Variational Analysis and Optimisation Webinar

Oct 13, 2021

A product space reformulation with reduced dimension

Rubén Campoy

Department of Statistics and Operational Research

Universitat de València

CONTENTS

CONTENTS

CONTENTS

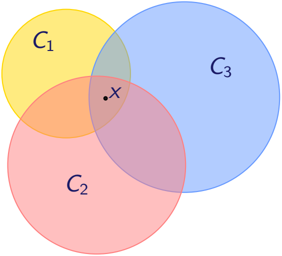
Let $\mathcal{H}$ be a Hilbert space with inner product $\langle\cdot,\cdot\rangle$ and induced norm $\|\cdot\|$.

$\rightharpoonup$: weak convergence $\rightarrow$: strong convergence

Consider $C_1, C_2, \ldots, C_r \subseteq \mathcal{H}$.

Feasibility Problem

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



- In many practical situations, finding a point in the intersection of the sets might be intricate.
- However, the projection onto each of these sets can be easily computed.
- In such cases, and when the sets are convex, the so-called projection algorithms are useful tools to solve the problem.

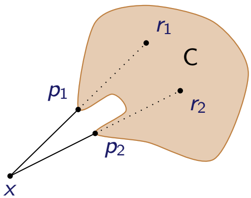
Projection mapping

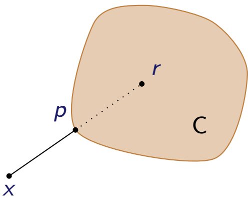
Let $C \subseteq \mathcal{H}$ be a closed nonempty set.

- The projector onto C is the (possibly set-valued) mapping

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

- The reflector with respect to C is the mapping $R_C := 2P_C - I$.





When C is closed and convex, P_C and R_C are single-valued.

FUNDAMENTAL PROJECTION ALGORITHMS

Alternating Projections (AP)

1933 Von Neumann

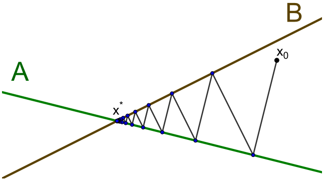
AP for two subspaces

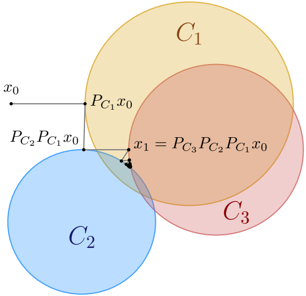
1962 Halperin

Generalization for any finite number of subspaces

1965 Bregman

Extension for arbitrary closed and convex sets





FUNDAMENTAL PROJECTION ALGORITHMS

Alternating Projections (AP)

1933 Von Neumann

AP for two subspaces

1962 Halperin

Generalization for any finite number of subspaces

1965 Bregman

Extension for arbitrary closed and convex sets

Douglas–Rachford (DR)

1956 Douglas and Rachford

Originally proposed for solving a system of linear equations arising in heat conduction problems.

1979 Lions and Mercier

Extension of the algorithm for convex feasibility problems
(In fact, for monotone inclusions)

The Douglas–Rachford algorithm

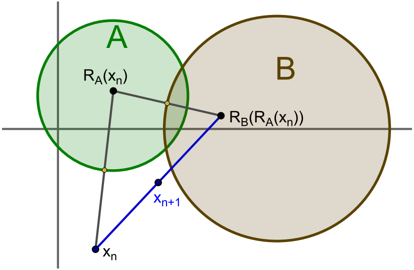
Definition (Douglas–Rachford operator)

Given two sets $A, B \subseteq \mathcal{H}$, the Douglas–Rachford operator is defined by

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

- The DR algorithm is the fixed point iteration $x_{n+1} = DR_{A,B}(x_n)$.
- Also known as Averaged Alternating Reflections method:

REFLECT
⇓
REFLECT
⇓
AVERAGE



- Can be generalized to $DR_{A,B,\lambda} = (1-\lambda)I + \lambda R_B R_A$, for $\lambda \in]0,1[$.

Convergence of Douglas–Rachford

$$x_{n+1} = DR_{A,B,\lambda}(x_n) := (1-\lambda)x_n + \lambda(2P_B - I)(2P_A - I)(x_n)$$

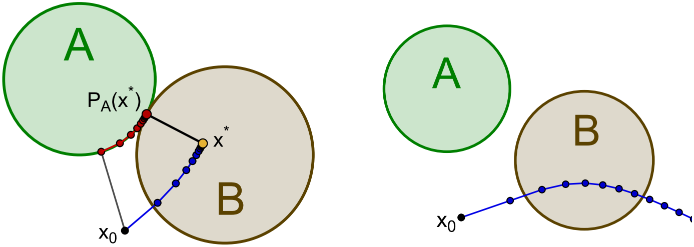
Theorem [Lions and Mercier (1979), Svaiter (2011)]

Let $A, B \subseteq \mathcal{H}$ be closed and convex sets. Given any $x_0 \in \mathcal{H}$, for every $n \geq 0$, define $x_{n+1} = DR_{A,B,\lambda}(x_n)$. Then, the following holds.

(i) If $A \cap B \neq \emptyset$, then $\{x_n\} \rightharpoonup x^\star \in \operatorname{Fix} DR_{A,B,\alpha}$ such that $P_A(x^\star) \in A \cap B$.

Moreover, the shadow sequence $\{P_A(x_n)\} \overset{w}{\rightharpoonup} P_A(x^\star) \in A \cap B$.

(ii) If $A \cap B = \emptyset$, then $\|x_n\| \to +\infty$.



CONTENTS

Framework

Let $\mathcal{H}$ be a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and induced norm $\|\cdot\|$.

$\rightharpoonup$: weak convergence $\to$: strong convergence

Monotone inclusion

Find $x \in \mathcal{H}$ such that $0 \in A(x) + B(x)$,

where $A, B : \mathcal{H} \rightrightarrows \mathcal{H}$ are maximally monotone operators.

Definition: A set-valued operator $A : \mathcal{H} \rightrightarrows \mathcal{H}$ is said to be

- monotone if

$$\langle x - y, u - v \rangle \geq 0, \quad \text{for all } (x, u), (y, v) \in \operatorname{gra} A;$$

- maximally monotone if it is monotone and there exists no other monotone operator $\widetilde{A} : \mathcal{H} \rightrightarrows \mathcal{H}$ such that $\operatorname{gra} A \subsetneq \operatorname{gra} \widetilde{A}$.

Examples of maximally monotone operators

- The **subdifferential** of a proper lsc convex funcion:

$$\partial f(x) := \{u \in \mathcal{H} \mid \langle y - x, u\rangle + f(x) \leq f(y), \quad \forall y \in \mathcal{H}\}.$$

Monotone inclusion

Find $\bar{x}$ s.t. $0 \in \partial f(\bar{x}) + \partial g(\bar{x})$ $\Leftrightarrow$

*Under constraint qualification

Minimization problem

$$\begin{array}{ll} \text{Min} & f(x) + g(x) \\ \text{s.a.} & x \in \mathcal{H}. \end{array}$$

- The **normal cone** to a closed and convex set:

$$N_C(x) := \begin{cases} \{u \in \mathcal{H} \mid \langle u, c - x\rangle \leq 0, \quad \forall c \in C\}, & \text{if } x \in C; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Monotone inclusion

Find $\bar{x}$ s.t. $0 \in N_A(\bar{x}) + N_B(\bar{x})$ $\Leftrightarrow$

Feasibility problem

Find $\bar{x} \in A \cap B$

The Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}$ and $\gamma > 0$, the Douglas–Rachford iteration is defined by:

$$x_{n+1} = \frac{1}{2}x_n + \frac{1}{2}(2J_{\gamma B} - I)(2J_{\gamma A} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Definition: Given a set-valued operator $A : \mathcal{H} \rightrightarrows \mathcal{H}$

- the resolvent of A with parameter $\gamma > 0$ is the operator

$$J_{\gamma A} := (\mathrm{Id} + \gamma A)^{-1}.$$

- the reflected resolvent is $R_{\gamma A} := 2J_{\gamma A} - \mathrm{Id}$.

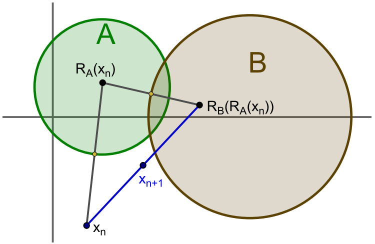
Douglas–Rachford for minimization and feasibility problems

- The resolvent of the **subdifferential** of a proper lsc convex funcion becomes the **proximity mapping**

$$J_{\gamma\partial f} = \operatorname{prox}_{\gamma f}(x) := \operatorname*{argmin}_{u\in\mathcal{H}} \left(f(u) + \frac{1}{2\gamma}\|x-u\|^2 \right).$$

- The resolvent of the **normal cone** to a closed and convex set becomes the **projector**

$$J_{\gamma N_C} = P_C(x) := \operatorname*{argmin}_{c\in C} \|x-c\|,$$



Convergence of Douglas–Rachford

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma B} - I)(2J_{\gamma A} - I)(x_n)$$

Theorem [Lions and Mercier (1979), Svaiter (2011)]

Let $A, B : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone operators such that $\operatorname{zer}(A+B) \neq \emptyset$. Let $\gamma > 0$ and let $\lambda \in]0,1[$. Given any $x_0 \in \mathcal{H}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda R_{\gamma B} R_{\gamma A}(x_n), \quad \text{for } n = 0, 1, 2, \ldots;$$

where $$R_{\gamma A} := 2J_{\gamma A} - I \quad \text{and} \quad R_{\gamma B} := 2J_{\gamma B} - I.$$

Then there exists $x^\star \in \operatorname{Fix}(R_{\gamma B} R_{\gamma A})$ such that following assertions hold:

(i) $\{x_n\} \rightharpoonup x^\star$ with $J_{\gamma A}(x^\star) \in \operatorname{zer}(A+B)$.

(ii) $\{J_{\gamma A}(x_n)\} \rightharpoonup J_{\gamma A}(x^\star) \in \operatorname{zer}(A+B)$.

SPLITTING ALGORITHMS

Those algorithms that solve the monotone inclusion

$$\text{Find } x \in \mathcal{H} \quad \text{such that} \quad 0 \in A(x) + B(x),$$

by taking advantage of the decomposition.

Their iteration is described by:

- Direct evaluations of A or B (forward-steps)
- Computations of the resolvents J_A and/or J_B (backward-steps)

Douglas–Rachford

Forward-Backward

ADMM

SPLITTING ALGORITHMS

Those algorithms that solve the monotone inclusion

$$\text{Find } x \in \mathcal{H} \quad \text{such that} \quad 0 \in A_1(x) + A_2(x) + \cdots + A_r(x),$$

by taking advantage of the decomposition.

Their iteration is described by:

- Direct evaluations of A_i (forward-steps)
- Computations of the resolvent J_{A_i} (backward-steps)

What if we deal with more than two operators?

Let's study first the case of feasibility problems with more than two sets

FUNDAMENTAL PROJECTION ALGORITHMS

Alternating Projections (AP)

1933 Von Neumann

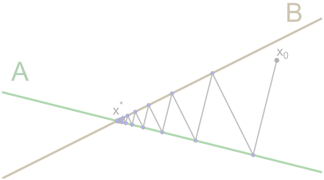
AP for two subspaces

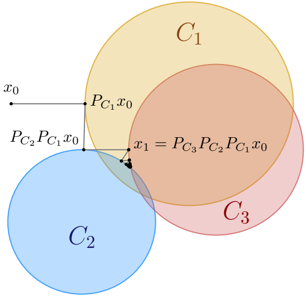
1962 Halperin

Generalization for any finite number of subspaces

1965 Bregman

Extension for arbitrary closed and convex sets





The Douglas–Rachford algorithm

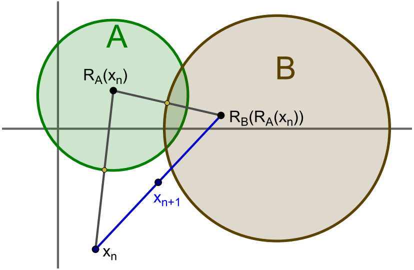
Definition (Douglas–Rachford operator)

Given two sets $A, B \subseteq \mathcal{H}$, the Douglas–Rachford operator is defined by

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

- The DR algorithm is the fixed point iteration $x_{n+1} = DR_{A,B}(x_n)$.
- Also known as Averaged Alternating Reflections method:

REFLECT
⇓
REFLECT
⇓
AVERAGE



- Can be generalized to $DR_{A,B,\lambda} = (1-\lambda)I + \lambda R_B R_A$, for $\lambda \in]0,1[$.

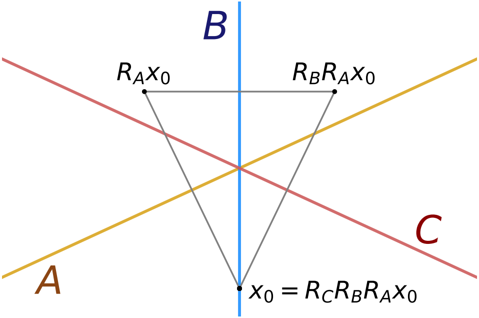
Douglas–Rachford for 3 sets

$$DR_{A,B,C} := \frac{\mathrm{Id} + R_C R_B R_A}{2}$$

- The iteration generated by the above operator still converges

$$x_n \rightharpoonup x^\star \in \operatorname{Fix} DR_{A,B,C}$$

- However the reached fixed point may not lead to a solution.



CONTENTS

CONTENTS

Product space reformulation

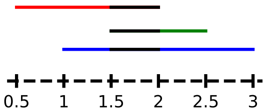
- Finitely many sets $C_1, C_2, \ldots, C_r \subseteq \mathcal{H}$, can be handled by a product space formulation.
- We work on the product Hilbert space $\mathcal{H}^r := \mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$.
- Define $\boldsymbol{C} := C_1 \times C_2 \times \cdots \times C_r$ and $\boldsymbol{D}_r := \{(x, x, \ldots, x) \in \mathcal{H}^r : x \in \mathcal{H}\}$.
- We now have an equivalent two-set feasibility problem since

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

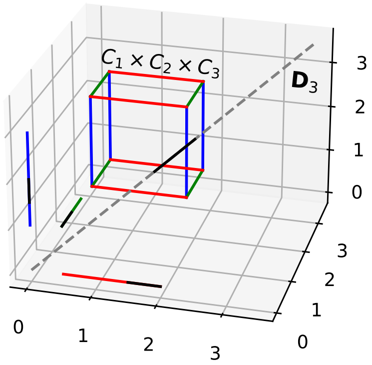
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



Standard product space reformulation



Product space reformulation

- Finitely many sets $C_1, C_2, \ldots, C_r \subseteq \mathcal{H}$, can be handled by a product space formulation.
- We work on the product Hilbert space $\mathcal{H}^r := \mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$.
- Define $\boldsymbol{C} := C_1 \times C_2 \times \cdots \times C_r$ and $\boldsymbol{D}_r := \{(x, x, \ldots, x) \in \mathcal{H}^r : x \in \mathcal{H}\}$.
- We now have an equivalent two-set feasibility problem since

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

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

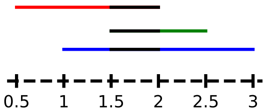
$$P_{\boldsymbol{C}}(\boldsymbol{x}) = \left(P_{C_1}(x_1),\ P_{C_2}(x_2),\ \ldots,\ P_{C_r}(x_r)\right),$$

$$P_{\boldsymbol{D}_r}(\boldsymbol{x}) = \left(\frac{1}{r}\sum_{i=1}^{r} x_i, \frac{1}{r}\sum_{i=1}^{r} x_i, \cdots, \frac{1}{r}\sum_{i=1}^{r} x_i\right).$$

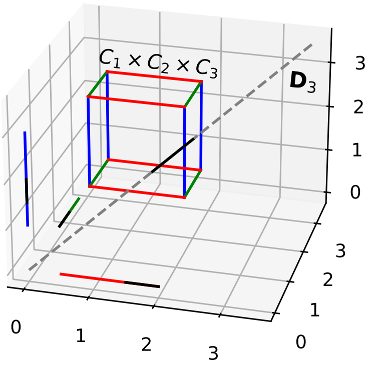
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



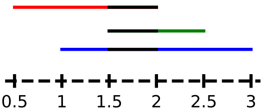
Standard product space reformulation



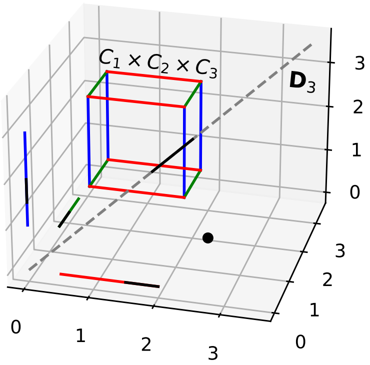
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



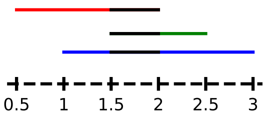
Standard product space reformulation



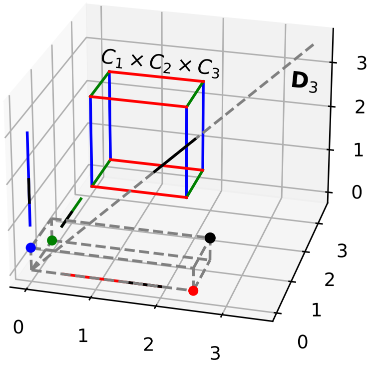
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



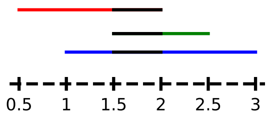
Standard product space reformulation



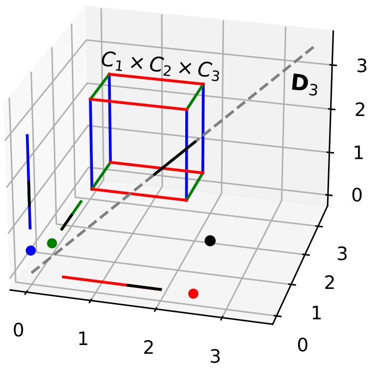
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],\quad C_2 := [1.5, 2.5],\quad C_3 := [1, 3].$$



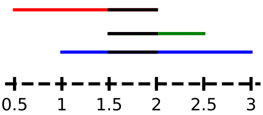
Standard product space reformulation



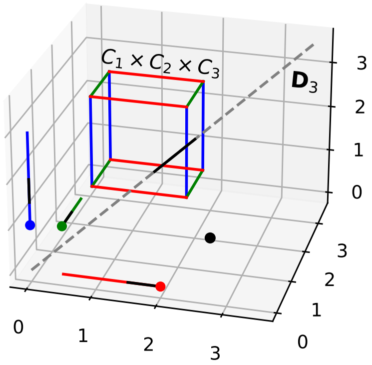
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



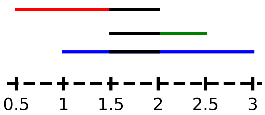
Standard product space reformulation



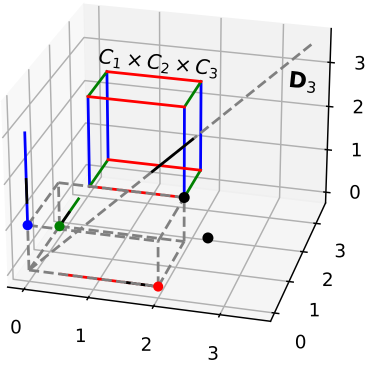
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



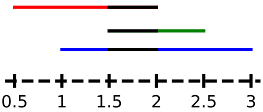
Standard product space reformulation



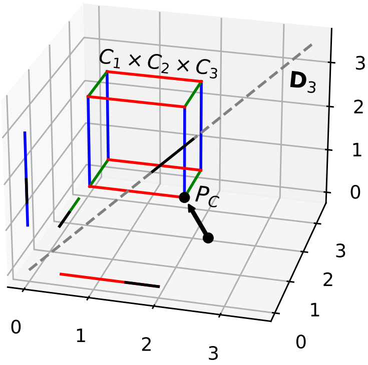
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



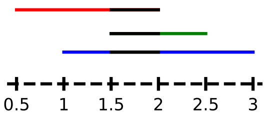
Standard product space reformulation



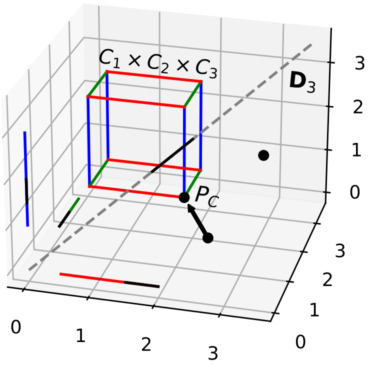
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



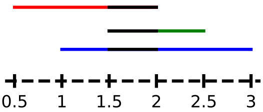
Standard product space reformulation



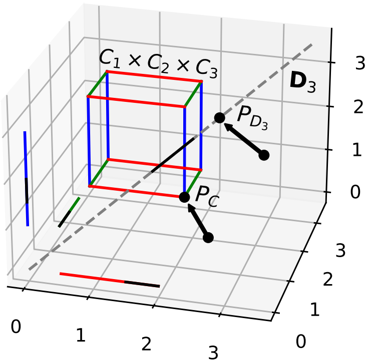
Product space reformulation

Example:

Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



Standard product space reformulation



Product space reformulation

- The product space trick is commonly known as *Pierra's product space reformulation*, credited to Guy Pierra in the paper:

  Pierra, G.: Decomposition through formalization in a product space. *Math. Program.* 28(1), 96–115 (1984)

- The reformulation was indepedently employed in earlier papers such as:

  Kruger, A. Y., Mordukhovich, B. S.: Generalized normals and derivatives and necessary conditions for an extremum in problems of nondifferentiable programming II. *VINITI*, no. 494-80 (1980)

  Kruger, A. Y.: Generalized differentials of nonsmooth functions. *VINITI*, no. 1332-81 (1981)

  Spingarn, J. E.: Partial inverse of a monotone operator. *Appl. Math. Optim.* 10(1), 247–265 (1983)

- It seems it first appeared in Pierra's thesis:

  Pierra, G.: Méthodes de décomposition et croisement d'algorithmes pour des problèmes d'optimisation. Doctoral dissertation, Institut National Polytechnique de Grenoble-INPG; Université Joseph-Fourier-Grenoble I, 1976.

Product space reformulation for splitting algorithms

Find $x \in \mathcal{H}$ such that $0 \in A_1(x) + A_2(x) + \cdots + A_r(x)$,

with $A_1, A_2, \ldots, A_r : \mathcal{H} \rightrightarrows \mathcal{H}$ maximally monotone.

Define the operator $\boldsymbol{A} : \mathcal{H}^r \rightrightarrows \mathcal{H}^r$ as

$$\boldsymbol{A}(\boldsymbol{x}) := A_1(x_1) \times A_2(x_2) \times \cdots \times A_r(x_r), \quad \forall \boldsymbol{x} = (x_1, x_2, \ldots, x_r) \in \mathcal{H}^r.$$

Proposition (Standard product space reformulation)

1. $\boldsymbol{A}$ is maximally monotone and

$$J_{\gamma \boldsymbol{A}}(\boldsymbol{x}) = \left(J_{\gamma A_1}(x_1), J_{\gamma A_2}(x_2), \cdots, J_{\gamma A_r}(x_r)\right), \quad \forall \boldsymbol{x} = (x_1, x_2, \ldots, x_r) \in \mathcal{H}^r.$$

2. The normal cone to $\boldsymbol{D}_r$, $N_{\boldsymbol{D}_r}$, is a maximally monotone operator and

$$J_{\gamma N_{\boldsymbol{D}_r}}(\boldsymbol{x}) = P_{\boldsymbol{D}_r}(\boldsymbol{x}) = \boldsymbol{j}_r\left(\frac{1}{r}\sum_{i=1}^{r} x_i\right), \quad \forall \boldsymbol{x} = (x_1, x_2, \ldots, x_r) \in \mathcal{H}^r.$$

3. $\operatorname{zer}(\boldsymbol{A} + N_{\boldsymbol{D}_r}) = \boldsymbol{j}_r\left(\operatorname{zer}\left(\sum_{i=1}^r A_i\right)\right)$.

$\boldsymbol{j}_r : \mathcal{H} \to \boldsymbol{D}_r : x \mapsto (x, x, \ldots, x)$

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots .$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k},$

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$

$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

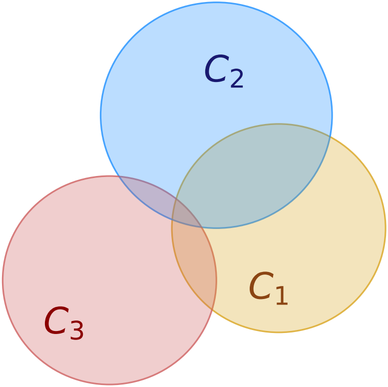
$$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

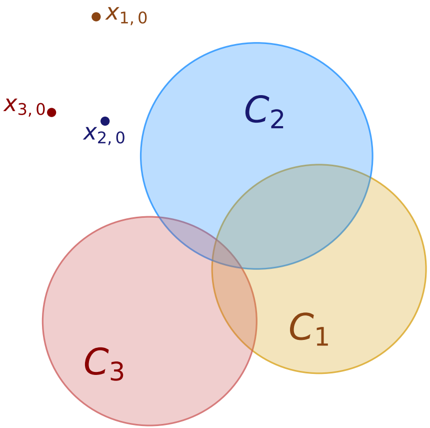
$p_k = \frac{1}{r}\sum_{i=1}^r x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right)$,

$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right)$.

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

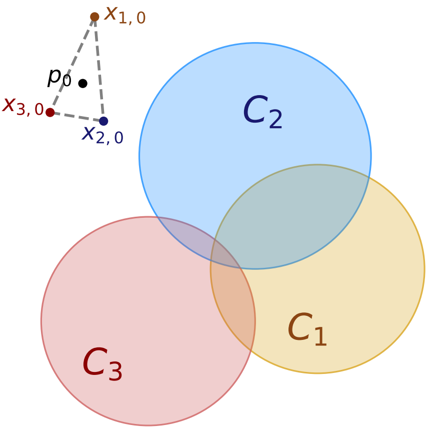
$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right)$,

$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right)$.

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

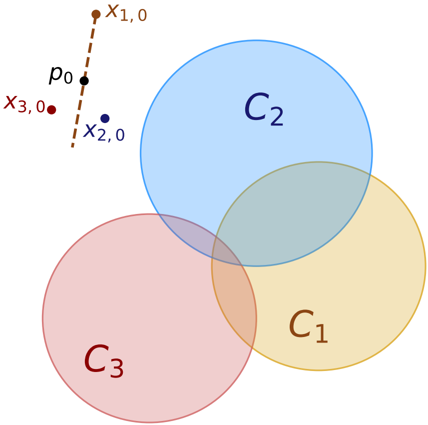
$$p_k = \frac{1}{r}\sum_{i=1}^r x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

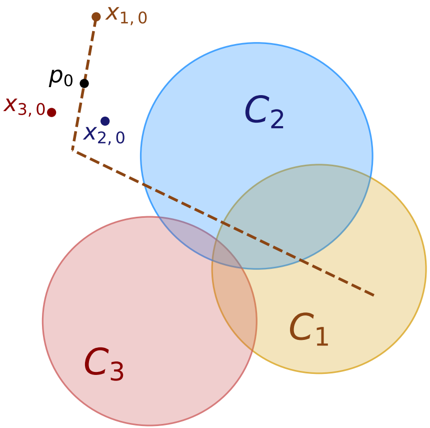
$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k})$,

$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k)$.

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

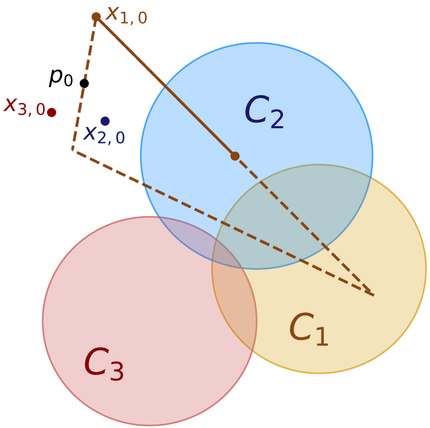
$$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots .$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

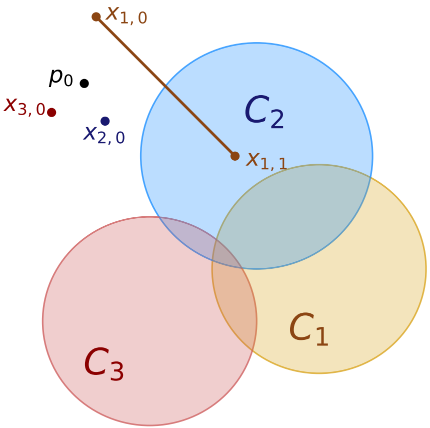
$$p_k = \frac{1}{r}\sum_{i=1}^r x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

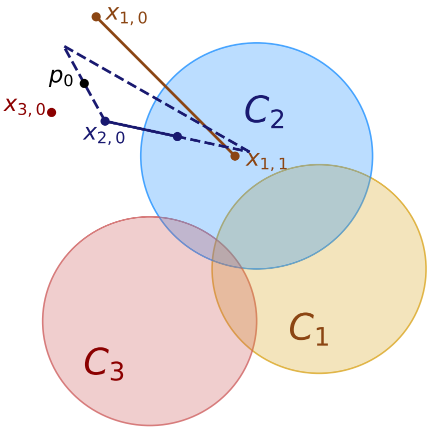
$p_k = \frac{1}{r}\sum_{i=1}^r x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k})$,

$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k)$.

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

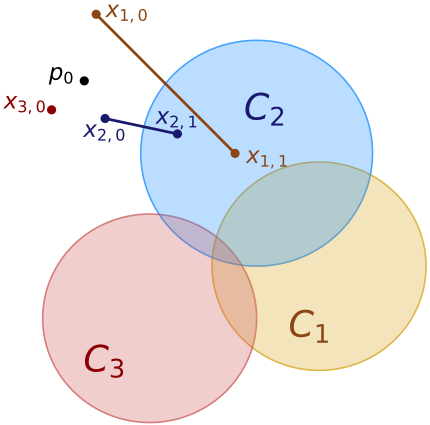
$$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

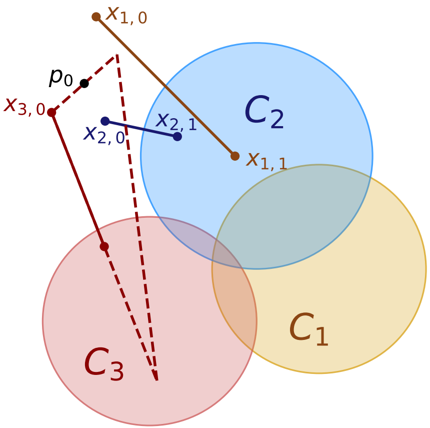
$$p_k = \frac{1}{r}\sum_{i=1}^r x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

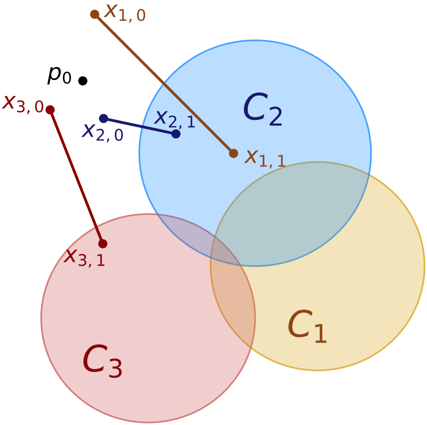
$$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \mathbf{A}} - I)(2J_{\gamma N_{\mathbf{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

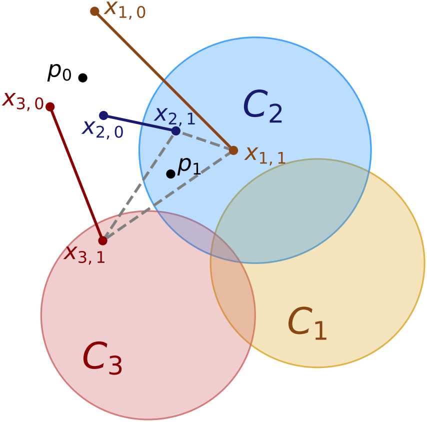
$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k})$,

$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k)$.

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

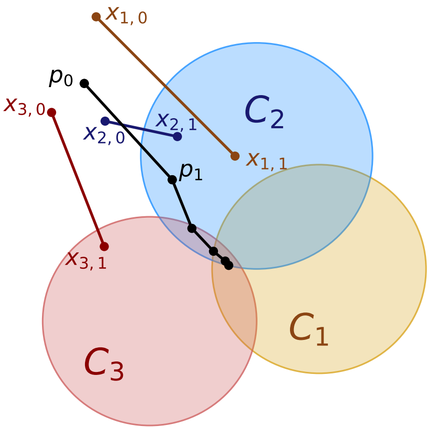
$$p_k = \frac{1}{r}\sum_{i=1}^r x_{i,k},$$

for $i = 1, 2, \ldots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \mathbf{A}} - I)(2J_{\gamma N_{\mathbf{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

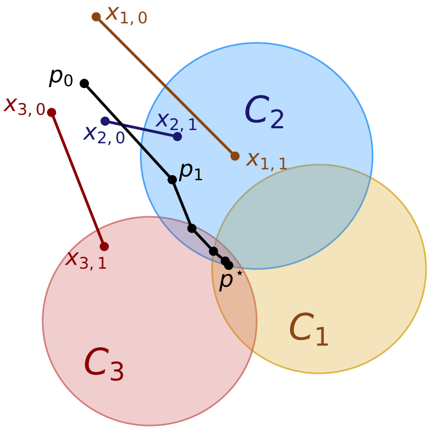
$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right)$,

$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right)$.

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^r$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{A}} - I)(2J_{\gamma N_{\boldsymbol{D}_r}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$p_k = \frac{1}{r} \sum_{i=1}^{r} x_{i,k}$,

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right)$,

$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right)$.

← Parallel algorithm

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

We need to work simultaneously with r sequences

This has been recently called **r-fold lifting**

Douglas–Rachford with minimal lifting

- Consider a monotone inclusion described by **two operators**:
  - DR on the product space reformulation: 2-fold lifting
  - DR on the original problem: no lifting (1-fold lifting) $\leftarrow$ Minimal
- Consider now the case of **three operators**:
  - DR on the product space reformulation: 3-fold lifting $\leftarrow$ Minimal?

Ryu, E. K.: Uniqueness of DRS as the 2 operator resolvent-splitting and impossibility of 3 operator resolvent-splitting. *Math. Program.* 182(1), 233–273 (2020)

$\hookrightarrow$ Imposibility of 1-fold lifting + Minimal lifting: 2-fold

- Can it be generalized for an arbitrary family of **r operators**?
  - DR on the product space reformulation: r-fold lifting $\leftarrow$ Minimal?
  - Minimal lifting: $(r-1)$-fold $\leftarrow$

Malitsky, Y., Tam, M. K.: Resolvent Splitting for Sums of Monotone Operators with Minimal Lifting. *ArXiv Prerprint* (2021)

CONTENTS

New product space reformulation with reduced dimension

Find $x \in \operatorname{zer}\{A_1(x)+A_2(x)+\cdots+A_r(x)\}$, $\quad A_1, A_2, \ldots, A_r : \mathcal{H} \rightrightarrows \mathcal{H}$ maximally monotone.

Consider the operators $\boldsymbol{B}, \boldsymbol{K} : \mathcal{H}^{r-1} \rightrightarrows \mathcal{H}^{r-1}$ defined, at $\boldsymbol{x} = (x_1, \ldots, x_{r-1}) \in \mathcal{H}^{r-1}$, by

$$\boldsymbol{B}(\boldsymbol{x}) := A_1(x_1) \times \cdots \times A_{r-1}(x_{r-1})$$

$$\boldsymbol{K}(\boldsymbol{x}) := \underbrace{\frac{1}{r-1}\left(A_r(x_1) \times \cdots \times A_r(x_{r-1})\right)}_{\boldsymbol{S}(\boldsymbol{x})} + N_{\boldsymbol{D}_{r-1}}(\boldsymbol{x}).$$

Theorem (Product space reformulation with reduced dimension)

1. $\boldsymbol{B}$ is maximally monotone and

$$J_{\gamma \boldsymbol{B}}(\boldsymbol{x}) = \left(J_{\gamma A_1}(x_1), \ldots, J_{\gamma A_{r-1}}(x_{r-1})\right), \quad \forall \boldsymbol{x} = (x_1, \ldots, x_{r-1}) \in \mathcal{H}^{r-1}.$$

2. $\boldsymbol{K} = \boldsymbol{S} + N_{\boldsymbol{D}_{r-1}}$ is maximally monotone and

$$J_{\gamma \boldsymbol{K}}(\boldsymbol{x}) = J_{\gamma(\boldsymbol{S}+N_{\boldsymbol{D}_{r-1}})}(\boldsymbol{x}) = J_{\gamma \boldsymbol{S}}\left(J_{\gamma N_{\boldsymbol{D}_{r-1}}}(\boldsymbol{x})\right) = \boldsymbol{j}_{r-1}\left(J_{\frac{\gamma}{r-1}A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_i\right)\right).$$

3. $\operatorname{zer}(\boldsymbol{B}+\boldsymbol{K}) = \boldsymbol{j}_{r-1}\left(\operatorname{zer}\left(\sum_{i=1}^{r} A_i\right)\right)$.

New product space reformulation for feasibility problems

- We can again tackle a feasibility problem described by $C_1, C_2, \ldots, C_r \subseteq \mathcal{H}$.
- We now work on the product Hilbert space $\mathcal{H}^{r-1} := \mathcal{H} \overset{(r-1)}{\times \cdots \times} \mathcal{H}$.
- Define the sets

$$\begin{aligned} \boldsymbol{B} &:= C_1 \times \cdots \times C_{r-1} \subseteq \mathcal{H}^{r-1}, \\ \boldsymbol{K} &:= (C_r \overset{(r-1)}{\times \cdots \times} C_r) \cap \boldsymbol{D}_{r-1} \subseteq \mathcal{H}^{r-1}. \end{aligned}$$

- We still have an equivalent two-set feasibility problem since

$$x \in \bigcap_{i=1}^{r} C_i \quad \Leftrightarrow \quad (x, \overset{(r-1)}{\ldots}, x) \in \boldsymbol{B} \cap \boldsymbol{K}.$$

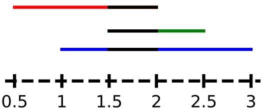
- Moreover, knowing the projections onto $C_1, \ldots, C_r$, the projections onto $\boldsymbol{B}$ and $\boldsymbol{K}$ can be easily computed. Indeed, for any $\boldsymbol{x} = (x_1, \ldots, x_r) \in \mathcal{H}^r$,

$$\begin{aligned} P_{\boldsymbol{B}}(\boldsymbol{x}) &= \left(P_{C_1}(x_1), \ \ldots, \ P_{C_{r-1}}(x_{r-1})\right), \\ P_{\boldsymbol{K}}(\boldsymbol{x}) &= \left(P_{C_r}\left(\frac{1}{r}\sum_{i=1}^{r} x_i\right), \overset{(r-1)}{\ldots}, P_{C_r}\left(\frac{1}{r}\sum_{i=1}^{r} x_i\right)\right). \end{aligned}$$

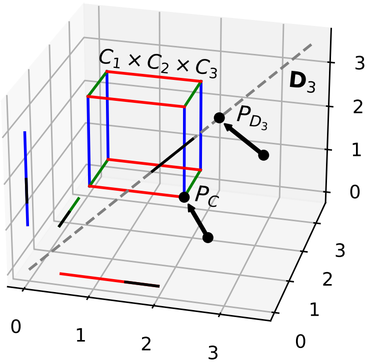
Product space reformulation

Example:

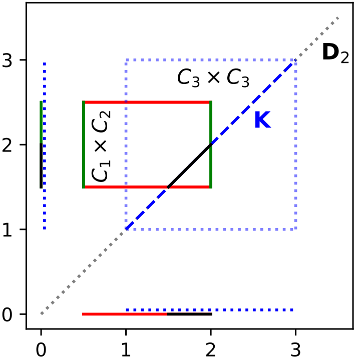
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



Standard product space reformulation



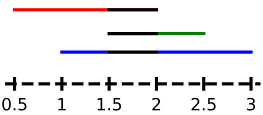
Product space reformulation with reduced dimension



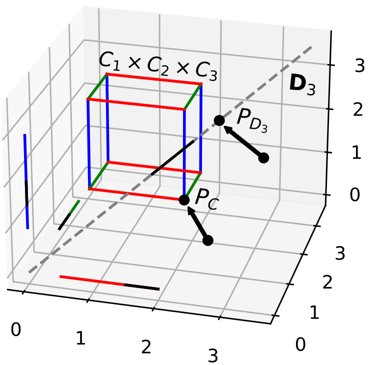
Product space reformulation

Example:

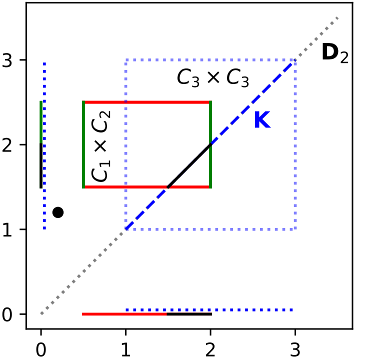
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



Standard product space reformulation



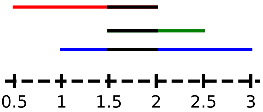
Product space reformulation with reduced dimension



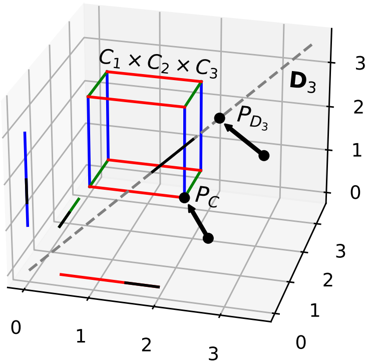
Product space reformulation

Example:

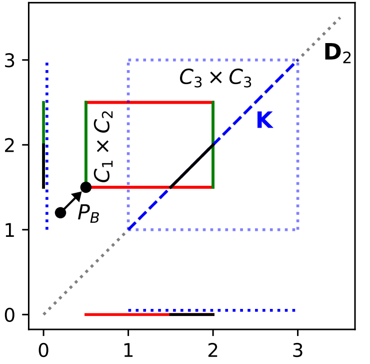
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $C_3 := [1, 3]$.



Standard product space reformulation



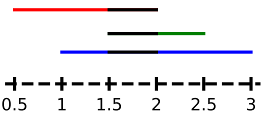
Product space reformulation with reduced dimension



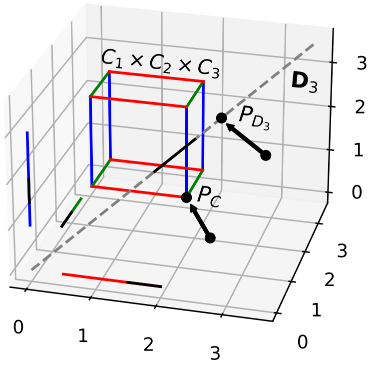
Product space reformulation

Example:

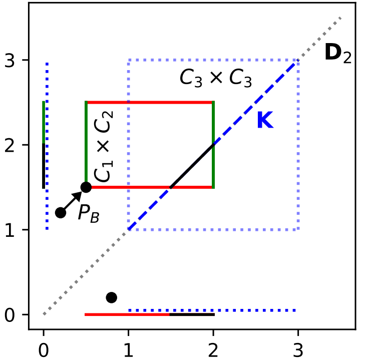
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



Standard product space reformulation



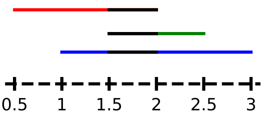
Product space reformulation with reduced dimension



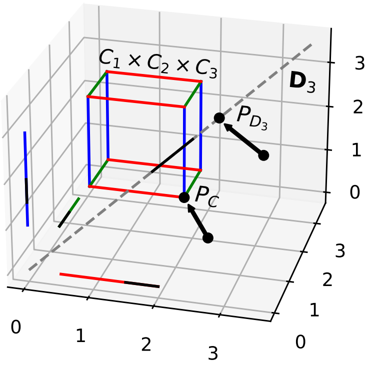
Product space reformulation

Example:

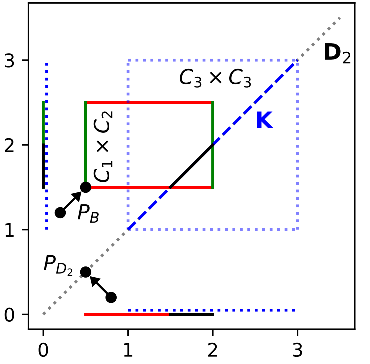
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



Standard product space reformulation



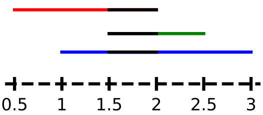
Product space reformulation
with reduced dimension



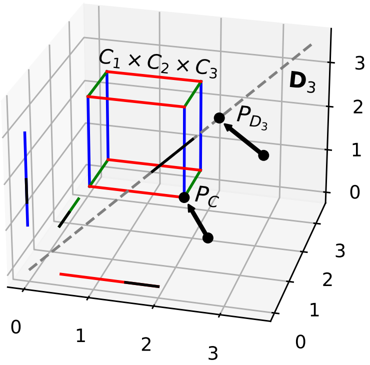
Product space reformulation

Example:

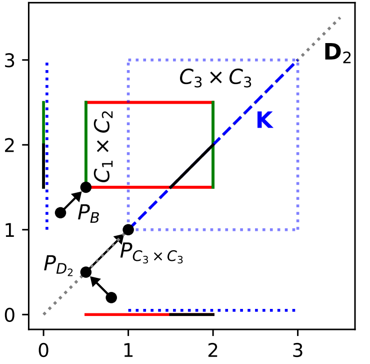
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],\quad C_2 := [1.5, 2.5],\quad C_3 := [1, 3].$$



Standard product space reformulation



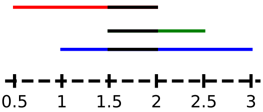
Product space reformulation with reduced dimension



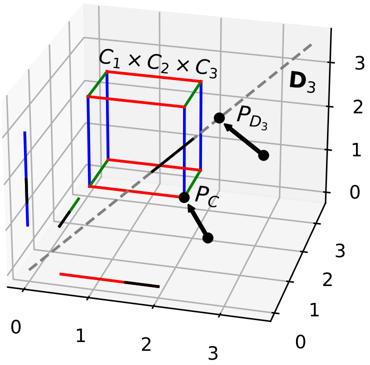
Product space reformulation

Example:

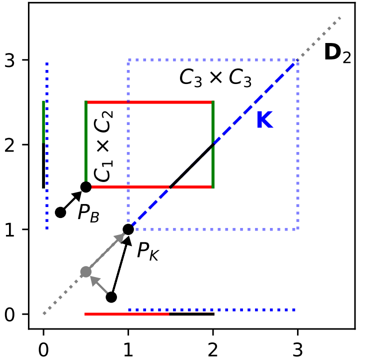
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],$$
$$C_2 := [1.5, 2.5],$$
$$C_3 := [1, 3].$$



Standard product space reformulation



Product space reformulation with reduced dimension



Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$

Standard product space reformulation

Given $x_{1,0}, x_{2,0}, \ldots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$p_k = \frac{1}{r} \sum_{i=1}^{r} x_{i,k},$

for $i = 1, 2, \ldots, r$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$

$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1} \sum_{i=1}^{r-1} x_{i,k}\right),$

for $i = 1, 2, \ldots, r-1$:

$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$

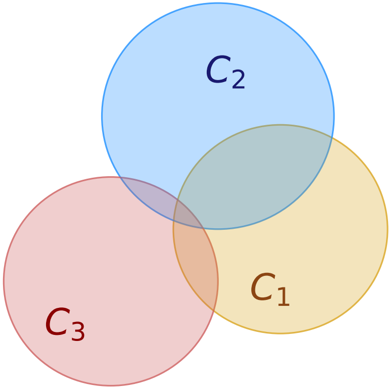
$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1}A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

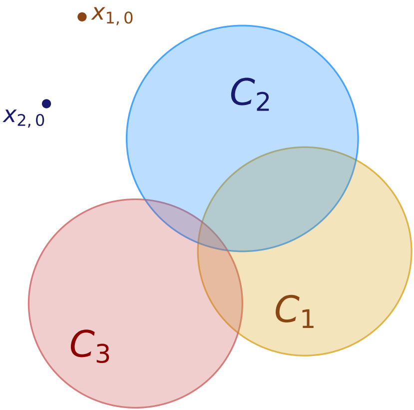
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

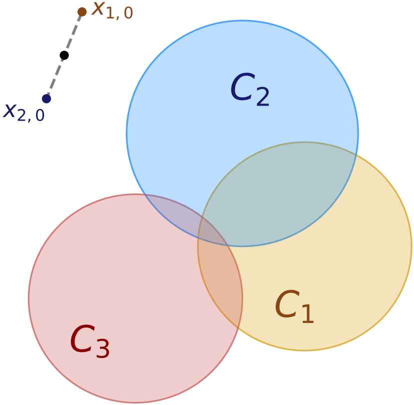
$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots .$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

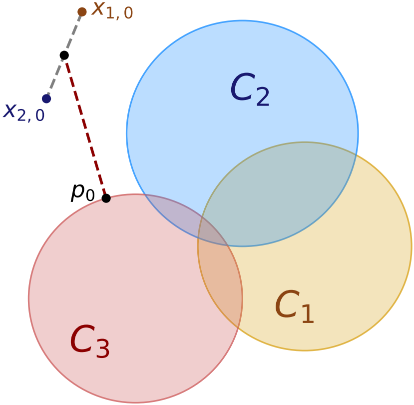
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

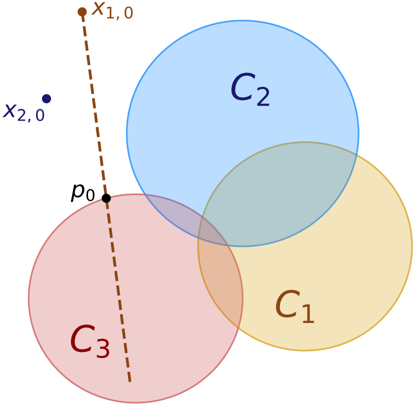
$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

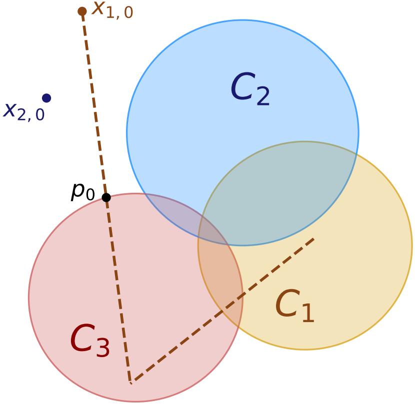
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

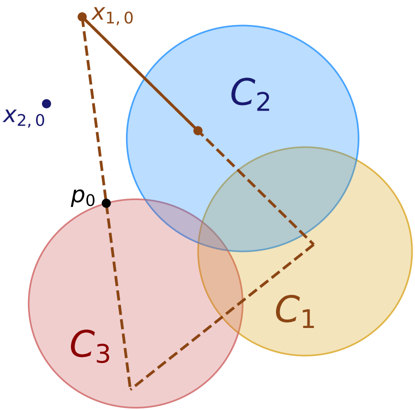
$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

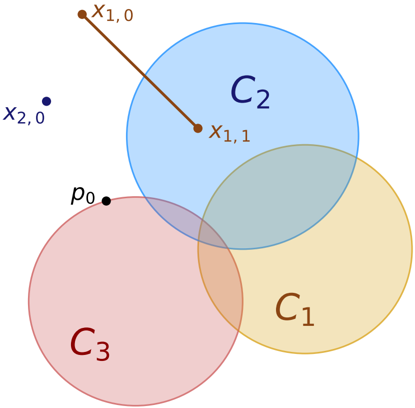
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^r A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1}A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

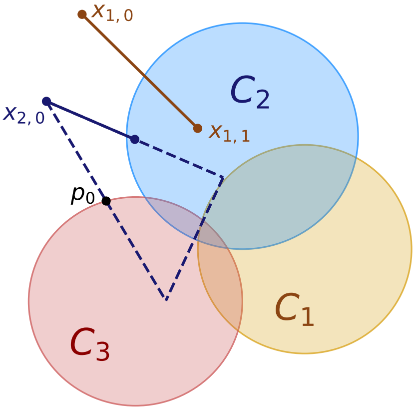
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \text{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

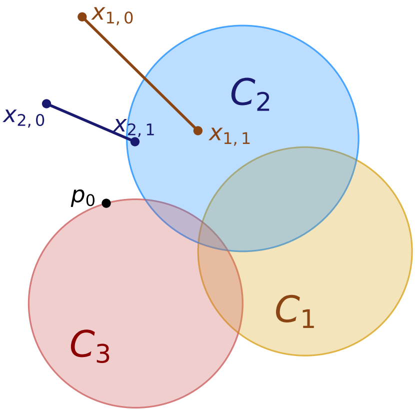
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

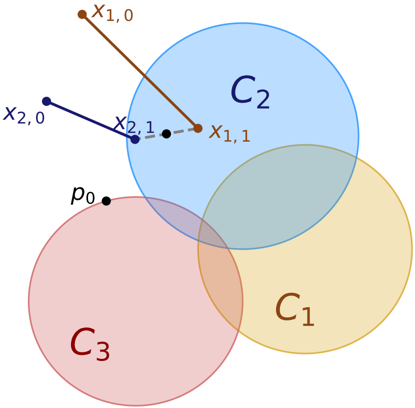
$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \mathrm{zer}(\sum_{i=1}^r A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

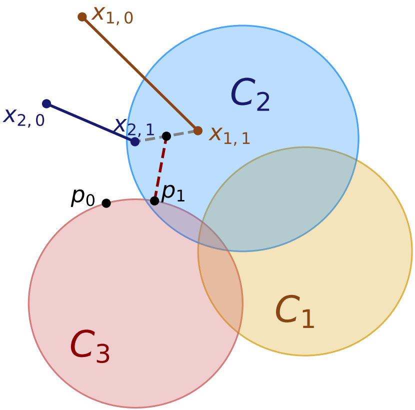
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

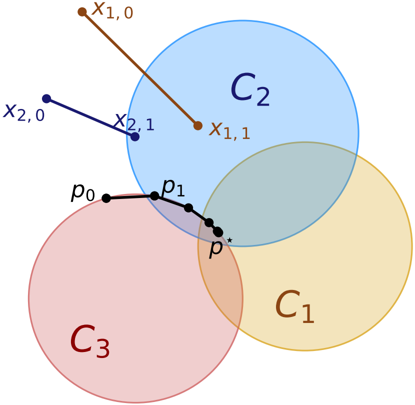
$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \ldots.$$



Product space reformulation with reduced dimension

Given $x_{1,0}, \ldots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \ldots$:

$$p_k = J_{\frac{\gamma}{r-1} A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \ldots, r-1$:

$$z_{i,k} = J_{\gamma A_i}(2p_k - x_{i,k}),$$

$$x_{i,k+1} = x_{i,k} + \lambda(z_{i,k} - p_k).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Convergence of Douglas–Rachford

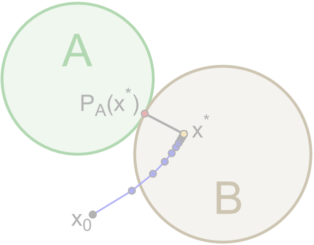
$$x_{n+1} = DR_{A,B}(x_n) := (1-\alpha)x_n + \alpha(2P_B - I)(2P_A - I)(x_n)$$

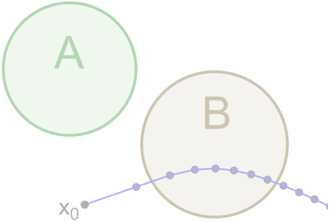
Theorem [Lions and Mercier (1979)]

Let $A, B \subseteq \mathcal{H}$ be closed and convex sets. Given any $x_0 \in \mathcal{H}$, for every $n \geq 0$, define $x_{n+1} = DR_{A,B,\alpha}(x_n)$. Then, the following holds.

(i) If $A \cap B \neq \emptyset$, then $\{x_n\} \to x^\star \in \operatorname{Fix} DR_{A,B,\alpha}$ such that $P_A(x^\star) \in A \cap B$.

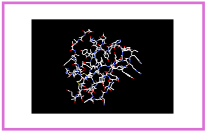
(ii) If $A \cap B = \emptyset$, then $\|x_n\| \to +\infty$.



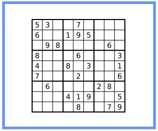


Douglas–Rachford in Non-Convex Settings

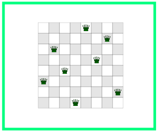
- The method has been successfully employed for solving many different nonconvex optimization problems, specially those of combinatorial nature.



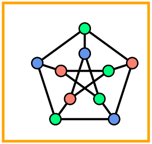
Protein reconstruction



Sudoku

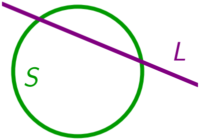


8 Queens

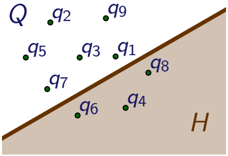


Graph coloring

- There are very few results explaining why the algorithm still works in nonconvex settings, and even less justifying its good global performance.



A sphere and a line,
Benoist (2015).

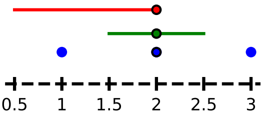


A half-space and a potentially non-convex set,
Aragón Artacho, Borwein and Tam (2016).

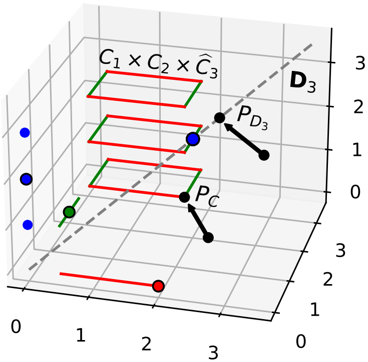
Product space reformulation

Example:

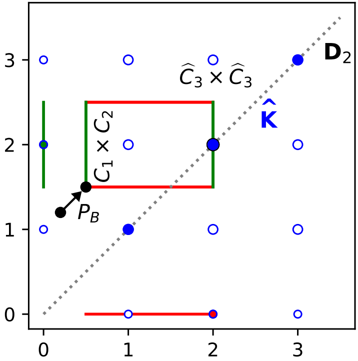
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



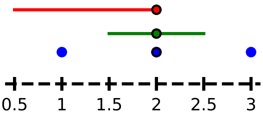
Product space reformulation with reduced dimension



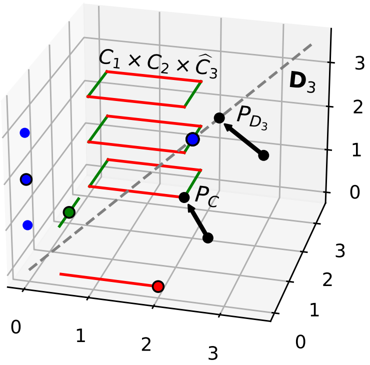
Product space reformulation

Example:

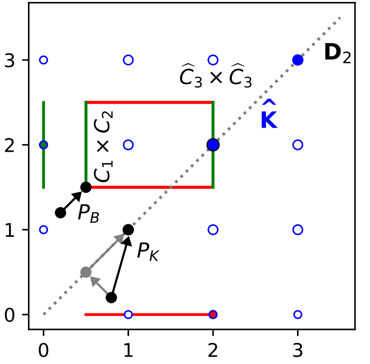
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$C_1 := [0.5, 2]$,
$C_2 := [1.5, 2.5]$,
$\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



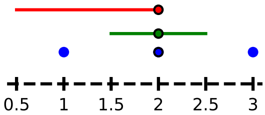
Product space reformulation
with reduced dimension



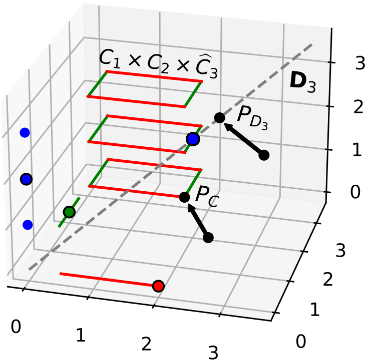
Product space reformulation

Example:

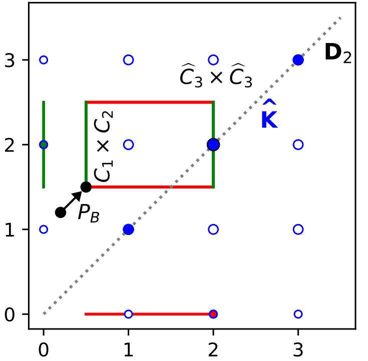
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



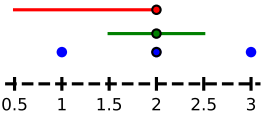
Product space reformulation with reduced dimension



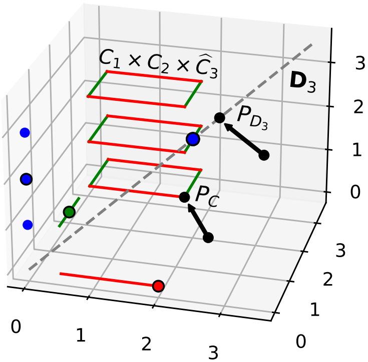
Product space reformulation

Example:

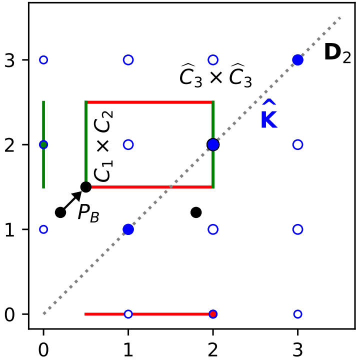
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



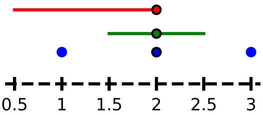
Product space reformulation with reduced dimension



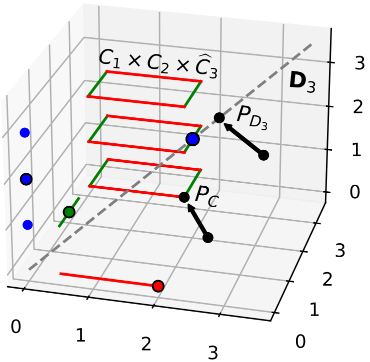
Product space reformulation

Example:

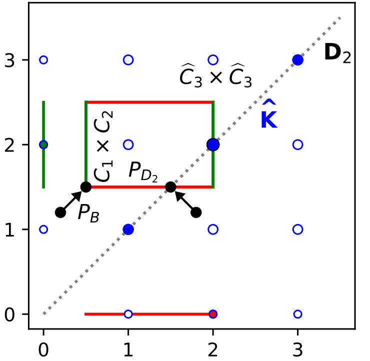
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with
$$C_1 := [0.5, 2],\quad C_2 := [1.5, 2.5],\quad \widehat{C}_3 := \{1, 2, 3\}.$$



Standard product space reformulation



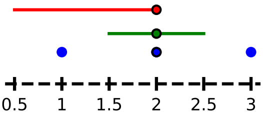
Product space reformulation with reduced dimension



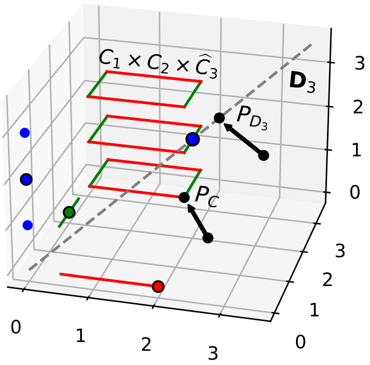
Product space reformulation

Example:

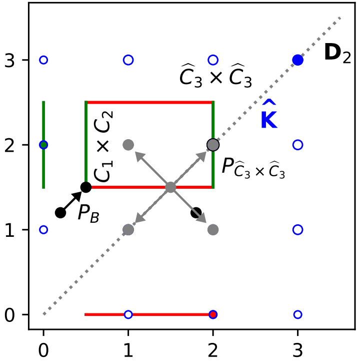
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



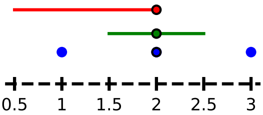
Product space reformulation with reduced dimension



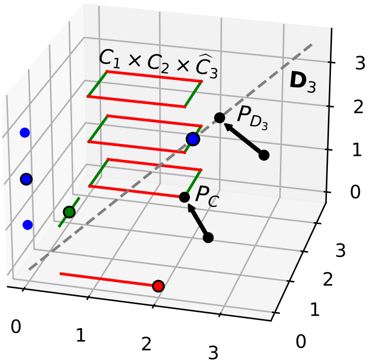
Product space reformulation

Example:

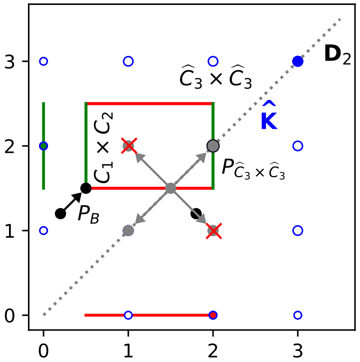
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



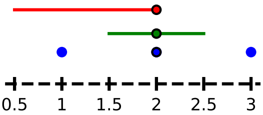
Product space reformulation with reduced dimension



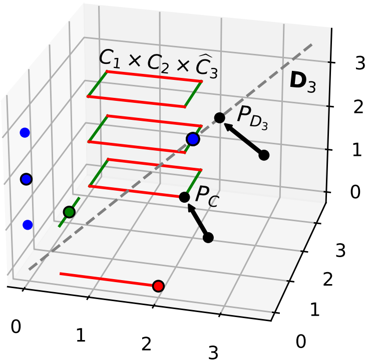
Product space reformulation

Example:

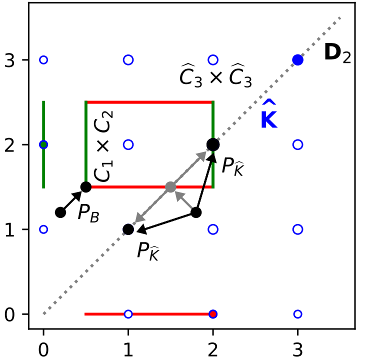
Find $x \in C_1 \cap C_2 \cap C_3 \subseteq \mathbb{R}$, with $C_1 := [0.5, 2]$, $C_2 := [1.5, 2.5]$, $\widehat{C}_3 := \{1, 2, 3\}$.



Standard product space reformulation



Product space reformulation with reduced dimension

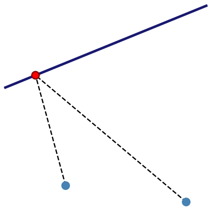


CONTENTS

CONTENTS

The Heron problem

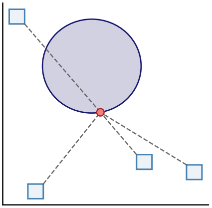
Find the point in a line L that minimizes the sum of the distances to two given points $x_1, x_2 \in \mathbb{R}^2$ in the plane:



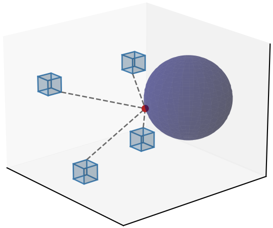
$$\text{(H)} \quad \text{Min} \quad \|x_1 - x\|^2 + \|x_2 - x\|^2$$
$$\text{s.t.} \quad x \in L \subseteq \mathbb{R}^2.$$

The generalized Heron problem

Find the point in a set $\Omega_r \subseteq \mathbb{R}^n$ that minimizes the sum of the distances to $r-1$ given sets $\Omega_1, \ldots, \Omega_{r-1} \subseteq \mathbb{R}^n$ in a eculidean space:



$$\text{(GH)} \quad \text{Min} \quad \sum_{i=1}^{r-1} d_{\Omega_i}(x) \qquad \text{s.t.} \quad x \in \Omega_r \subseteq \mathbb{R}^n.$$



The problem can be solved through

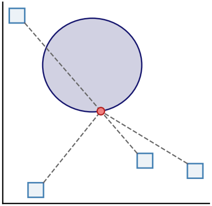
$$\text{Find } x^* \in \mathbb{R}^n \text{ such that } 0 \in \sum_{i=1}^{r-1} \partial d_{\Omega_i}(x^*) + N_{\Omega_r}(x^*).$$

Mordukhovich, B.S., Nam, N.M, Salinas, J.: Solving a generalized Heron problem by means of convex analysis. *Amer. Math. Monthly* 119(2), 87–99 (2012)

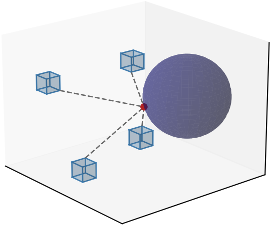
Mordukhovich, B.S., Nam, N.M, Salinas, J.: Applications of variational analysis to a generalized Heron problem. *Appl. Anal.* 91(10), 1915–1942 (2012)

The generalized Heron problem

Find the point in a set $\Omega_r \subseteq \mathbb{R}^n$ that minimizes the sum of the distances to $r-1$ given sets $\Omega_1, \ldots, \Omega_{r-1} \subseteq \mathbb{R}^n$ in a eculidean space:



$$\text{(GH)} \quad \text{Min} \quad \sum_{i=1}^{r-1} d_{\Omega_i}(x) \qquad \text{s.t.} \quad x \in \Omega_r \subseteq \mathbb{R}^n.$$



- We consider randomly generated instances with

$$\Omega_r := \text{ball} \quad \text{and} \quad \Omega_i := \text{hypercube}, \quad \forall i = 1, \ldots, r-1.$$

- We compare the performance of Standard-DR vs Reduced-DR.

Parallel Douglas–Rachford splitting algorithm

Given $x_0 \in \mathcal{H}^{r-1}$, set

$$x_{n+1} = (1-\lambda)x_n + \lambda(2J_{\gamma \boldsymbol{B}} - I)(2J_{\gamma \boldsymbol{K}} - I)(x_n), \quad n = 0, 1, 2, \dots.$$

Standard-DR

Given $x_{1,0}, x_{2,0}, \dots, x_{r,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \dots$:

$$p_k = \frac{1}{r}\sum_{i=1}^{r} x_{i,k},$$

for $i = 1, 2, \dots, r$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

Reduced-DR

Given $x_{1,0}, \dots, x_{r-1,0} \in \mathcal{H}$, set

for $k = 0, 1, 2, \dots$:

$$p_k = J_{\frac{\gamma}{r-1}A_r}\left(\frac{1}{r-1}\sum_{i=1}^{r-1} x_{i,k}\right),$$

for $i = 1, 2, \dots, r-1$:

$$z_{i,k} = J_{\gamma A_i}\left(2p_k - x_{i,k}\right),$$

$$x_{i,k+1} = x_{i,k} + \lambda\left(z_{i,k} - p_k\right).$$

Then $p_k \rightharpoonup p^* \in \operatorname{zer}(\sum_{i=1}^{r} A_i)$.

The generalized Heron problem

Find the point in a set $\Omega_r \subseteq \mathbb{R}^n$ that minimizes the sum of the distances to $r-1$ given sets $\Omega_1, \ldots, \Omega_{r-1} \subseteq \mathbb{R}^n$ in a eculidean space:

$$\text{(GH)} \quad \text{Min} \quad \sum_{i=1}^{r-1} d_{\Omega_i}(x)$$
$$\text{s.t.} \quad x \in \Omega_r \subseteq \mathbb{R}^n.$$

First we compute numerical experiments to choose the parameters of both algorithms

(results not shown)

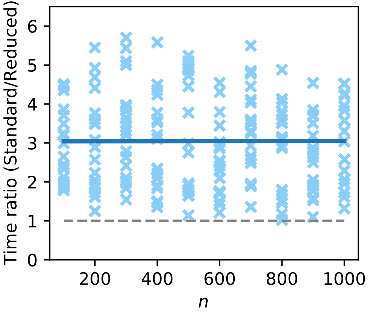
The generalized Heron problem

Find the point in a set $\Omega_r \subseteq \mathbb{R}^n$ that minimizes the sum of the distances to $r-1$ given sets $\Omega_1, \ldots, \Omega_{r-1} \subseteq \mathbb{R}^n$ in a eculidean space:

$$\text{(GH)} \quad \text{Min} \quad \sum_{i=1}^{r-1} d_{\Omega_i}(x)$$
$$\text{s.t.} \quad x \in \Omega_r \subseteq \mathbb{R}^n.$$

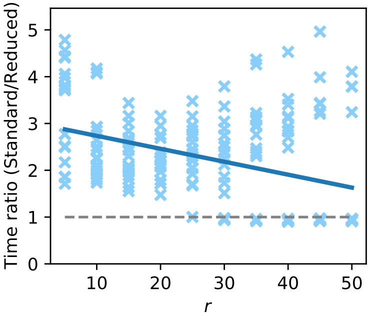
Experiment 1

Fixed $r = 5$



Experiment 2

Fixed $n = 100$



Douglas–Rachford with minimal lifting

- Consider a monotone inclusion described by **two operators**:
  - DR on the product space reformulation: 2-fold lifting
  - DR on the original problem: no lifting (1-fold lifting) $\leftarrow$ Minimal
- Consider now the case of **three operators**:
  - DR on the product space reformulation: 3-fold lifting $\leftarrow$ Minimal?

  Ryu, E. K.: Uniqueness of DRS as the 2 operator resolvent-splitting and impossibility of 3 operator resolvent-splitting. *Math. Program.* 182(1), 233–273 (2020)

  $\longrightarrow$ Imposibility of 1-fold lifting $+$ Minimal lifting: 2-fold
- Can it be generalized for an arbitrary family of **r operators**?
  - DR on the product space reformulation: r-fold lifting $\leftarrow$ Minimal?
  - Minimal lifting: $(r-1)$-fold $\longleftarrow$

  Malitsky, Y., Tam, M. K.: Resolvent Splitting for Sums of Monotone Operators with Minimal Lifting. *ArXiv Preprint* (2021)

The generalized Heron problem

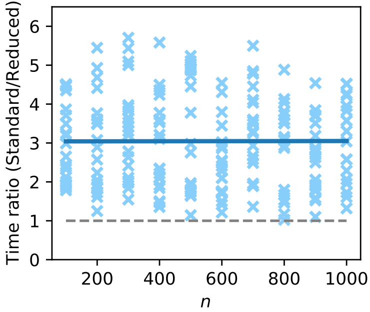
Find the point in a set $\Omega_r \subseteq \mathbb{R}^n$ that minimizes the sum of the distances to $r-1$ given sets $\Omega_1, \ldots, \Omega_{r-1} \subseteq \mathbb{R}^n$ in a eculidean space:

$$\text{(GH)} \quad \text{Min} \quad \sum_{i=1}^{r-1} d_{\Omega_i}(x) \qquad \text{s.t.} \quad x \in \Omega_r \subseteq \mathbb{R}^n.$$

We incorporate Ryu algorithm into the comparison

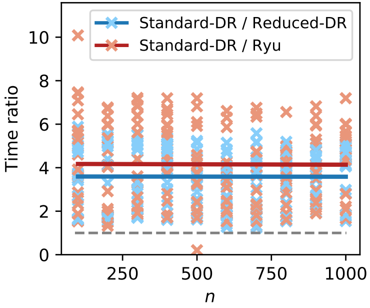
Experiment 1

Fixed $r = 5$



Experiment 3

Fixed $r = 3$



CONTENTS

Sudokus

A **Sudoku** is a puzzle whose objective is to fill a 9×9 grid with digits from 1 to 9 verifying the following constraints:

- Some cells are already filled and fixex.
- Each row must contain all digits from 1 to 9 exactly once.
- Each column must contain all digits from 1 to 9 exactly once.
- Each 3×3 subrid must contain all digits from 1 to 9 exactly once.

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

Sudokus as feasibility problems

A Sudoku can be modeled as a feasibility problem.

Sudokus as feasibility problems

A Sudoku can be modeled as a feasibility problem. Consider the following sets:

- $C_1 := \{$Completions of the given matrix$\}$

0	0	5	3	0	0	0	0	0
8	0	0	0	0	0	0	2	0
0	7	0	0	1	0	5	0	0
4	0	0	0	0	5	0	0	0
0	1	0	0	7	0	0	0	6
0	0	3	0	0	0	0	8	0
0	6	0	5	0	0	0	0	9
0	0	4	0	0	0	0	3	0
0	0	0	0	0	9	7	0	0

1	2	5	3	3	1	9	0	8
8	1	2	0	2	4	3	2	6
1	7	2	5	1	2	5	2	3
4	2	6	4	0	5	3	1	1
6	1	8	0	7	2	7	0	6
2	0	3	0	4	0	5	8	8
1	6	4	5	6	5	5	2	9
5	5	4	1	7	5	2	3	8
8	6	3	2	1	9	7	1	9

Sudokus as feasibility problems

A Sudoku can be modeled as a feasibility problem. Consider the following sets:

- $C_1 := \{$Completions of the given matrix$\}$
- $C_2 := \{$Matrices whose rows are permutations of $\{1, 2, \ldots 9\}\}$

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

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

Sudokus as feasibility problems

A Sudoku can be modeled as a feasibility problem. Consider the following sets:

- $C_1 := \{$Completions of the given matrix$\}$
- $C_2 := \{$Matrices whose rows are permutations of $\{1, 2, \ldots 9\}\}$
- $C_3 := \{$Matrices whose columns are permutations of $\{1, 2, \ldots 9\}\}$

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

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

Sudokus as feasibility problems

A Sudoku can be modeled as a feasibility problem. Consider the following sets:

- $C_1 := \{$Completions of the given matrix$\}$
- $C_2 := \{$Matrices whose rows are permutations of $\{1, 2, \dots 9\}\}$
- $C_3 := \{$Matrices whose columns are permutations of $\{1, 2, \dots 9\}\}$
- $C_4 := \{$Matrices whose 3x3 subgrids are permutations of $\{1, 2, \dots 9\}\}$

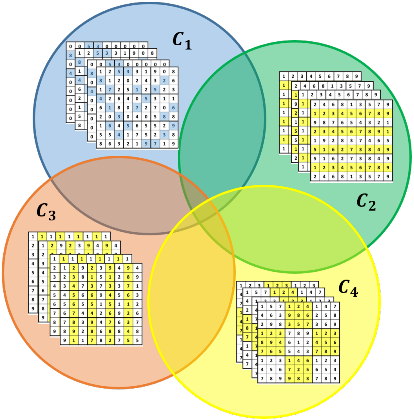
1	2	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9
1	2	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9
1	2	3	1	2	3	1	2	3
4	5	6	4	5	6	4	5	6
7	8	9	7	8	9	7	8	9

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

Sudokus as feasibility problems

Solving the Sudoku is equivalent to solving the following nonconvex feasibility problem:

$$\text{(P)} \quad \text{Find } M \in C_1 \cap C_2 \cap C_3 \cap C_4$$

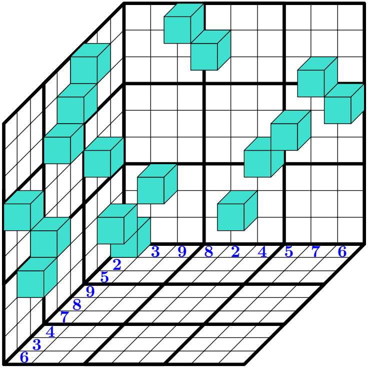


DR fails to solve the previous feasibility problem

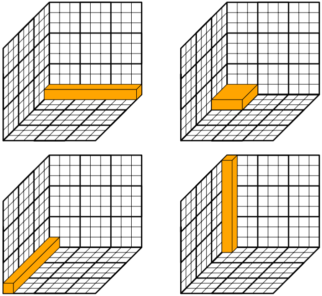
Sudokus as feasibility problems (reformulated)

The problem can be reformulated as a 3-dimensional multiarray $X \in \mathbb{R}^{9\times 9\times 9}$ with binary entries defined componentwise as

$$X[i,j,k] = \begin{cases} 1, & \text{if digit } k \text{ is assigned to the } (i,j)\text{th entry of the Sudoku,} \\ 0, & \text{otherwise;} \end{cases}$$



Feasibility constraint sets



Sudokus as feasibility problems (reformulated)

The problem can be reformulated as a 3-dimensional multiarray $X \in \mathbb{R}^{9\times 9\times 9}$ with binary entries defined componentwise as

$$X[i,j,k] = \begin{cases} 1, & \text{if digit } k \text{ is assigned to the } (i,j)\text{th entry of the Sudoku}, \\ 0, & \text{otherwise;} \end{cases}$$

- Elser, V., Rankenburg, I., Thibault, P.: Searching with iterated maps. *Proc. Natl. Acad. Sc.* 104(2), 418–423 (2007)
- 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)

Sudokus as feasibility problems (reformulated)

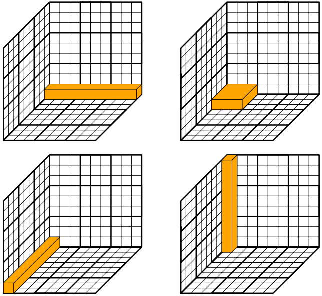
The problem can be reformulated as a 3-dimensional multiarray $X \in \mathbb{R}^{9\times 9\times 9}$ with binary entries defined componentwise as

$$X[i,j,k] = \begin{cases} 1, & \text{if digit } k \text{ is assigned to the } (i,j)\text{th entry of the Sudoku}, \\ 0, & \text{otherwise;} \end{cases}$$

Projections onto the constraint sets are computed through the projector onto the canonical basis:

$$\text{Max}\ \begin{pmatrix} x_1 \\ \boxed{x_2} \\ \vdots \\ x_9 \end{pmatrix} \overset{P}{\rightarrow} \left\{ \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \right\}$$

Feasibility constraint sets



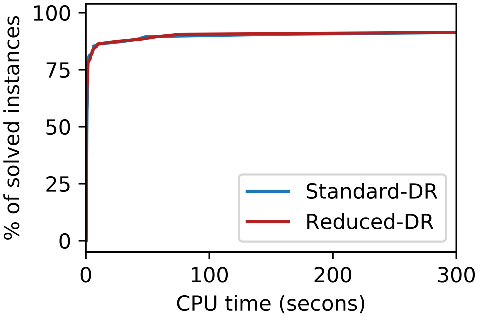
Sudokus as feasibility problems (reformulated)

- Again, we compare the performance of Standard-DR vs Reduced-DR.
- We consider 95 hard Sudokus from the dataset `top95`.
- For each sudoku: 10 random initializations.
- Instances were labeled as *unsolved* after 5 minutes of CPU running time.

Algorithm	Solved	Wins	Average time
Standard-DR	89.68%	23.79%	3.95 s
Reduced-DR	90.42%	66.52%	3.11 s

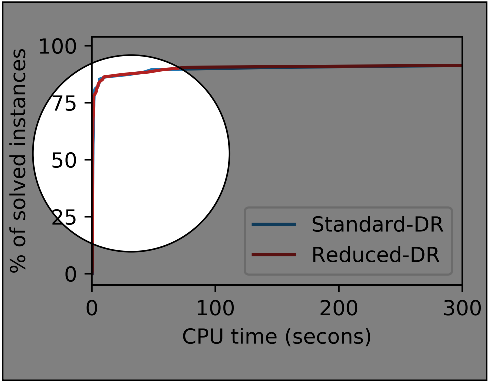
Sudokus as feasibility problems (reformulated)

- Again, we compare the performance of Standard-DR vs Reduced-DR.
- We consider 95 hard Sudokus from the dataset `top95`.
- For each sudoku: 10 random initializations.
- Instances were labeled as *unsolved* after 5 minutes of CPU running time.



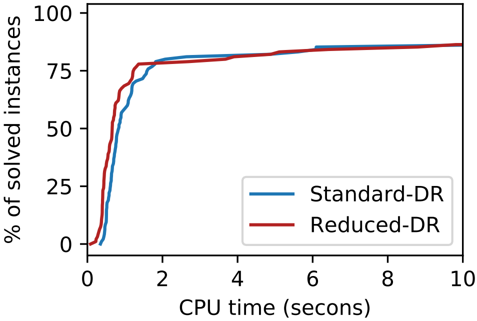
Sudokus as feasibility problems (reformulated)

- Again, we compare the performance of Standard-DR vs Reduced-DR.
- We consider 95 hard Sudokus from the dataset `top95`.
- For each sudoku: 10 random **initializations**.
- Instances were labeled as *unsolved* after 5 **minutes** of CPU running time.



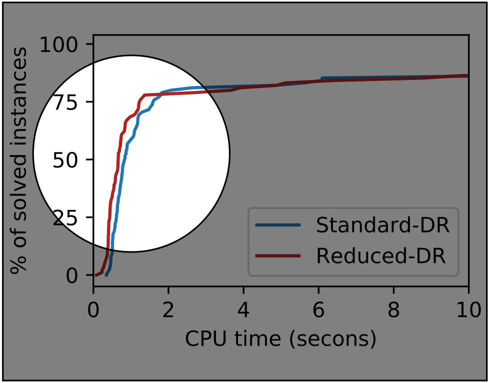
Sudokus as feasibility problems (reformulated)

- Again, we compare the performance of Standard-DR vs Reduced-DR.
- We consider 95 hard Sudokus from the dataset `top95`.
- For each sudoku: 10 random initializations.
- Instances were labeled as *unsolved* after 5 minutes of CPU running time.



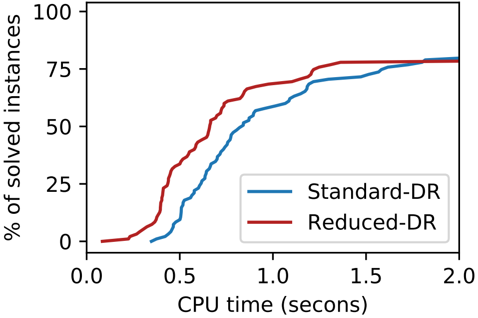
Sudokus as feasibility problems (reformulated)

- Again, we compare the performance of Standard-DR vs Reduced-DR.
- We consider 95 hard Sudokus from the dataset top95.
- For each sudoku: 10 random initializations.
- Instances were labeled as *unsolved* after 5 minutes of CPU running time.



Sudokus as feasibility problems (reformulated)

- Again, we compare the performance of Standard-DR vs Reduced-DR.
- We consider 95 hard Sudokus from the dataset `top95`.
- For each sudoku: 10 random initializations.
- Instances were labeled as *unsolved* after 5 minutes of CPU running time.



Thank you for your attention!

MAIN REFERENCE

Campoy, R.: A Product Space Reformulation with Reduced Dimension for Splitting Algorithms.
ArXiv Preprint (July 2021) http://arxiv.org/abs/2107.12355

ruben.campoy@uv.es

https://sites.google.com/view/rcampoy

Ruben_Campoy