Douglas–Rachford for Combinatorial Optimisation

Matthew K. Tam
Joint work with Dr. Fran Aragón and Laur. Prof. Jon Borwein

School of Mathematical and Physical Sciences
University of Newcastle, Australia

AMSSC, 15th–17th July 2013

With generous support from AustMS and AMSSC

In Sudoku the player fills entries of an incomplete Latin square subject to constraints. As a decision problem, it is NP-complete.

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

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

Some questions to ponder during this talk are:

- How can we solve large Sudoku puzzles? ($n^2 \times n^2$ instances)
- If any single entry is removed, how many distinct solutions can a puzzle have?
- Can such characteristics be used to better understand why algorithms work?

Introduction

The Douglas–Rachford method (a projection algorithm) was originally introduced in connection with PDEs arising in heat conduction. Applied to convex problems, the methods have a strong theoretical foundation, and its behaviour well understood.

I will discuss recent applications of the Douglas–Rachford method to a number of NP-complete combinatorial optimisation problems which are far from convex. Despite a lack of sufficient theoretical justification, the method performs quite satisfactorily.

Let $S \subseteq \mathbb{R}^n$. Recall, S is convex if

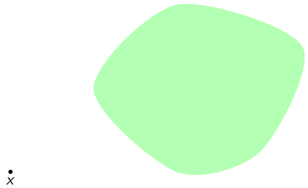
$$\lambda S + (1-\lambda)S \in S, \quad \forall \lambda \in [0,1].$$

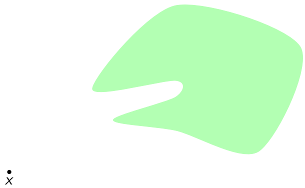
The (nearest point) projection onto S is the (set-valued) mapping,

$$P_S x := \operatorname*{argmin}_{s \in S} \|s - x\|.$$

The reflection w.r.t. S is the (set-valued) mapping,

$$R_S := 2P_S - I.$$





Let $S \subseteq \mathbb{R}^n$. Recall, S is convex if

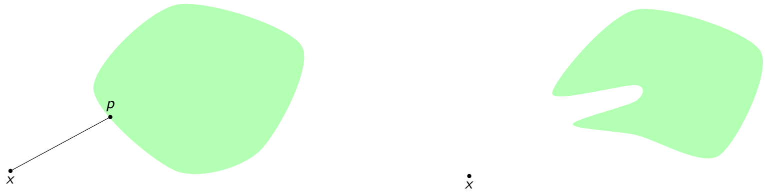
$$\lambda S + (1-\lambda)S \in S, \quad \forall \lambda \in [0,1].$$

The (nearest point) projection onto S is the (set-valued) mapping,

$$P_S x := \operatorname*{argmin}_{s \in S} \|s - x\|.$$

The reflection w.r.t. S is the (set-valued) mapping,

$$R_S := 2P_S - I.$$



A Variational Toolkit

Let $S \subseteq \mathbb{R}^n$. Recall, S is convex if

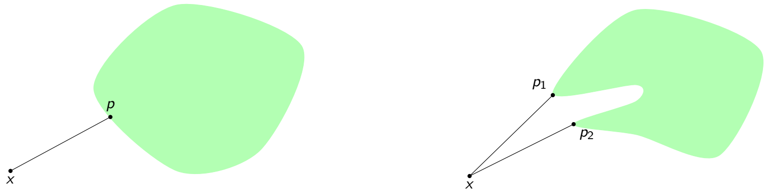
$$\lambda S + (1-\lambda)S \in S, \quad \forall \lambda \in [0,1].$$

The (nearest point) projection onto S is the (set-valued) mapping,

$$P_S x := \operatorname*{argmin}_{s \in S} \|s - x\|.$$

The reflection w.r.t. S is the (set-valued) mapping,

$$R_S := 2P_S - I.$$



Let $S \subseteq \mathbb{R}^n$. Recall, S is convex if

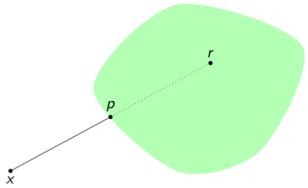
$$\lambda S + (1-\lambda)S \in S, \quad \forall \lambda \in [0,1].$$

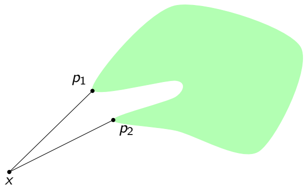
The (nearest point) projection onto S is the (set-valued) mapping,

$$P_S x := \operatorname*{argmin}_{s \in S} \|s - x\|.$$

The reflection w.r.t. S is the (set-valued) mapping,

$$R_S := 2P_S - I.$$





A Variational Toolkit

Let $S \subseteq \mathbb{R}^n$. Recall, S is convex if

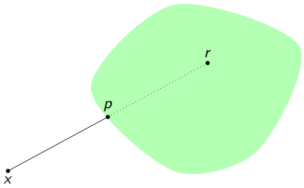
$$\lambda S + (1-\lambda)S \in S, \quad \forall \lambda \in [0,1].$$

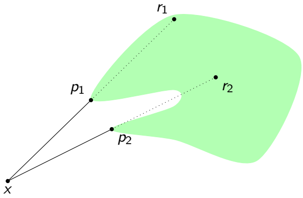
The (nearest point) projection onto S is the (set-valued) mapping,

$$P_S x := \operatorname*{argmin}_{s \in S} \|s - x\|.$$

The reflection w.r.t. S is the (set-valued) mapping,

$$R_S := 2P_S - I.$$





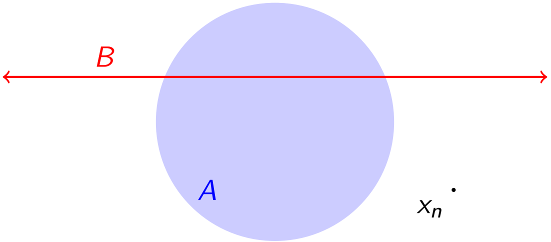
The Douglas–Rachford Scheme

Theorem (Douglas–Rachford, Lions–Mercier)

Suppose $A, B \subseteq \mathbb{R}^n$ are closed and convex with $A \cap B \neq \emptyset$. For any $x_0 \in \mathbb{R}^n$ define

$$x_{n+1} := Tx_n \text{ where } T := \frac{I + R_B R_A}{2}.$$

Then (x_n) converges to a point x such that $P_A x \in A \cap B$.



$$A = \{x \in \mathbb{R}^n : \|x\| \leq 1\}, \quad B = \{x \in \mathbb{R}^n : \langle a, x \rangle = b\}.$$

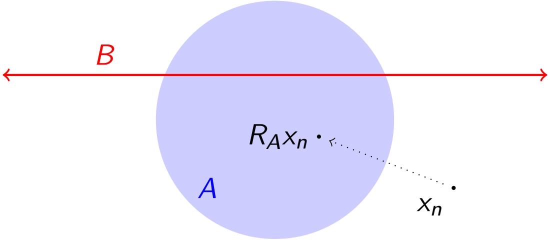
The Douglas–Rachford Scheme

Theorem (Douglas–Rachford, Lions–Mercier)

Suppose $A, B \subseteq \mathbb{R}^n$ are closed and convex with $A \cap B \neq \emptyset$. For any $x_0 \in \mathbb{R}^n$ define

$$x_{n+1} := Tx_n \text{ where } T := \frac{I + R_B R_A}{2}.$$

Then (x_n) converges to a point x such that $P_A x \in A \cap B$.



$$A = \{x \in \mathbb{R}^n : \|x\| \leq 1\}, \quad B = \{x \in \mathbb{R}^n : \langle a, x \rangle = b\}.$$

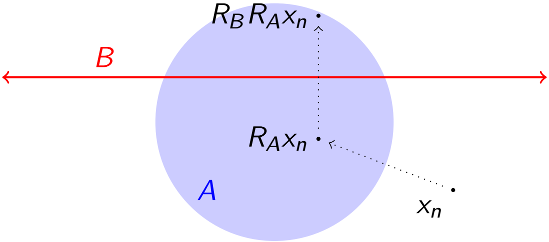
The Douglas–Rachford Scheme

Theorem (Douglas–Rachford, Lions–Mercier)

Suppose $A, B \subseteq \mathbb{R}^n$ are closed and convex with $A \cap B \neq \emptyset$. For any $x_0 \in \mathbb{R}^n$ define

$$x_{n+1} := Tx_n \text{ where } T := \frac{I + R_B R_A}{2}.$$

Then (x_n) converges to a point x such that $P_A x \in A \cap B$.



$$A = \{x \in \mathbb{R}^n : \|x\| \leq 1\}, \quad B = \{x \in \mathbb{R}^n : \langle a, x \rangle = b\}.$$

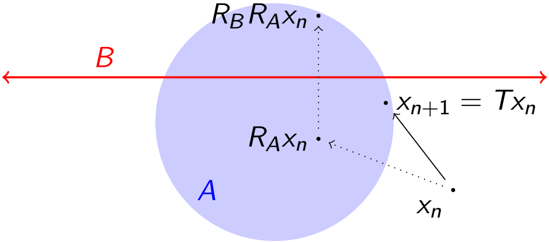
The Douglas–Rachford Scheme

Theorem (Douglas–Rachford, Lions–Mercier)

Suppose $A, B \subseteq \mathbb{R}^n$ are closed and convex with $A \cap B \neq \emptyset$. For any $x_0 \in \mathbb{R}^n$ define

$$x_{n+1} := Tx_n \text{ where } T := \frac{I + R_B R_A}{2}.$$

Then (x_n) converges to a point x such that $P_A x \in A \cap B$.



$$A = \{x \in \mathbb{R}^n : \|x\| \leq 1\}, \quad B = \{x \in \mathbb{R}^n : \langle a, x \rangle = b\}.$$

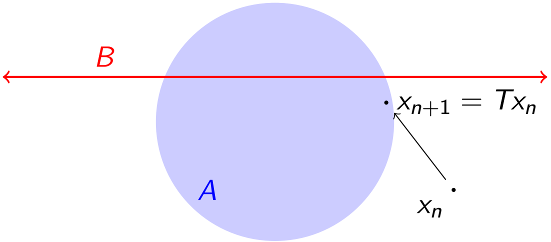
The Douglas–Rachford Scheme

Theorem (Douglas–Rachford, Lions–Mercier)

Suppose $A, B \subseteq \mathbb{R}^n$ are closed and convex with $A \cap B \neq \emptyset$. For any $x_0 \in \mathbb{R}^n$ define

$$x_{n+1} := Tx_n \text{ where } T := \frac{I + R_B R_A}{2}.$$

Then (x_n) converges to a point x such that $P_A x \in A \cap B$.



$$A = \{x \in \mathbb{R}^n : \|x\| \leq 1\}, \quad B = \{x \in \mathbb{R}^n : \langle a, x \rangle = b\}.$$

Modelling Sudoku: an NP-Complete Non-Convex Problem

Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times 9\times 9}$ by

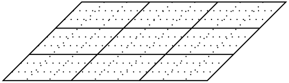
$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$

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

Modelling Sudoku: an NP-Complete Non-Convex Problem

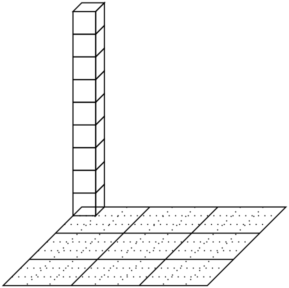
Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times9\times9}$ by

$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times 9\times 9}$ by

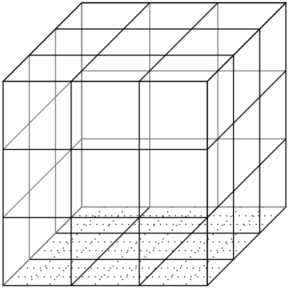
$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



Modelling Sudoku: an NP-Complete Non-Convex Problem

Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times 9\times 9}$ by

$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



The constraints are:

$C_1 = \{X : X_{ij} \in E\}$

$C_2 = \{X : X_{ik} \in E\}$

$C_3 = \{X : X_{jk} \in E\}$

$C_4 = \{X : \text{vec}(3 \times 3 \text{ submatrix}) \in E\}$

$C_5 = \{X : X \text{ matches original puzzle}\}$

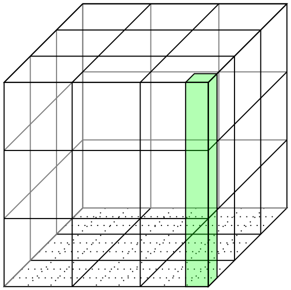
A solution is any

$$X \in \bigcap_{i=1}^{5} C_i.$$

Modelling Sudoku: an NP-Complete Non-Convex Problem

Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times 9\times 9}$ by

$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



The constraints are:

$C_1 = \{X : X_{ij} \in E\}$

$C_2 = \{X : X_{ik} \in E\}$

$C_3 = \{X : X_{jk} \in E\}$

$C_4 = \{X : \text{vec}(3 \times 3 \text{ submatrix}) \in E\}$

$C_5 = \{X : X \text{ matches original puzzle}\}$

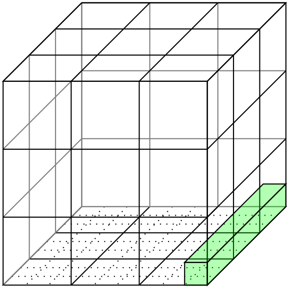
A solution is any

$$X \in \bigcap_{i=1}^{5} C_i.$$

Modelling Sudoku: an NP-Complete Non-Convex Problem

Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors. Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times 9\times 9}$ by

$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



The constraints are:

$C_1 = \{X : X_{ij} \in E\}$

$C_2 = \{X : X_{ik} \in E\}$

$C_3 = \{X : X_{jk} \in E\}$

$C_4 = \{X : \text{vec}(3 \times 3 \text{ submatrix}) \in E\}$

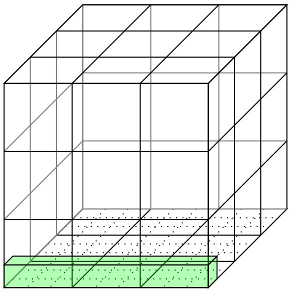
$C_5 = \{X : X \text{ matches original puzzle}\}$

A solution is any

$$X \in \bigcap_{i=1}^{5} C_i.$$

Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times 9\times 9}$ by

$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



The constraints are:

$C_1 = \{X : X_{ij} \in E\}$

$C_2 = \{X : X_{ik} \in E\}$

$C_3 = \{X : X_{jk} \in E\}$

$C_4 = \{X : \text{vec}(3 \times 3 \text{ submatrix}) \in E\}$

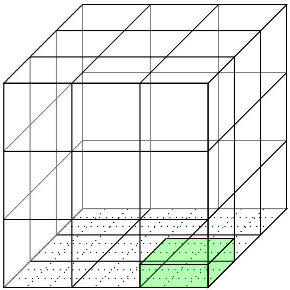
$C_5 = \{X : X \text{ matches original puzzle}\}$

A solution is any

$$X \in \bigcap_{i=1}^{5} C_i.$$

Let $E = \{e_j : j = 1, \ldots, 9\} \subset \mathbb{R}^9$ be the standard unit vectors.
Define the array $X = (X_{ijk}) \in \mathbb{R}^{9\times9\times9}$ by

$$X_{ijk} = \begin{cases} 1 & \text{if } ij\text{th entry is } k, \\ 0 & \text{otherwise.} \end{cases}$$



The constraints are:

$C_1 = \{X : X_{ij} \in E\}$

$C_2 = \{X : X_{ik} \in E\}$

$C_3 = \{X : X_{jk} \in E\}$

$C_4 = \{X : \text{vec}(3 \times 3 \text{ submatrix}) \in E\}$

$C_5 = \{X : X \text{ matches original puzzle}\}$

A solution is any

$$X \in \bigcap_{i=1}^{5} C_i.$$

Modelling Sudoku: an NP-Complete Non-Convex Problem

$P_{C_1}, P_{C_2}, P_{C_3}, P_{C_4}$ are simple to compute since, for any $x \in \mathbb{R}^9$,

$$P_E x = \{e_j : x_j = \max_{1 \le i \le 9} x_i\}.$$

P_{C_5} is also simple and given by setting $A_{ijk} = 1$ if the incomplete puzzle has a k in the ijth position.

Modelling Sudoku: an NP-Complete Non-Convex Problem

$P_{C_1}, P_{C_2}, P_{C_3}, P_{C_4}$ are simple to compute since, for any $x \in \mathbb{R}^9$,

$$P_E x = \{e_j : x_j = \max_{1 \leq i \leq 9} x_i\}.$$

P_{C_5} is also simple and given by setting $A_{ijk} = 1$ if the incomplete puzzle has a k in the ijth position.

Reformulate as a two set feasibility problem in the product space:

$$x \in \bigcap_{i=1}^{5} C_i \subseteq \mathbb{R}^{9\times 9\times 9} \iff (x, x, x, x, x) \in D \cap C \subseteq (\mathbb{R}^{9\times 9\times 9})^5,$$

where

$$D := \{(x, x, x, x, x) \in (\mathbb{R}^{9\times 9\times 9})^5 : x \in \mathbb{R}^{9\times 9\times 9}\}, \quad C := \prod_{i=1}^{5} C_i.$$

Computational Details

We tested the Douglas–Rachford method (C++) on various large suites of Sudoku puzzles. We give details of the implementation.

- *Initialise:* $\mathbf{x}_0 = (x_0, x_0, x_0, x_0, x_0)$ for random $x_0 \in [0,1]^{9\times9\times9}$.
- *Iterate:* By setting

$$\mathbf{x}_{n+1} = T\mathbf{x}_n = \frac{\mathbf{x}_n + R_C R_D \mathbf{x}_n}{2}.$$

- *Terminate:* Either, if a solution is found, or if 10000 iterations have been performed. Specifically, a solution is found if

$$P_D \mathbf{x}_n \in C \cap D.$$

Computational Details

We tested the Douglas–Rachford method (C++) on various large suites of Sudoku puzzles. We give details of the implementation.

- *Initialise:* $\mathbf{x}_0 = (x_0, x_0, x_0, x_0, x_0)$ for random $x_0 \in [0,1]^{9\times 9\times 9}$.
- *Iterate:* By setting

$$\mathbf{x}_{n+1} = T\mathbf{x}_n = \frac{\mathbf{x}_n + R_C R_D \mathbf{x}_n}{2}.$$

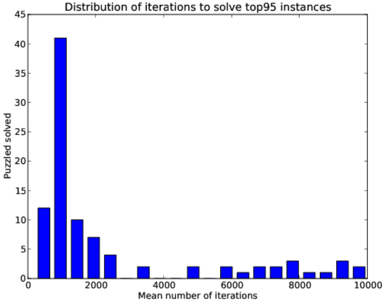
- *Terminate:* Either, if a solution is found, or if 10000 iterations have been performed. Specifically, a solution is found if

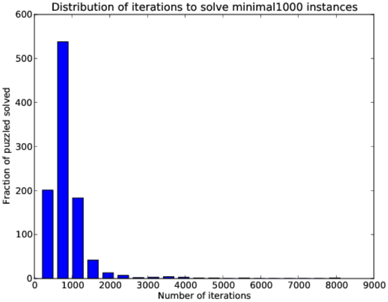
$$\mathrm{round}(P_D\mathbf{x}_n) \in C \cap D.$$

Computational Results: Success Rate

Table 1. % Solved by Test Library.

	top95	reglib-1.3	minimal1000	ksudoku16	ksudoku25
DR	86.53	99.35	99.59	92	100





Computational Example: A ‘Nasty’ Sudoku

This ‘nasty’ Sudoku cannot be solved reliably (20.2% success rate) by the Douglas–Rachford method.

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

Computational Example: A 'Nasty' Sudoku

This 'nasty' Sudoku cannot be solved reliably (20.2% success rate) by the Douglas–Rachford method.

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

Success rate when any single entry is removed:

- Top left 7 = 24%
- Any other entry = 99%

Computational Example: A ‘Nasty’ Sudoku

This ‘nasty’ Sudoku cannot be solved reliably (20.2% success rate) by the Douglas–Rachford method.

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

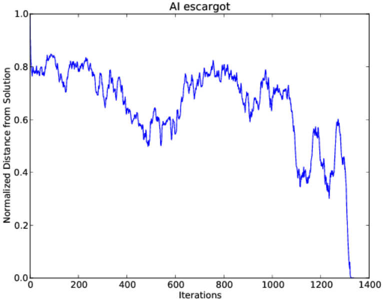
Success rate when any single entry is removed:

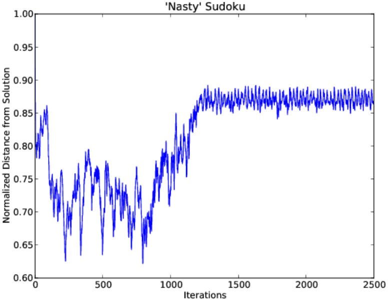
- Top left 7 $= 24\%$
- Any other entry $= 99\%$

Number of solutions when any single entry is removed:

- Top left 7 $= 5$
- Any other entry $= 200$–3800

Computational Results: Performance Comparison





Computational Results: Performance Comparison

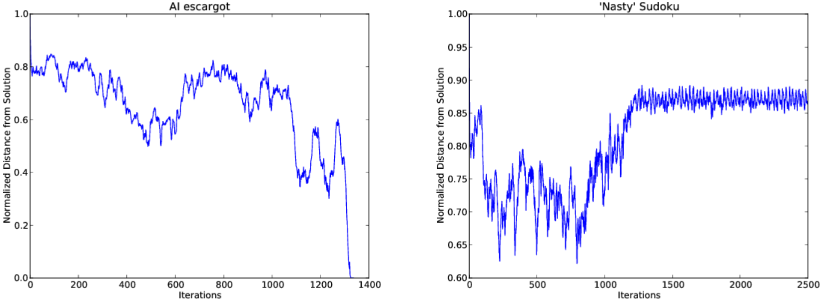


Table 2. Average Runtime (seconds).

	top95	reglib-1.3	minimal1000	ksudoku16	ksudoku25
DR	1.432	0.279	0.509	5.064	4.011
Gurobi	0.063	0.059	0.063	0.168	0.401
YASS	2.256	0.039	0.654	-	-
DLX	1.386	0.105	3.871	-	-

Concluding Remarks

When presented with a combinatorial feasibility problem it is well worth seeing if the Douglas–Rachford method can deal with it. It is conceptually simple, and easy to implement.

Other successful non-convex applications include:

- Boolean satisfiability, protein folding, graph colouring.
- TetraVex, generalised 8-queens problem.
- Nonograms – a Japanese number painting
- Matrix completion. e.g. low rank, various Hadamard matrices.
- Any suggestions?

Concluding Remarks

When presented with a combinatorial feasibility problem it is well worth seeing if the Douglas–Rachford method can deal with it. It is conceptually simple, and easy to implement.

Other successful non-convex applications include:

- Boolean satisfiability, protein folding, graph colouring.
- TetraVex, generalised 8-queens problem.
- Nonograms – a Japanese number painting
- Matrix completion. e.g. low rank, various Hadamard matrices.
- Any suggestions?

F.J. Aragón Artacho, J.M. Borwein & M.K. Tam. *Recent Results on Douglas–Rachford Methods for Combinatorial Optimization Problems*. Submitted, 2013.

Many resources can be found at the companion website:

`http://carma.newcastle.edu.au/DRmethods/comb-opt/`

The 'Nasty' Suduoku and its Unique Solution

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