	20	0/10	-	-	10/10	5.2	1531	10/10	0.45	109
15	5	0/10	-	-	10/10	0.54	330	10/10	0.06	44
	10	0/10	-	-	10/10	1.69	369	10/10	0.29	92
	15	0/10	-	-	10/10	5.21	588	10/10	0.85	144
	20	0/10	-	-	10/10	13.35	949	10/10	1.69	180
20	5	0/10	-	-	10/10	2.62	642	10/10	0.15	68
	10	0/10	-	-	10/10	12.52	1059	10/10	0.63	119
	15	0/10	-	-	10/10	16.95	729	10/10	1.83	170
	20	0/10	-	-	8/10	31.76	828	10/10	7.3	297

Table 2: Summary of the results of DR for finding proper colorings of windmill graphs. For each formulation, we show the number of solved instances, the averaged time (in seconds) and the averaged number of iterations. Instances were considered as unsolved after 60 seconds

rank formulation for this graph is apparent, both in terms of number of iterations and time. We emphasize again that the rank formulation does not use maximal clique information, and despite this, it achieved a success rate of 100%.

5.4 Sudokus

To test the rank matrix model for precoloring stated in Section 3.1, we turn to the Sudoku data set `top95`[1], which was the one used in the experiments in [5, 7] because it contains 95 hard Sudoku instances.

In our first experiment, we compare the rank formulation (with $T_{C_1,C_2,0.75}$) and the binary formulation for precoloring [7, Section 4], with maximal clique information included in the latter. We also compare with the standard divide-and-concur formulation, where solutions for $n \times n$ puzzles are encoded using four copies of $n \times n \times n$ grids of binary indicator variables (see [5, Section 6.2] for a more detailed explanation), which will be referred to as the *cubic* formulation. For each of these three formulations and each of the 95 puzzles, the DR algorithm was run from 10 random starting points. The performance profiles of the results is displayed in Figure 10.
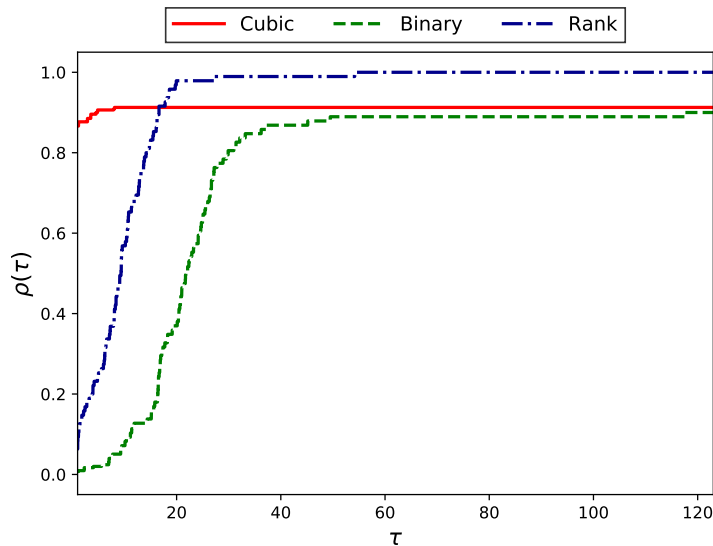

Figure 10: Performance profiles comparing the cubic, binary and rank formulations for solving 95 Sudoku problems. For each problem, 10 starting points were randomly generated. Instances were considered as unsolved after 5 minutes.

The cubic formulation was the fastest in 86.36% of the instances. On average it solved a Sudoku in 4.65 seconds, while the binary and rank formulations needed 35.8 and 13.79 seconds, respectively. Regarding the success of the algorithm, the cubic and binary formulations solved about 90% of the instances. The rank formulation was the clear winner in terms of success, as it solved every single instance, even for those puzzles in the library on which DR has been observed to be highly unsuccessful (see [7, Table 2]).

To further challenge the rank formulation, we performed experiments on the so-called 'nasty' Sudoku (shown in Figure 11). The 'nasty' Sudoku has very low success rate in the cubic formulation (see [5, Section 6.5]), as the algorithm almost always enters a limit cycle (see [5, Table 4]). This is not the case, however, for the rank formulation. In our next experiment we compare the cubic, binary and rank formulations for solving the 'nasty' Sudoku from 100 random starting points. The results are summarized in Figure 11. The rank formulation obtained again a success rate of 100%. The second most successful formulation was the binary one, which was only able to find a solution for 19% of the starting points. So far, we have not been able to find any Sudoku on which the rank formulation failed to find a solution for any starting point.

---

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

	Cubic		Binary		Rank	
Time	Inst.	Cumul.	Inst.	Cumul.	Inst.	Cumul.
0-24	12	12%	15	15%	61	61%
25-49	0	12%	2	17%	36	97%
50-99	0	12%	1	18%	3	100%
100-299	0	12%	1	19%	0	100%
Unsolved	88	100%	81	100%	0	100%

Figure 11: Number of solved instances (right), among 100 random starting points, to find the solution of the 'nasty' Sudoku (left) by DR with the cubic, binary, and rank formulations. For each interval of time (in seconds), we show the number of solved instances and the cummulative proportion of solved instances for each formulation. The algorithm was stopped after a maximum of 5 minutes, in which case the problem was labeled as "Unsolved"

5.5 DIMACS benchmark instances

In our final experiment, we test the rank formulation on the widely used graph coloring library from `DIMACS benchmark instances`[2]. This collection contains various classes of graphs, such as random or quasi-random graphs, problems based on register allocation for variables in real codes, or class scheduling graphs, among others.

The DR algorithm was applied to a wide sample of the aforementioned benchmark instances. Guided by the results in the previous experiments, we used the implementation $T_{C_1,C_2,0.75}$. For each graph, the algorithm was run from 10 random starting points and was stopped after a maximum time of one hour. In Table 3 we present the results of the experiment, as well as the main features of the selected instances. The unsuccessful instances mainly occurred on the very large graphs, on which the algorithm may have succeeded given more time.

6 Conclusions

In the emerging field of projection-based heuristic algorithms for solving combinatorially hard problems, the competition is usually framed to be about the choice of operator (DR, ADMM, etc.). In this study, featuring graph vertex coloring with the DR algorithm, we have shown that the choice of constraint formulation has a very significant effect, and in the end may prove to be even more important than the choice of operator. This conclusion comes from numerical experiments demonstrating, over a wide spectrum of instances, the superiority of the rank-constrained matrix formulation [26] over a previously studied formulation based on binary indicator variables.

The failure mechanism of projection-based heuristic algorithms is trapping on limit cycles. Our experiments indicate that the rank-constrained matrix formulation appears to be immune to this problem, achieving 100% success rates independent of the choice of starting point. Most notable is the success on the so-called 'nasty' Sudoku (treated as a graph pre-coloring instance), on which all other known formulations have no better than a 20% success rate. This approach also does not come at a great cost in implementation, and is able to solve graph coloring instances from the DIMACS benchmark collection with hundreds of vertices and thousands of edges, often in much less than an hour.

We speculate that the good performance of the rank-constrained matrix formulation may be linked to the elimination of the high symmetry-based solution multiplicity of competing

---

[2]DIMACS benchmark instances: http://cse.unl.edu/~tnguyen/npbenchmarks/graphcoloring.html

Instances	Vertices	Edges	Colors	Success	Iter	Time (s)
fpsol2.i.1	496	11,654	65	10/10	8,984	463.94
fpsol2.i.2	451	8,691	30	10/10	13,316	495.94
fpsol2.i.3	425	8,688	30	10/10	14,454	480.27
inithx.i.1	864	18,707	54	10/10	16,174	2,443.43
inithx.i.2	645	13,979	31	10/10	20,049	1,500.45
inithx.i.3	621	13,969	31	10/10	20,604	1,432.43
le450_15a	450	8,168	15	4/10	61,365	1,944.35
le450_15b	450	8,169	15	8/10	65,537	2,076.54
le450_15c	450	16,680	15	10/10	5,464	173.1
le450_15d	450	16,750	15	10/10	19,718	619.74
le450_25a	450	8,260	25	10/10	1,938	68.93
le450_25b	450	8,263	25	10/10	1,849	65.82
le450_25c	450	17,343	25	0/10	-	-
le450_25d	450	17,425	25	0/10	-	-
le450_5a	450	5,714	5	10/10	3,071	82.47
le450_5b	450	5,734	5	10/10	8,885	238.33
le450_5c	450	9,803	5	10/10	3,212	86.68
le450_5d	450	9,757	5	10/10	1,644	44.49
mulsol.i.1	197	3,925	49	10/10	2,331	18.79
mulsol.i.2	188	3,885	31	10/10	8,696	63.18
mulsol.i.3	184	3,916	31	10/10	7,814	55.88
mulsol.i.4	185	3,946	31	10/10	8,584	60.71
mulsol.i.5	186	3,973	31	10/10	8,685	62.72
zeroin.i.1	211	4,100	49	10/10	3,014	27.1
zeroin.i.2	211	3,541	30	10/10	4,775	39.08
zeroin.i.3	206	3,540	30	10/10	4,286	34.51
anna	138	493	11	10/10	354	1.04
david	87	406	11	10/10	167	0.26
homer	561	1,628	13	10/10	1,222	59.01
huck	74	301	11	10/10	81	0.11
jean	80	254	10	10/10	98	0.13
games120	120	638	9	10/10	109	0.24
miles1000	128	3,216	42	10/10	570	2.43
miles1500	128	5,198	73	10/10	4,736	24.65
miles250	128	387	8	10/10	173	0.4
miles500	128	1,170	20	10/10	307	1.07
miles750	128	2,113	31	10/10	671	2.54
myciel3	11	20	4	10/10	7	0.0
myciel4	23	71	5	10/10	15	0.0
myciel5	47	236	6	10/10	41	0.03
myciel6	95	755	7	10/10	179	0.26
myciel7	191	2,360	8	9/10	377	1.52
mug88_1	88	146	4	10/10	43	0.05
mug88_25	88	146	4	10/10	46	0.05
mug100_1	100	166	4	10/10	54	0.07
mug100_25	100	166	4	10/10	47	0.06

Table 3: Summary of the results of the DR algorithm implemented with $T_{C_1, C_2, 0.75}$ for finding proper colorings of a representative sample of DIMACS benchmark instances. For each problem, we show the number of solved runs, the average time (in seconds) and the average number of iterations. We also include the number of nodes and edges, and the chromatic number of each graph. Runs were considered as unsolved after 3600 seconds

formulations. This is strictly an empirical observation, and we have no proposal on how solution multiplicity might be linked to limit cycle behavior. Our results are offered as motivation for pursuing this direction in future research on projection based algorithms.

References

[1] Achlioptas, D., Friedgut, E.: A sharp threshold for k-colorability. Random Struct. Alg. 14, 63–70 (1999)

[2] Achlioptas, D., Molloy, M.: Almost all graphs with $2.522n$ edges are not 3-colorable. Elec. Jour. Of Comb. 6(1), R29 (1999)

[3] Aragón Artacho, F.J., Borwein, J.M., Martín-Márquez, V., Yao, L.: Applications of convex analysis within mathematics. Math. Program. 148(1–2), Ser. B, 49–88 (2014)

[4] 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)

[5] 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)

[6] 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)

[7] Aragón Artacho, F.J., Campoy, R.: Solving graph coloring problems with the Douglas–Rachford algorithm, Set-Valued Var. Anal. 26(2), 277–304 (2018)

[8] 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)

[9] Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in Hilbert spaces, 2nd edn. Springer, Berlin (2017)

[10] 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)

[11] Bauschke, H.H., Noll, D.: On the local convergence of the Douglas–Rachford algorithm. Arch. Math. 102(6), 589–600 (2014)

[12] Benoist, J.: The Douglas–Rachford algorithm for the case of the sphere and the line. J. Global Optim. 63(2), 363–380 (2015)

[13] Cegielski, A.: Iterative methods for fixed point problems in Hilbert spaces. Lecture Notes in Mathematics, 2057. Springer, Heidelberg (2012)

[14] Chaitin, G.J.: Register allocation and spilling via graph coloring. SIGPLAN Not., 39(4), 66–74 (2004)

[15] Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. 91(2), Ser. A, 201–213 (2002)

[16] Elser, V., Rankenburg, I., Thibault, P.: Searching with iterated maps. Proc. Natl. Acad. Sci. 104(2), 418–423 (2007)

[17] Erdös, P., Rényi, A.: On random graphs I. Publ. Math. Debrecen 6, 290–297 (1959)

[18] 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)

[19] 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)

[20] Hale, W.K.: Frequency assignment: Theory and applications. Proceedings of the IEEE, 68(12), 1497–1514 (1980)

[21] 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)

[22] Horn, R.A., Johnson, C.R.: Matrix analysis. Second edition. Cambridge University Press, Cambridge (2013)

[23] 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)

[24] Jensen, T.R., Toft, B.: Graph coloring problems. John Wiley & Sons, New York (1995)

[25] Johansson, F: mpmath, version 1.0 (2017), http://mpmath.org

[26] Karger, D., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. Journal of the ACM (JACM) 45(2), 246–265 (1998)

[27] Karp, R.M.: Reducibility among combinatorial problems. In Complexity of Computer Computations, R. Miller and J. Thatcher, eds., Plenum Press, New York, 85–103 (1972)

[28] Leighton, F.T.: A graph coloring algorithm for large scheduling problems. J. Res. Nat. Bur. Standard 84(6), 489–506 (1979)

[29] Lewis, R.M.R.: A Guide to Graph Colouring: Algorithms and Applications. Springer International Publishing (2016)

[30] Pardalos, P.M., Mavridou, T., Xue, J.: The graph coloring problem: A bibliographic survey. In Handbook of combinatorial optimization, Springer US, 1077–1141 (1998)

[31] OEIS Foundation Inc.: The On-Line Encyclopedia of Integer Sequences (2018), https://oeis.org/A088202

[32] Parks, H.R., Wills, D.C.: An Elementary Calculation of the Dihedral Angle of the Regular N-Simplex. The American Mathematical Monthly 109(8) (2002), 756–758.

[33] Phan, H.M.: Linear convergence of the Douglas–Rachford method for two closed sets. Optim. 65(2), 369–385 (2016)

[34] Pierra, G.: Decomposition through formalization in a product space. Math. Program. 28, 96–115 (1984)

[35] Tam, M.K.: Regularity properties of non-negative sparsity sets. J. Math. Anal. Appl. 447(2), 758–777 (2017)