Serdica
Mathematical Journal
Сердика
Математическо списание

For further information on
Serdica Mathematical Journal
which is the new series of
Serdica Bulgaricae Mathematicae Publicationes
visit the website of the journal http://www.math.bas.bg/~serdica
or contact: Editorial Office
Serdica Mathematical Journal
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Telephone: (+359-2)9792818, FAX:(+359-2)971-36-49
e-mail: serdica@math.bas.bg

Serdica Math. J. **39** (2013), 313–330

Serdica
Mathematical Journal

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

RECENT RESULTS ON DOUGLAS–RACHFORD METHODS

Francisco J. Aragón Artacho, Jonathan M. Borwein, Matthew K. Tam

Communicated by N. Ribarska

Dedicated to Asen Dontchev on the occasion of his 65th birthday, and to Vladimir Veliov on the occasion of his 60th birthday.

Abstract. Recent positive experiences applying convex feasibility algorithms of Douglas–Rachford type to highly combinatorial and far from convex problems are described.

1. Introduction. Douglas–Rachford iterations, as defined in Section 2, are moderately well understood when applied to finding a point in the intersection of two convex sets. Over the past decade, they have proven very effective in some highly non-convex settings; even more surprisingly this is the case for some highly discrete problems [3]. In this paper we wish to advertise the use of Douglas–Rachford methods in such combinatorial settings. The remainder of the paper is organized as follows.

In Section 2, we recount what is proven in the convex setting. In Section 3 we review the normal way of handling a (large) finite number of sets in a product

2010 *Mathematics Subject Classification:* 90C27, 90C59, 47N10.

Key words: Douglas–Rachford, projections, reflections, combinatorial optimization, modelling, feasibility, satisfiability, Sudoku, Nonograms.

space. In Section 4, we reprise what is known in the non-convex setting. There is less theory but significant and often positive experience. In Section 5, we turn to a more detailed discussion of combinatorial applications before focusing, in Section 6, on solving *Sudoku puzzles*, and, in Section 7, on solving *Nonograms*. More detailed numerical experience in these two cases is given in [3]. It is worth noting that both are NP-complete as decision problems. We end the paper with various concluding remarks in Section 8.

2. Convex Douglas–Rachford methods.

We now review Douglas–Rachford methods applied to closed and convex sets.

2.1. The classic Douglas–Rachford method.

The classical Douglas–Rachford scheme was originally introduced in connection with partial differential equations arising in heat conduction [14], and convergence later proven as part of [22]. Given initial point, x_0, and subsets, A and B, of a Hilbert space, $\mathcal{H}$, the scheme iterates by repeatedly setting $x_{n+1} = T_{A,B}x_n$, where $T_{A,B}$ is the 2-*set Douglas–Rachford operator* defined by

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

here I denotes the *identity mapping*, and $R_A(x)$ denotes the *reflection* of a point $x \in \mathcal{H}$ in the set A. The reflection can be defined as

$$R_A(x) := 2P_A(x) - x,$$

where $P_A(x)$ is the *closest point projection* of the point x onto the set A, that is,

$$P_A(x) := \left\{ z \in A \colon \|x - z\| = \inf_{a \in A} \|x - a\| \right\}.$$

In general, the projection P_A is a set-valued mapping. If A is closed and convex, the projection is uniquely defined for every point in $\mathcal{H}$, thus yielding a single-valued mapping (see e.g. [12, Th. 4.5.1]).

In the literature, the Douglas–Rachford scheme is also known as "*reflect–reflect–average*" [10], and "*averaged alternating reflections (AAR)*" [8].

Applied to closed and convex sets, convergence is understood and can be explained by using the theory of (firmly) nonexpansive mappings.

Theorem 2.1 (Douglas–Rachford [14], Lions–Mercier [22])**.** *Let $A, B \subseteq \mathcal{H}$ be closed and convex with nonempty intersection. For any $x_0 \in \mathcal{H}$, set $x_{n+1} = T_{A,B}x_n$. Then (x_n) converges weakly to a point x such that $P_A x \in A \cap B$.*

In an analysis of *von Neumann's alternating projection method*, Bauschke and Borwein [5] introduced the concept of the *displacement vector*, v, and used the sets E and F to generalize $A \cap B$.

$$v := P_{\overline{B-A}}(0), \quad E := A \cap (B - v), \quad F := (A + v) \cap B.$$

Note, if $A \cap B \neq \emptyset$ then $E = F = A \cap B$.

The same framework was utilized by Bauschke, Combettes and Luke [8] to analyze the Douglas–Rachford method.

Theorem 2.2 (Convex Douglas–Rachford [8, Th. 3.13])**.** *Let $A, B \subseteq \mathcal{H}$ be closed and convex. For any $x_0 \in \mathcal{H}$, set $x_{n+1} = T_{A,B}x_n$. Then:*

(i) $x_{n+1} - x_n = P_B R_A x_n - P_A x_n \to v$ *and* $P_B P_A x_n - P_A x_n \to v$.

(ii) *If $A \cap B \neq \emptyset$ then (x_n) converges weakly to a point in*

$$\operatorname{Fix}(T_{A,B}) = (A \cap B) + N_{\overline{A-B}}(0);$$

otherwise, $\|x_n\| \to +\infty$.

(iii) *Exactly one of the following two alternatives holds.*

(a) *$E = \emptyset$, $\|P_A x_n\| \to +\infty$, and $\|P_B P_A x_n\| \to +\infty$.*

(b) *$E \neq \emptyset$, the sequences $(P_A x_n)$ and $(P_B P_A x_n)$ are bounded, and their weak cluster points belong to E and F, respectively; in fact, the weak cluster points of*

$$((P_A x_n, P_B R_A x_n)) \text{ and } ((P_A x_n, P_B P_A x_n))$$

are best approximation pairs relative to (A, B).

Here, $N_C(x) := \{u \in \mathcal{H} : \langle c - x, u\rangle \leq 0, \forall c \in C\}$ denotes the *normal cone* to a convex set $C \subset \mathcal{H}$ at a point $x \in C$.

2.2. The cyclic Douglas–Rachford method. There are many possible generalizations of the classic Douglas–Rachford iteration. Given three sets A, B, C and $x_0 \in \mathcal{H}$, an obvious candidate is the iteration defined by repeatedly setting $x_{n+1} := T_{A,B,C}x_n$ where

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

So far, application of the above scheme has been fruitless. Some of the difficulties are illustrated by considering an example involving three lines in $\mathbb{R}^2$ [3, Ex. 2.1].

Instead, Borwein and Tam [11] considered cyclic applications of 2-set Douglas–Rachford operators. Given N sets $C_1, C_2, \ldots, C_N$, and $x_0 \in \mathcal{H}$, their *cyclic Douglas–Rachford scheme* iterates by repeatedly setting $x_{n+1} := T_{[C_1,C_2,\ldots,C_N]}x_n$, where $T_{[C_1,C_2,\ldots,C_N]}$ denotes the *cyclic Douglas–Rachford operator* defined by

$$T_{[C_1,C_2,\ldots,C_N]} := T_{C_N,C_1}T_{C_{N-1},C_N}\ldots T_{C_2,C_3}T_{C_1,C_2}.$$

In the consistent case, the iterations behave analogously to the classical Douglas–Rachford scheme (cf. Theorem 2.2).

Theorem 2.3 (Cyclic Douglas–Rachford)**.** *Let $C_1, C_2, \ldots, C_N \subseteq \mathcal{H}$ be closed and convex sets with a nonempty intersection. For any $x_0 \in \mathcal{H}$, set $x_{n+1} = T_{[C_1\,C_2\,\ldots\,C_N]}x_n$. Then (x_n) converges weakly to a point x such that $P_{C_i}x = P_{C_j}x$, for all indices i, j. Moreover, $P_{C_j}x \in \bigcap_{i=1}^N C_i$, for each index j.*

If $N = 2$ and $C_1 \cap C_2 = \emptyset$ (the inconsistent case), unlike the classical Douglas–Rachford scheme, the iterates are not unbounded (cf. Theorem 2.2). Moreover, there is evidence to suggest that the scheme can be used to produce best approximation pairs relative to (C_1, C_2) whenever they exist.

3. Feasibility problems in the product space. Given $C_1, C_2, \ldots, C_N \subset \mathbb{R}^n$, the *feasibility problem*[1] asks:

$$\text{Find } x \in \bigcap_{i=1}^N C_i \subset \mathbb{R}^n. \tag{2}$$

A great many optimization and reconstruction problems, both continuous and combinatorial, can be cast within this framework.

Define two sets $C, D \subset (\mathbb{R}^n)^N$ by

$$C := \prod_{i=1}^N C_i, \quad D := \{(x, x, \ldots, x) \in (\mathbb{R}^n)^N : x \in \mathbb{R}^n\}.$$

While the set D, the *diagonal*, is always a closed subspace, the properties of C are largely inherited. For instance, when $C_1, C_2, \ldots, C_N$ are closed and convex, so is C. Consider, now, the equivalent feasibility problem:

$$\text{Find } \mathbf{x} \in C \cap D \subset (\mathbb{R}^n)^N. \tag{3}$$

[1] In this context, "feasibility" and "satisfiability" can be used interchangeably.

It is equivalent in the sense that

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

Moreover, knowing the projections onto $C_1, C_2, \ldots, C_N$, the projections onto C and D can be easily computed.

Proposition 3.1 (Product projections)**.** *Let $C = \prod_{i=1}^{N} C_i$. Then for any $\mathbf{x} \in (\mathbb{R}^n)^N$,*

$$P_D \mathbf{x} = \left(\frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \right)^N, \tag{4}$$

and if $P_{C_1}(\mathbf{x}_1), \ldots, P_{C_N}(\mathbf{x}_N)$ are nonempty, then

$$P_C \mathbf{x} = \prod_{i=1}^{N} P_{C_i}(\mathbf{x}_i). \tag{5}$$

P r o o f. See, for example, [3, Prop. 3.1]. □

Most projection algorithms can be applied to feasibility problems with any finite number of sets without significant modification. An exception is the Douglas–Rachford scheme. Before [11] it had only been successfully investigated for two sets: making the product formulation crucial for the Douglas–Rachford scheme.

4. Non-convex Douglas–Rachford methods.

There is little theory in the non-convex setting; but some useful beginnings:

4.1. Theoretical underpinnings.

As a prototype, Borwein and Sims [10] considered the Douglas–Rachford scheme for a Euclidean sphere and a line. More precisely, for the sets

$$S := \{x \in \mathbb{R}^n : \|x\| = 1\}, \quad L := \{\lambda a + \alpha b \in \mathbb{R}^n : \lambda \in \mathbb{R}\},$$

where, without loss, $\|a\| = \|b\| = 1, a \perp b, \alpha > 0$. We summarize their findings ($\alpha \in [0, 1]$ represents the consistent case, and $\alpha > 1$ the inconsistent).

Theorem 4.1. *Given $x_0 \in \mathbb{R}^n$ define $x_{n+1} := T_{S,L} x_n$. Then:*

1. *If* $0 < \alpha < 1$, (x_n) *is locally convergent at each of* $\pm\sqrt{1-\alpha^2}a + \alpha b$.
2. *If* $\alpha = 0$ *and* $x_0(1) > 0$, (x_n) *converges to* a.
3. *If* $\alpha = 1$ *and* $x_0(1) \neq 0$, (x_n) *converges to* $\hat{y}b$ *for some* $\hat{y} > 1$.
4. *If* $\alpha > 1$ *and* $x_0(1) \neq 0$, $\|x_n\| \to \infty$.

In $\mathbb{R}^2$ with $\alpha = 1/\sqrt{2}$, Aragón and Borwein gave an explicit region of convergence [1]. Restriction to $\alpha = 1/\sqrt{2}$ was made for notational simplicity.

Recently, Hesse and Luke [20] have utilized a relaxed local version of (firm) nonexpansiveness, to quantify how "close" to being (firmly) nonexpansive a mapping is. Within their framework, local convergence of the Douglas–Rachford scheme, if the first reflection is performed with respect to a subspace, were obtained provided a coercivity condition and appropriate regularity conditions hold. The order of reflection is reversed, so the results of Hesse and Luke do not directly overlap with that of Aragón, Borwein and Sims. This is not a substantive difference.

4.2. A summary of applications. We briefly list a variety of highly non-convex, primarily combinatorial, problems where some form of Douglas–Rachford algorithm has proven very fruitful.

1. *Protein folding* and *graph coloring* problems were first studied via Douglas–Rachford methods in by Elser and his colleagues in [15] and [16], respectively. Indeed Elser seems to have been the first to see the remarkable potential of the method for non-convex problems.

2. *Image retrieval* and *phase reconstruction* problems are analyzed in some detail in [6, 7]. The *bit retrieval* problem is considered in [16].

3. The *N-queens problem* which requests the placement of N queens on a $N \times N$ chessboard is studied and solved in [23].

4. *Boolean satisfiability* is treated in [16, 19]. Note that the three variable case, *3-SAT*, was the first problem to be shown *NP-complete* [18].

5. *TetraVex*, also known as *McMahon Squares* in honour of the great English combinatorialist, Percy MacMahon, is an edge-matching puzzle, whose NP-completeness is discussed in [25], was studied in [4]. Problems up to size 4×4 could be solved in an average of 200 iterations. There are $10^{2n(n+1)}$ base-10 $n \times n$ boards, with $n = 3$ being the most popular.

6. Solutions of (very large) *Sudoku puzzles* have been studied in [23, 16]. For a discussion of NP-completeness of determining solvability of Sudokus see [24]. The solution of Sudoku puzzles by Douglas–Rachford methods [3] is described in Section 6.

7. *Nonograms* [26, 27] are a more recent NP-complete Japanese puzzle whose solution by Douglas–Rachford methods is discussed in Section 7.

8. Many *matrix completion problems* [21] can be successfully solved by Douglas–Rachford methods [2]: convex problems include matrix completion problems with positive semi-definite matrices, doubly-stochastic matrices, and Euclidean distance matrices; non-convex problems include matrix completion problems with low-rank constraints (in particular, low-rank Euclidean distance problems—such as protein reconstruction from NMR data), and Hadamard, skew-Hadamard and circulant-Hadamard matrix problems.

5. Successful combinatorial applications. The key to successful application is two-fold. First, the iteration must converge—at least with high probability. Our experience is when that happens, random restarts in case of failure are very fruitful. As we shall show, often this depends on making good decisions about how to model the problem. Second, one must be able to compute the requisite projections in closed form—or to approximate them efficiently numerically. As we shall indicate this is frequently possible for significant problems.

When these two events obtain, we are in the pleasant position of being able to lift much of our experience as continuous optimizers to the combinatorial milieu.

5.1. Model formulation. Within the framework of feasibility problems, there can be numerous ways to model a given type of problem. The product space formulation (3) gives one, even without assuming any additional knowledge of the underlying problem.

The chosen formulation heavily influences the performance of projection algorithms. For example, in initial numerical experiments, the cyclic Douglas–Rachford scheme of Section 2.2, was directly applied to (2). As a serial algorithm, it appears to outperform the classic Douglas–Rachford scheme, which must be applied in the product space (3). For details see [11].

As an heuristic for problems involving one or more non-convex set, the sensitivity of the Douglas–Rachford method to the formulation used must be emphasized. In the (continuous) convex setting, the formulation ensures success

of the algorithm, while in the combinatorial setting, the formulation determines whether or not the algorithm can successfully and reliably solve the problem. Direct applications to feasibility problems with integer constraints have been largely unsuccessful. On the other hand, many of the successful applications listed in Section 4.2 use binary formulations.

We now outline the basic idea behind these reformations. If

$$x \in \{c_1, c_2, \ldots, c_n\} \subset \mathbb{R}, \tag{6}$$

we reformulate x as a vector $y \in \mathbb{R}^n$. If $x = c_i$, then y is defined by

$$y_j = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases}$$

With this interpretation (6) is equivalent to:

$$y \in \{e_1, e_2, \ldots, e_n\} \subset \mathbb{R}^n,$$

with $y = e_i$ if and only if $x = c_i$.

5.2. Projection onto the set of permutations of points. In many situations, in order to apply the Douglas–Rachford iteration, one needs to compute the projection of a point $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ onto the set of permutations of n given points $c_1, \ldots, c_n \in \mathbb{R}$, a set that will be denoted by $\mathcal{C}$. We will shall see below that this is the case for the Sudoku puzzle.

Fix $x \in \mathbb{R}^n$. Denote by $[\mathcal{C}]_x$ the set of vectors in $\mathcal{C}$ (which therefore have the same components but perhaps permuted) such that $y \in [\mathcal{C}]_x$ if the ith largest entry of y has the same index in y as the ith largest entry of x.

Proposition 5.1. *Denote by $\mathcal{C} \subset \mathbb{R}^n$ the set of vectors whose entries are all permutations of $c_1, c_2, \ldots, c_n \in \mathbb{R}$. Then for any $x \in \mathbb{R}^n$,*

$$P_{\mathcal{C}} x = [\mathcal{C}]_x.$$

P r o o f. See [3, Prop. 5.1]. □

6. Solving Sudoku puzzles. We illustrate the reformulation described in Section 5 with Sudoku, modelled first as an integer feasibility problem, and secondly as binary feasibility. One should acknowledge the fundamental contributions of Veit Elser [15, 16], who came up with the binary formulation of Sudoku as well as the 'nasty' Sudoku shown in Figure 2.

Denote by $A[i,j]$, the (i,j)-th entry of the matrix A. Denote by $A[i{:}i', j{:}j']$ the submatrix of A formed by taking rows i through i' and columns j through j' (inclusive). When i and i' are the indices of the first and last rows, we abbreviate by $A[:, j : j']$. We abbreviate similarly for the column indices. The *vectorization* of the matrix A by columns, is denoted by $\operatorname{vec} A$. For multidimensional arrays, the notation extends in the obvious way.

Let S denote the partially filled 9×9 integer matrix representing the incomplete Sudoku. For convenience, let $I = \{1, 2, \ldots, 9\}$ and let $J \subseteq I^2$ be the set of indices for which S is filled. Whilst we will formulate the problem for 9×9 Sudoku, we note that the same principles can be applied to larger Sudoku puzzles.

6.1. Sudoku modelled as integer program. Sudoku is modelled as an integer feasibility problem in the obvious way. Denote by C, the set of vectors which are permutations of $1, 2, \ldots, 9$. Let $A \in \mathbb{R}^{9\times 9}$. Then A is a completion of S if and only if

$$A \in C_1 \cap C_2 \cap C_3 \cap C_4,$$

where

$$\begin{aligned}
C_1 &= \{A : A[i,:] \in C \text{ for each } i \in I\},\\
C_2 &= \{A : A[:,j] \in C \text{ for each } j \in I\},\\
C_3 &= \{A : \operatorname{vec} A[3i+1 : 3(i+1), 3j+1 : 3(j+1)] \in C \text{ for } i,j = 0,1,2\},\\
C_4 &= \{A : A[i,j] = S[i,j] \text{ for each } (i,j) \in J\}.
\end{aligned}$$

The projections onto C_1, C_2, C_3 are given by Proposition 5.1, and can be efficiently computed by using the algorithm outlined in [3, Remark 5.2]. The projection onto C_4 is given, pointwise, by

$$(P_{C_4}A)[i,j] = \begin{cases} S[i,j] & \text{if } (i,j) \in J,\\ A[i,j] & \text{otherwise;} \end{cases}$$

for each $(i,j) \in I^2$.

6.2. Sudoku modelled as a zero-one program. Denote by C, the set of all n-dimensional standard basis vectors. To model Sudoku as a binary feasibility problem, we define $B \in \mathbb{R}^{9\times 9\times 9}$ by

$$B[i,j,k] = \begin{cases} 1 & \text{if } A[i,j] = k,\\ 0 & \text{otherwise.} \end{cases}$$

Let S' denote the partially filled $9 \times 9 \times 9$ zero-one array representing the incomplete Sudoku, S, under the reformulation, and let $J' \subseteq I^3$ be the set of indices for which S' is filled.

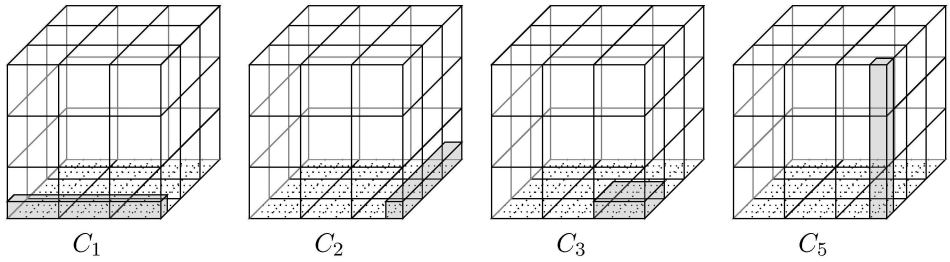


Fig. 1. Visualization of B showing the types of constraints encountered in Sudoku modelled as a zero-one program. The entries in the shaded "blocks" are all "0", except for a single "1" entry

The four constraints of the previous section become

$$\begin{aligned}
C_1 &= \{B : B[i,:,k] \in C \text{ for each } i,k \in I\},\\
C_2 &= \{B : B[:,j,k] \in C \text{ for each } j,k \in I\},\\
C_3 &= \{B : \operatorname{vec} B[3i+1:3(i+1),3j+1:3(j+1),k] \in C\\
&\qquad \text{for } i,j = 0,1,2 \text{ and } k \in I\},\\
C_4 &= \{B : B[i,j,k] = 1 \text{ for each } (i,j,k) \in J'\}.
\end{aligned}$$

In addition, since each Sudoku square has precisely one entry, we require

$$C_5 = \{B : B[i,j,:] \in C \text{ for each } i,j \in I\}.$$

A visualization of the constraints is provided in Figure 1. Clearly there is a one-to-one correspondence between completed integer Sudokus, and zero-one arrays contained in the intersection of the five constraint sets. Moreover, B is a completion of S' if and only if

$$B \in C_1 \cap C_2 \cap C_3 \cap C_4 \cap C_5.$$

The projections onto C_1, C_2, C_3, C_5 are again given by Proposition 5.1. The projection onto C_4 is given, pointwise, by

$$(P_{C_4}B)[i,j,k] = \begin{cases} S[i,j,k] & \text{if } (i,j,k) \in J',\\ B[i,j,k] & \text{otherwise;} \end{cases}$$

for each $(i,j,k) \in I^3$.

6.3. Models that succeeded. The Douglas–Rachford iteration applied to the binary formulation of Section 6.2 has been used to solve to Sudoku puzzles

fairly reliably. In [3], the present authors bench-marked the Douglas–Rachford implementation against three more specialized Sudoku solvers: a binary program solved using *Gurobi Optimizer*, an exact cover formulation solved using a *Dancing Links* implementation of Knuth's *Algorithm X*, and a hybrid method which combines a reasoning algorithm with a recursive search. These other solvers employ either back-tracking or branch-and-bound at some stage.

The above methods were compared on various libraries of 9×9 Sudoku problems including Gordon Royle's *minimum Sudoku*, and Dukuso's *top95* and *top1465*. These are frequently used by programmers to test their solvers. A number of larger 16×16 and 25×25 Sudoku puzzles were also generated for comparison, although only the performance of the binary program model could be compared. This is because, almost all available solvers are designed to handle only the standard 9×9 puzzle.

We summarize the results of the comparison. The binary program outperformed all the methods considered, regardless of the test library. The Gurobi optimizer is, of course, the most sophisticated in terms of its implementation and code optimization. The appeal of the Douglas–Rachford method is, in part, the simplicity of its implementation. It would be interesting to compare a more sophisticated implementation of the scheme with Gurobi, especially on larger problems. Excluding the binary program, our Douglas–Rachford implementation was at least competitive with methods examined, and for some test libraries it was better. Typically puzzles were solved within the first 2000 iterations, and for a given test library, the method successfully solved puzzles at worst 85% of the time. Typically the success rate was much better.

In his MSc thesis [23], Jason Schaad created a web-based Douglas–Rachford Sudoku solver.[2] Recently, Sudoku puzzles which, as set, would not be solved by Schaad's solver were found. One is given in Figure 2, which is based on an example due to Veit Elser [17], who found the first puzzle which could not be solved using Douglas–Rachford methods. This 'nasty' Sudoku has also proven hard for our implementation, having a success rate of only 20%. As a first step to explaining this difficulty, the 'nasty' Sudoku was compared to each of the puzzles obtained by removing any one of its entries. With the top-left "7" removed, the puzzle was just as difficult – a success rate of 24% which is comparable to the original puzzle. If any other entry was removed, the puzzle was relatively easy – a success rate of 99%.

When entries are removed from a Sudoku, the puzzle no longer necessarily has unique solution. To our surprise, the puzzle obtained by removing the

[2]Schaad's web-based solver: `https://people.ok.ubc.ca/bauschke/Jason/`

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

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

Fig. 2. The 'nasty' Sudoku (left), and its unique[3] solution (right)

top-left "7" had only five solutions, while the puzzles obtained by removing any other single entry had anywhere from a few hundred to a few thousand solutions. In comparison, the puzzles obtained by removing entries from Arto Inkala's *AI escargot* had at most a few hundred solutions. *AI escargot* is a Sudoku puzzle purposely designed to be really difficult, but nevertheless can be solved by the Douglas–Rachford with a success rate of 99%. Perhaps, applied to the 'nasty' Sudoku, the Douglas–Rachford algorithm is overwhelmed by an abundance of 'near' solutions.

6.4. Models that failed. To our consternation, the integer formulation of Section 6.1 was ineffective, except for 4×4 Sudoku, while the binary reformulation of the cyclic Douglas–Rachford method described in Section 2.2 also failed in both the original space and the product space.

Clearly we have a lot of work to do to understand the model characteristics which lead to success and those which lead to failure. We should also like to understand how to diagnose infeasibility in Sudoku via the binary model. This would give a full treatment of Sudoku as an NP-complete problem.

7. Solving nonograms. A nonogram puzzle consists of a blank $m \times n$ grid of pixels (the canvas) together with $(m + n)$ cluster-size sequences, one for

[3]The number of distinct solutions was determined using *SudokuSolver*, see `http://infohost.nmt.edu/tcc/help/lang/python/examples/sudoku/`

each row and each column [13]. The goal is to paint the canvas with a picture that satisfies the following constraints:

- Each pixel must be black or white.
- If a row (resp. column) has cluster-size sequence $s_1, s_2, \ldots, s_k$ then it must contain k clusters of black pixels, separated by at least one white pixel, while the ith leftmost (resp. uppermost) cluster contains s_i black pixels.

An example of a nonogram puzzle is given in Figure 3.

			5	1			5						1					
			1	1			3	1	1	1	5		1	3		2	2	
	2	1	1	1	1	1	2	2	9	4	1	1	7	4	7	4	5	
10	8	2	1	6	12	5	2	1	2	5	7	5	2	1	3	1	2	12

			1	2	2	2	2	1
1	1	1	1	1	1	1	1	1
					1	1	2	2
		1	1	1	1	3	4	1
		1	1	1	1	3	4	1
			2	2	3	2	1	2
2	1	1	1	1	1	1	1	2
			2	2	1	1	1	2
	2	1	1	1	1	1	2	3
			2	2	2	1	2	3
					2	10	3	1
			1	1	4	4	3	1
				1	3	3	1	1
				1	1	4	5	2
					1	1	11	3

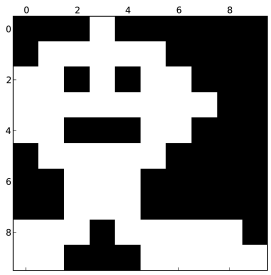
Fig. 3. A *special* nonogram. Cluster-size sequences for each row and column are shown. Its solution, found by the Douglas–Rachford algorithm, is shown in Figure 5

We model nonograms as binary feasibility problems. The $m \times n$ grid is represented as a matrix $A \in \mathbb{R}^{m \times n}$. We define

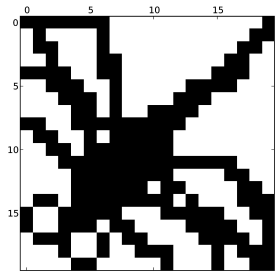
$$A[i,j] = \begin{cases} 0 & \text{if the } (i,j)\text{-th entry of the grid is white,} \\ 1 & \text{if the } (i,j)\text{-th entry of the grid is black.} \end{cases}$$

Let $\mathcal{R}_i \subset \mathbb{R}^m$ (resp. $\mathcal{C}_j \subset \mathbb{R}^n$) denote the set of vectors having cluster-size sequences matching row i (resp. column j).

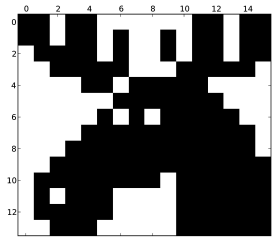
$$C_1 = \{A : A[i,:] \in \mathcal{R}_i \text{ for } i = 1, \ldots, m\},$$



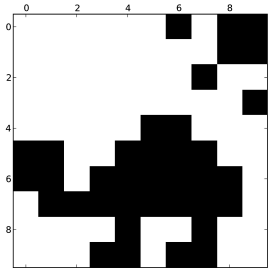
(a) A spaceman



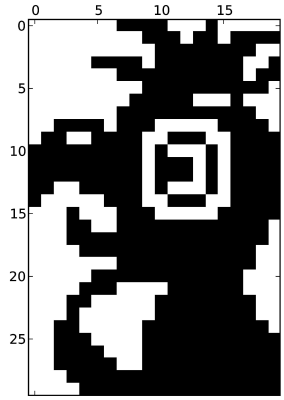
(b) A dragonfly



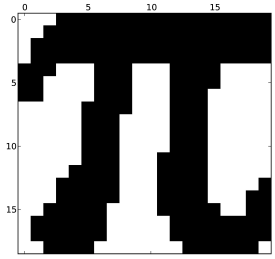
(c) A moose



(d) A turtle



(e) A parrot



(f) The number π

Fig. 4. Solutions to six nonograms found by Douglas–Rachford

$$C_2 = \{A : A[:, j] \in \mathcal{C}_j \text{ for } j = 1, \ldots, n\}.$$

Given an incomplete nonogram puzzle, A is a solution if and only if

$$A \in C_1 \cap C_2.$$

In [3], the present authors investigated viability of the Douglas–Rachford method applied to nonogram puzzles. Seven puzzles were examined: the puzzle in Figure 3, and the six puzzles shown in Figure 4. Appropriately modified, the

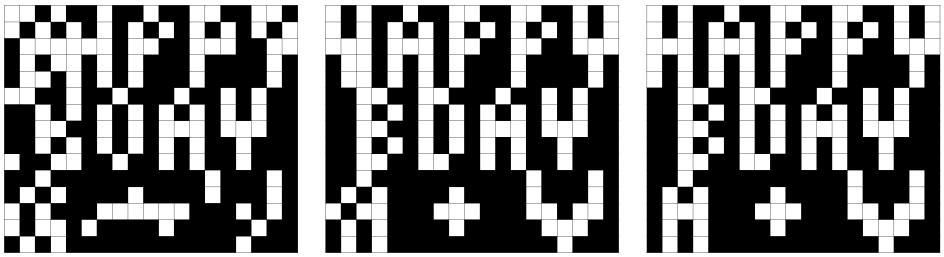


Fig. 5. Solution to the *special* nonogram from Figure 3 found by the Douglas–Rachford algorithm in only three iterations. We show here the projection onto C_1 of these three Douglas–Rachford iterations

method is the same as that of Section 6.3. This Douglas–Rachford application was highly successful.

From 1000 random initializations, all puzzles considered were solved with a 100% success rate. However, a difficulty within the above model, is that the projections onto C_1 and C_2 seem to have no simple form, unlike the Sudoku model in Section 6. So far, attempts to find an efficient method to compute them have been unsuccessful. The implementation described in [3] pre-computes $\mathcal{R}_i$ and $\mathcal{C}_j$, for all indices i, j, and at each iteration chooses the nearest point by computing the distance to each point in the appropriate set. While the Douglas–Rachford iterations themselves are fast, for very large nonograms, such enumeration of $\mathcal{R}_i$ and $\mathcal{C}_j$ becomes intractable.

8. Conclusion. The message of the list in Section 4.2 and of the previous two sections is the following. *When presented with a new combinatorial feasibility problem it is well worth seeing if Douglas–Rachford can deal with it—it is conceptually very simple and is usually relatively easy to implement.*

Finally, many resources can be found at the paper's companion website:
`http://carma.newcastle.edu.au/DRmethods/comb-opt/`

Acknowledgements. We wish to thank Heinz Bauschke, Russell Luke, Ian Searston and Brailey Sims for many useful insights.

REFERENCES

[1] F. J. Aragón Artacho, J. M. Borwein. Global convergence of a non-convex Douglas–Rachford iteration. *J. Global Optim.* **57**, *3* (2013), 753–769, DOI: 10.1007/s10898-012-9958-4.

[2] F. J. Aragón Artacho, J. M. Borwein, M. K. Tam. Douglas–Rachford feasibility methods for matrix completion problems. Preprint, http://arxiv.org/abs/1308.4243, August 2013.

[3] F. J. Aragón Artacho, J. M. Borwein, M. K. Tam. Recent results on Douglas–Rachford methods for combinatorial optimization problems. *J. Optim. Theory Appl.* (accepted for publication in November, 2013), DOI:10.1007/s10957-013-0488-0.

[4] P. Bansal. Code for solving Tetravex using Douglas–Rachford algorithm. http://people.ok.ubc.ca/bauschke/Pulkit/pulkitreport.pdf, 2010.

[5] H. H. Bauschke, J. M. Borwein. On the convergence of von Neumann's alternating projection algorithm for two sets. *Set-Valued Anal.* **1**, *2* (1993), 185–212.

[6] H. H. Bauschke, P. L. Combettes, D. R. Luke. Phase retrieval, error reduction algorithm, and Fienup variants: a view from convex optimization. *J. Opt. Soc. Amer. A* **19**, *7* (2002), 1334–1345.

[7] H. H. Bauschke, P. L. Combettes, D. R. Luke. Hybrid projection–reflection method for phase retrieval. *J. Opt. Soc. Amer. A* **20**, *6* (2003), 1025–1034.

[8] H. H Bauschke, P. L. Combettes, D. R. Luke. Finding best approximation pairs relative to two closed convex sets in Hilbert spaces. *J. Approx. Theory*, **127**, *2* (2004), 178–192.

[9] J. M. Borwein. Maximum entropy and feasibility methods for convex and nonconvex inverse problems. *Optimization* **61**, *1* (2012), 1–33. DOI: 10.1080/02331934.2011.632502.

[10] J. M. Borwein, B. Sims. The Douglas–Rachford algorithm in the absence of convexity. Fixed-Point Algorithms for Inverse Problems in Science and Engineering, 93–109, Springer Optim. Appl., vol. **49**. New York, Springer, 2011.

[11] J. M. Borwein, M. K. Tam. A cyclic Douglas–Rachford iteration scheme. *J. Optim. Theory Appl.* (accepted for publication August, 2013), DOI: 10.1007/s10957-013-0381-x.

[12] J. M. Borwein, Q. J. Zhu. Techniques of Variational Analysis. New York, Springer, 2005.

[13] R. A. Bosch. Painting by numbers. *Optima* **65** (2001), 16–17, `www.oberlin.edu/math/faculty/bosch/pbn.ps`.

[14] J. Douglas, H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.* **82**, *2* (1956), 421–439.

[15] V. Elser, I. Rankenburg. Deconstructing the energy landscape: Constraint-based algorithms for folding heteropolymers. *Physical Review E* **73**, *2* (2006), 026702.

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

[17] V. Elser. Personal communication, August 27th, 2012.

[18] R. Garey, D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. A Series of Books in the Mathematical Sciences. San Francisco, Calif., W. H. Freeman and Co, 1979.

[19] S. Gravel, V. Elser. Divide and concur: A general approach to constraint satisfaction. *Physical Review E* **78**, *3* (2008), 036706.

[20] R. Hesse, D. R. Luke. Nonconvex notions of regularity and convergence of fundamental algorithms for feasibility problems. Preprint, `http://arxiv.org/pdf/1205.0318v1`, 2012.

[21] C. R. Johnson. Matrix completion problems: A survey. Matrix theory and applications (Phoenix, Ariz., 1989), 171–198, Proc. Sympos. Appl. Math., vol. **40**, Providence, RI, Amer. Math. Soc., 1990.

[22] P. L. Lions, B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.* **16**, *6* (1979), 964–979.

[23] J. Schaad. Modeling the 8-queens problem and sudoku using an algorithm based on projections onto nonconvex sets. Master's thesis, Univ. of British Columbia, 2010.

[24] Y. Takayuki, S. Takahiro. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **E86**, *5* (2003), 1052–1060.

[25] Y. Takenaga, T. Walsh. Tetravex is NP-complete. *Information Processing Letters* **99** (2006), 171–174.

[26] N. Ueda, T. Nagao. NP-completeness results for nonogram via parsimonious reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institute of Technology, 2012. CiteSeerX: `10.1.1.57.5277`.

[27] J. N. van Rijn. Playing games: The complexity of Klondike, Mahjong, nonograms and animal chess. Master's thesis, Leiden Institute of Advanced Computer Science, Leiden University, 2012.

Francisco J. Aragón Artacho
e-mail: `francisco.aragon@ua.es`
Jonathan M. Borwein
e-mail: `jon.borwein@gmail.com`
Matthew K. Tam
e-mail: `matthew.k.tam@gmail.com`
CARMA, University of Newcastle
Callaghan, NSW 2308, Australia

Received May 10, 2013