Chapter 1
Numerical explorations of feasibility algorithms for finding points in the intersection of finite sets

Heinz H. Bauschke, Sylvain Gretchko, and Walaa M. Moursi

Abstract Projection methods are popular algorithms for iteratively solving feasibility problems in Euclidean or even Hilbert spaces. They employ (selections of) nearest point mappings to generate sequences that are designed to approximate a point in the intersection of a collection of constraint sets. Theoretical properties of projection methods are fairly well understood when the underlying constraint sets are convex; however, convergence results for the nonconvex case are more complicated and typically only local. In this paper, we explore the perhaps simplest instance of a feasibility algorithm, namely when each constraint set consists of only finitely many points. We numerically investigate four constellations: either few or many constraint sets, with either few or many points. Each constellation is tackled by four popular projection methods each of which features a tuning parameter. We examine the behaviour for a single and for a multitude of orbits, and we also consider local and global behaviour. Our findings demonstrate the importance of the choice of the algorithm and that of the tuning parameter.

Key words: cyclic Douglas–Rachford algorithm, Douglas–Rachford algorithm, extrapolated parallel projection method, method of cyclic projections, nonconvex feasibility problem, optimization algorithm, projection
AMS 2010 Subject Classification: 49M20, 49M27, 49M37, 65K05, 65K10, 90C25, 90C26, 90C30

Heinz H. Bauschke
Mathematics, UBCO, ASC 352, 3187 University Way, Kelowna, B.C. V1V 1V7, Canada, e-mail: heinz.bauschke@ubc.ca

Sylvain Gretchko
Mathematics, UBCO, ASC 352, 3187 University Way, Kelowna, B.C. V1V 1V7, Canada, e-mail: sylvain.gretchko@gmail.com,

Walaa M. Moursi
Electrical Engineering, Stanford University, 350 Serra Mall, Stanford, CA 94305, USA,
and Mansoura University, Faculty of Science, Mathematics Department, Mansoura 35516, Egypt
e-mail: wmoursi@stanford.edu

1.1 Introduction

Background Let X be a Euclidean space (i.e., a finite-dimensional Hilbert space), with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$. The *feasibility problem* is a common problem in science and engineering: given finitely many closed subsets $C_1, \ldots, C_m$ of X, it asks to

$$\text{Find } x \in C := C_1 \cap \cdots \cap C_m. \tag{FP}$$

We henceforth assume that the intersection C is nonempty. Algorithms for solving (FP) exist when the constraint sets C_i allow for simple projectors P_{C_i} (i.e., nearest-point mappings). When C_i is convex, then the projector P_{C_i} is a nice (firmly nonexpansive and single-valued) operator defined on the entire space X; when C_i is not convex, then P_{C_i} is nonempty and set-valued. For notational simplicity, we will use P_{C_i} to denote an arbitrary but fixed selection of the set-valued projector. (If S is a subset of X, then $P_S(x)$ is a minimizer of the function $s \mapsto \|x - s\|$, where $s \in S$. For other notions not explicitly defined in this paper, we refer the reader to [1].)

Assuming that the operators $P_{C_1}, \ldots, P_{C_m}$ are readily available and implementable, one may try to solve (FP) iteratively by generating a sequence $(x_k)_{k \in \mathbb{N}}$ of vectors in X that employs the projection operators P_{C_i} in some fashion to produce the next update. There are hundreds of papers dealing with algorithms for solving convex or nonconvex feasibility problems. Thus, we refrain from providing a comprehensive list of references and rather point to the following recent books and "meta" papers as starting points: [1, 2, 4, 10, 12, 13, 14, 15]. (We note that the recent manuscript [5] deals with a feasibility problem where one set is a doubleton.) The convergence theory in the nonconvex case is much more challenging and usually of local character.

Goal of this paper *The goal of this paper is to showcase the surprising numerical complexity of the most simple instance of* (FP); namely, when each constraint set

$$C_i \text{ contains a } \textit{finite} \text{ number of points.}$$

In this case, the projection operator is very easy to implement — this is achieved by simply measuring the distance of the point to each point in C_i and returning the closest one. Furthermore, we will restrict ourselves to the simple case when the underlying space

$$X = \mathbb{R}^2$$

is simply the *Euclidean plane*. Even in this setting, the difficulty and richness of the dynamic behaviour is impressively illustrated.

It is our hope that the complexity revealed will spark further analytical research in feasibility algorithms with the goal to explain the observed complexity and ulti-

mately to aid in the design of new algorithms for solving difficult feasibility problems.

Organization of the paper The remainder of the paper is organized as follows. In Section 1.2, we present the four constellations we will use for our numerical exploration throughout the remainder of the paper. These constellations correspond to feasibility problems that we will attempt to solve using the algorithms listed in Section 1.3. Section 1.4 provides details on the implementation and execution of the numerical experiments. The "best" tuning parameter λ_{best} is determined in Section 1.5. We then track typical orbits of the algorithms in Section 1.6. Local and global behaviour is investigated in Section 1.7. Some interesting (and beautiful) behaviour outside of the main numerical experiments are collected in Section 1.8. The final Section 1.9 contains some concluding remarks.

1.2 The four constellations

Even though we restrict ourselves already to finitely many constraint sets with finitely many points in the Euclidean plane, the infinitely many possibilities to experiment make it a daunting task to explore this space. We opted to probe this universe as follows.

The points in each constraint set C_i are chosen randomly. We will ensure that the origin belongs to each set C_i

$$0 \in C_i \subset [-10,10] \times [-10,10]$$

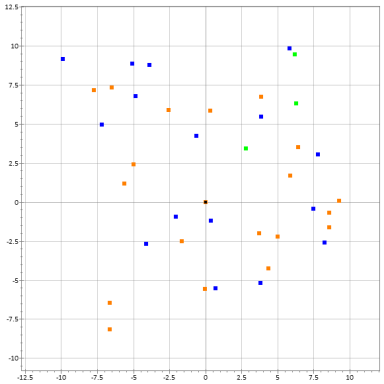
to have a consistent feasibly problem with

$$C = C_1 \cap \cdots \cap C_m = \{0\}.$$

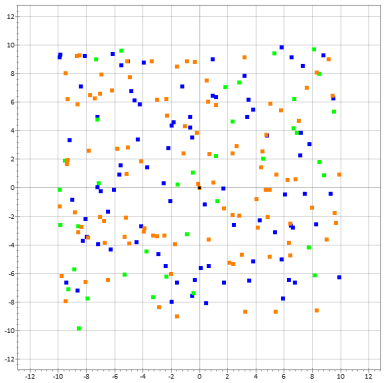
We will focus on two alternatives for the *number of constraint sets*, either "few" or "many". We will also consider constraint sets with a *maximum number of points* in the constraint sets, either "few" or "many". From now on, we will use the following language:

- The number of **few sets** is 3.
- The number of **many sets** is 10.
- The number of **few points** is 20.
- The number of **many points** is 100.

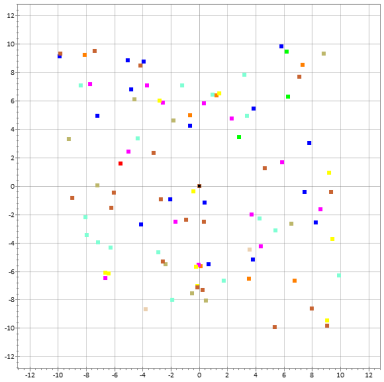
This will give rise to *four constellations*: few sets with few points, few sets with many points, many sets with few points, and many sets with many points. The four constellations used in our numerical experiments are shown in Figure 1.1.



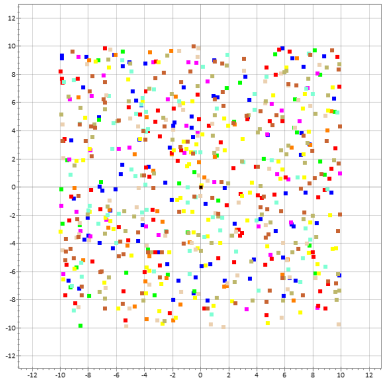
(a) Few sets with few points



(b) Few sets with many points



(c) Many sets with few points



(d) Many sets with many points

Fig. 1.1: The four constellations explored in this paper. See Section 1.2 for further information.

1.3 The four feasibility algorithms

We will numerically solve instances of (FP) using four algorithms which we briefly review in this section. While there is a myriad of competing algorithms available, our selection consists of trustworthy "work horses" that have been employed else-

where and for which the convergence theory in the *convex* case is fairly well understood. Each of these algorithms has a "tuning" parameter λ in the range $]0,2[$. The *default value* λ_{default} is 1. Guided by experiments, we will also (numerically) look for the "best" value λ_{best}. We now turn to these four algorithms. Each algorithm will have a *governing sequence* driving the iteration, and a (possibly different) *monitored sequence* which is meant to find a solution of (FP).

Cyclic Projections (CycP) Given $x_0 \in X$, the governing sequence is defined by

$$x_{k+1} := \big((1-\lambda)\mathrm{Id} + \lambda P_{C_m}\big) \circ \cdots \circ \big((1-\lambda)\mathrm{Id} + \lambda P_{C_1}\big) x_k. \tag{1.1}$$

The default parameter is $\lambda_{\text{default}} = 1$, from the range $]0,2[$. The sequence monitored is $\big(\frac{1}{m}\sum_{i=1}^m P_{C_i} x_k\big)_{k\in\mathbb{N}}$. Selected references: [1, 2, 4, 10, 11, 13, 15].

Extrapolated Parallel Projections (ExParP) Given $x_0 \in X$, the governing and monitored sequence is defined by

$$x_{k+1} := x_k + \lambda \cdot \frac{\sum_{i=1}^m \|x_k - P_{C_i} x_k\|^2}{\|\sum_{i=1}^m (x_k - P_{C_i} x_k)\|^2} \sum_{i=1}^m (P_{C_i} x_k - x_k) \tag{1.2}$$

if $x_k \notin C$; $x_{k+1} = x_k$ otherwise. The default parameter is $\lambda_{\text{default}} = 1$, from the range $]0,2[$. Selected references: [2, 3, 14].

Douglas–Rachford (DR) Given $x_0 \in X$, $\mathbf{x}_0 := (x_{0,1}, \ldots, x_{0,m}) = (x_0, \ldots, x_0) \in \mathbf{X} := X^m$, $\mathbf{x}_k = (x_{k,1}, \ldots, x_{k,m}) \in \mathbf{X}$, and $\bar{x}_k := \frac{1}{m}\sum_{i=1}^m x_{k,i}$, the next iterate is $\mathbf{x}_{k+1} = (x_{k+1,1}, \ldots, x_{k+1,m})$, where

$$(\forall i \in \{1, \ldots, m\}) \quad x_{k+1,i} := x_{k,i} + \lambda \big(P_{C_i}(2\bar{x}_k - x_{k,i}) - \bar{x}_k\big). \tag{1.3}$$

The default parameter is $\lambda_{\text{default}} = 1$, from the range $]0,2[$. The sequence monitored is $(\bar{x}_k)_{k\in\mathbb{N}}$. Selected references: [2, 6, 17, 18, 19].

Cyclic Douglas–Rachford (CycDR) Given $x_0 \in X$, the governing sequence is defined by

$$\begin{aligned} x_{k+1} := {} & \big((1-\tfrac{\lambda}{2})P_{C_m} + \tfrac{\lambda}{4}\big(\mathrm{Id} + R_{C_1} R_{C_m}\big)\big) \circ \cdots \circ \\ & \big((1-\tfrac{\lambda}{2})P_{C_2} + \tfrac{\lambda}{4}\big(\mathrm{Id} + R_{C_3} R_{C_2}\big)\big) \circ \big((1-\tfrac{\lambda}{2})P_{C_1} + \tfrac{\lambda}{4}\big(\mathrm{Id} + R_{C_2} R_{C_1}\big)\big) x_k. \end{aligned} \tag{1.4}$$

The default parameter is $\lambda_{\text{default}} = 1$, from the range $]0,2[$. The sequence monitored is $\left(\frac{1}{m}\sum_{i=1}^{m} P_{C_i} x_k\right)_{k\in\mathbb{N}}$. Selected references: [20, 16, 21]. (For other cyclic version of DR, see [7, 9]. Also, if $m = 2$ and $C_1 = C_2 = \{0\}$, then $(x_k)_{k\in\mathbb{N}} = ((\lambda/2)^k x_0)_{k\in\mathbb{N}}$ is actually unbounded when $x_0 \neq 0$ and $\lambda > 2$.)

1.4 Setting up the numerical explorations

Stopping criteria The feasibility measure

$$d : X \to \mathbb{R}_+ : x \mapsto \sqrt{\frac{\sum_{i=1}^{m} \|x - P_{C_i} x\|^2}{\sum_{i=1}^{m} \|x_0 - P_{C_i} x_0\|^2}},$$

where $x_0 \in X \smallsetminus C$, vanishes exactly when $x \in C$. We stop each algorithm with monitored sequence $(y_k)_{k\in\mathbb{N}}$ either when

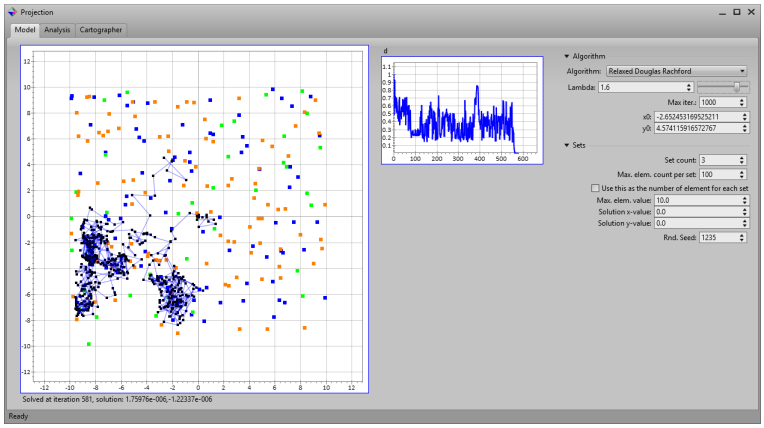
$$d(y_k) < \varepsilon := 10^{-6}$$

or when the maximum number of iterations, which we set to 1000, is reached. These values were chosen to allow a reasonable exploration of the feasibility problem while maintaining computational efficiency.

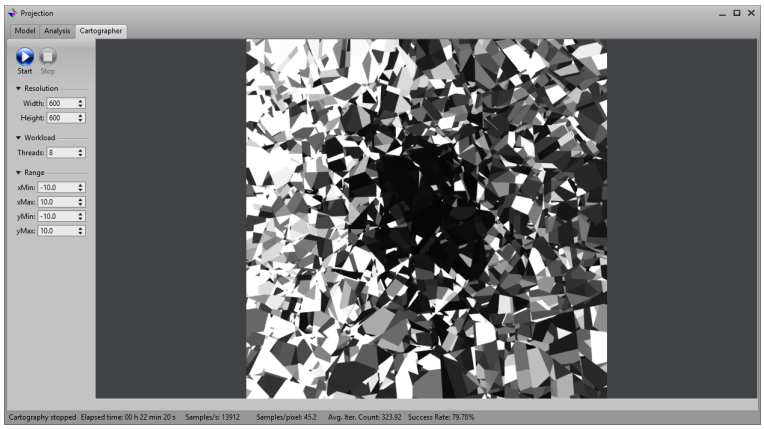
Details on program A program was developed in C++ to run the different experiments, see Figure 1.2 for two screenshots which we describe next. In the main tab of the user interface one can select the algorithm to be used and set up the problem to be solved by choosing the number of sets and the maximum number of elements per set. By clicking on the diagram showing the current constellation of points, the user can select a starting point and immediately observe the resulting orbit being rendered over the constellation. The graph of the feasibility measure d, corresponding to the current orbit, is also displayed.
The Cartographer tab allows the exploration of a very large number of starting points to construct a picture of the performance of a given algorithm. This two-dimensional plot shows for each starting point the number of iterations required to solve the problem, ranging from zero (black) to the maximum number of iterations allowed (white). The plot is generated progressively and uses Quasi-Monte Carlo sampling for the selection of the starting points. This is the most computationally intensive part of the software, and it is fully multi-threaded to take advantage of modern processor architectures.

1.5 Determining the "best" parameter λ_{best}

In this section, we consider our four constellations (see Section 1.2) and run on each of them the four algorithms (see Section 1.3) with the parameter λ ranging over $]0,2[$. The curves shown in Fig. 1.3, 1.4, 1.5, and 1.6 give an estimate of the success rate of each algorithm, evaluated for 200 evenly-spaced values of λ. For each value



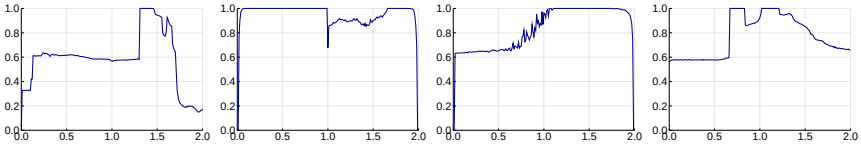
(a)



(b)

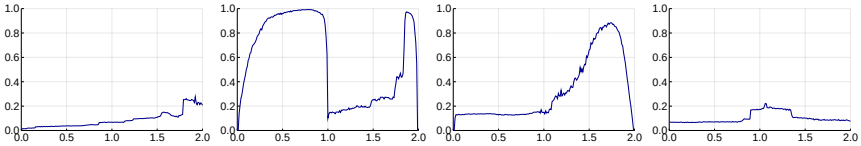
Fig. 1.2: The software developed for this work. Setting the constellation of points and the algorithm to be used is done in the main tab, shown in (a). The generation of a performance plot is done in the cartographer tab, shown in (b).

of λ, 5000 starting points are drawn from $[-10,10]\times[-10,10]$ using Quasi-Monte Carlo sampling, and the success rate is estimated by dividing the number of times the algorithm is successful by this number of starting points. Thus, a "best" parameter λ_{best} is determined. It is this parameter that we will use to compare with the default parameter λ_{default}, which is 1 in all cases.



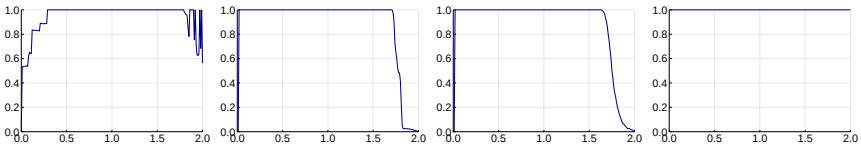
(a) CycP (b) ExParP (c) DR (d) CycDR

Fig. 1.3: Success rates in terms of λ for the few sets with few points constellation.



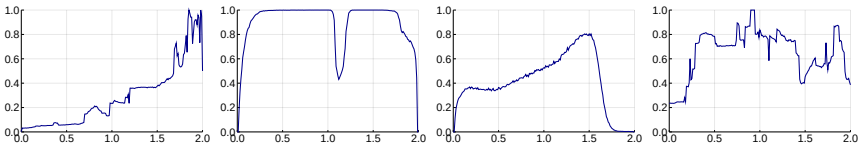
(a) CycP (b) ExParP (c) DR (d) CycDR

Fig. 1.4: Success rates in terms of λ for the few sets with many points constellation.



(a) CycP (b) ExParP (c) DR (d) CycDR

Fig. 1.5: Success rates in terms of λ for the many sets with few points constellation.



(a) CycP (b) ExParP (c) DR (d) CycDR

Fig. 1.6: Success rates in terms of λ for the many sets with many points constellation.

Discussion For each of the four constellations considered above, we visually inspected the λ-curves indicating success rates. We then picked for each algorithm a parameter called λ_{best} which improved performance over the default parameter $\lambda_{\text{default}} = 1$. The results are recorded in the following table.

Algorithm	CycP	ExParP	DR	CycDR
λ_{default}	1.0	1.0	1.0	1.0
λ_{best}	1.5	0.8	1.6	1.2

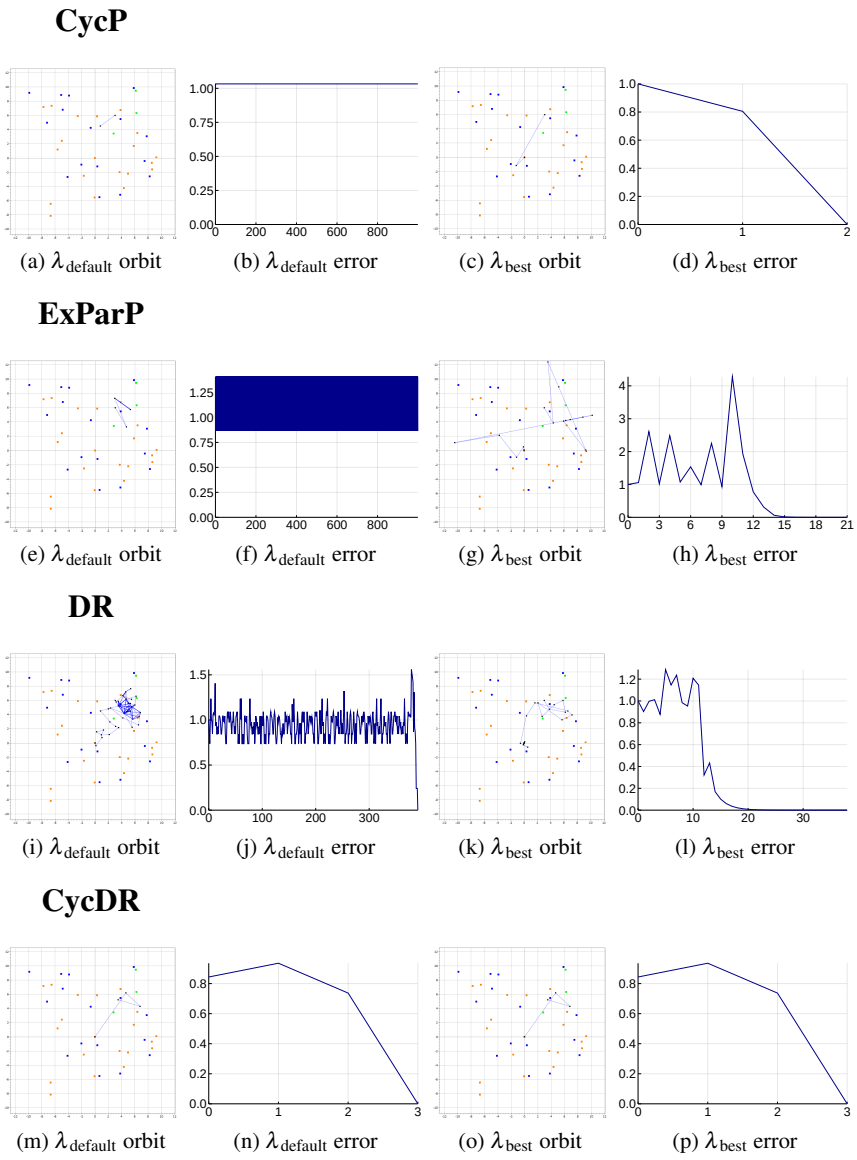
Fig. 1.7: Best parameters λ_{best} chosen by inspecting the success rates curves

We will use the parameters λ_{best} for the experiments in subsequent sections.

1.6 Tracking orbits

In this section, we consider our four given constellations (see Section 1.2). For each constellation, which is organized in a separate subsection, the same starting point is used. We then consider each of our four fixed algorithms (see Section 1.3) and show orbits for $\lambda_{\text{default}} = 1$ and for λ_{best} (see Section 1.5), and the corresponding feasibility measure d (see Section 1.4).

1.6.1 Few sets with few points



(a) λ_{default} orbit (b) λ_{default} error (c) λ_{best} orbit (d) λ_{best} error

(e) λ_{default} orbit (f) λ_{default} error (g) λ_{best} orbit (h) λ_{best} error

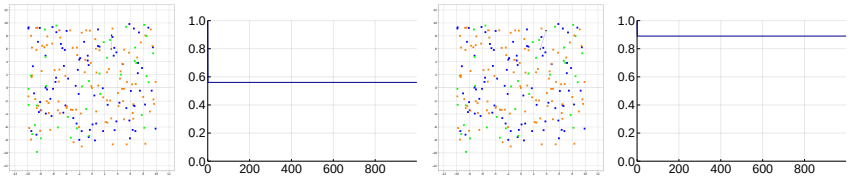
(i) λ_{default} orbit (j) λ_{default} error (k) λ_{best} orbit (l) λ_{best} error

(m) λ_{default} orbit (n) λ_{default} error (o) λ_{best} orbit (p) λ_{best} error

Fig. 1.8: Orbits and errors for CycP, ExParP, DR, and CycDR in the few sets with few points constellation

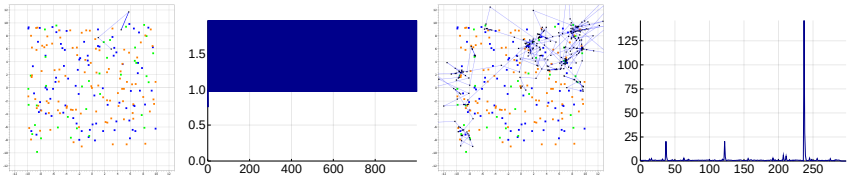
1.6.2 Few sets with many points

CycP



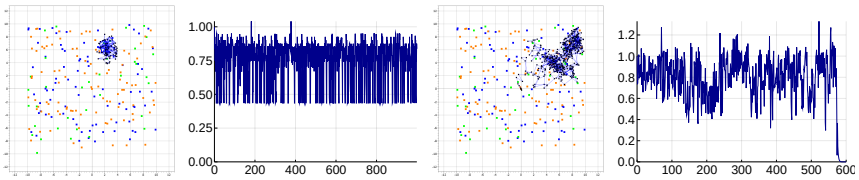
(a) λ_{default} orbit (b) λ_{default} error (c) λ_{best} orbit (d) λ_{best} error

ExParP



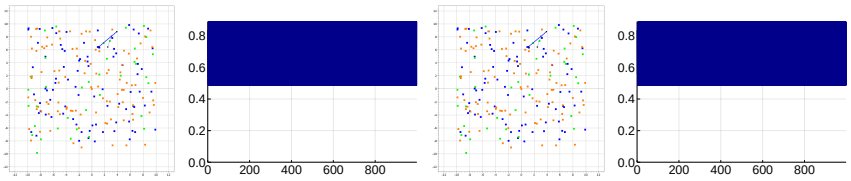
(e) λ_{default} orbit (f) λ_{default} error (g) λ_{best} orbit (h) λ_{best} error

DR



(i) λ_{default} orbit (j) λ_{default} error (k) λ_{best} orbit (l) λ_{best} error

CycDR

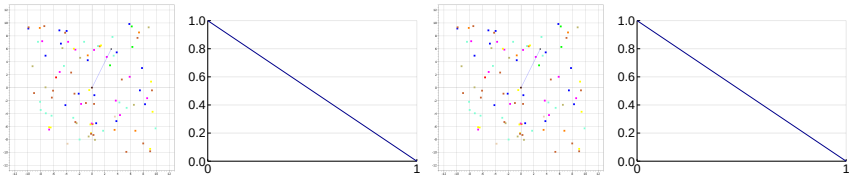


(m) λ_{default} orbit (n) λ_{default} error (o) λ_{best} orbit (p) λ_{best} error

Fig. 1.9: Orbits and errors for CycP, ExParP, DR, and CycDR in the few sets with many points constellation

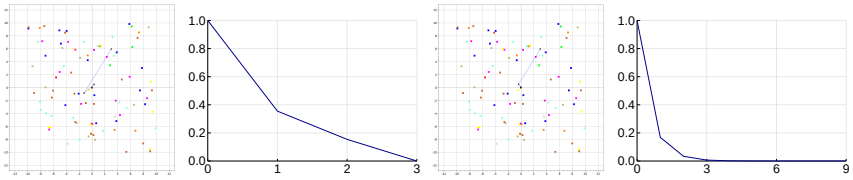
1.6.3 Many sets with few points

CycP



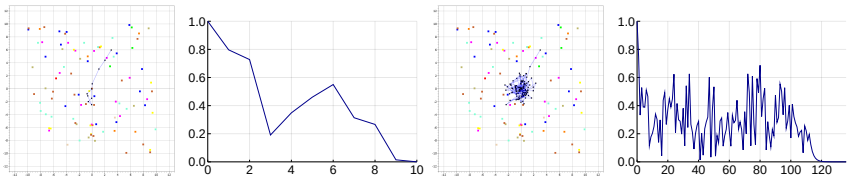
(a) λ_{default} orbit (b) λ_{default} error (c) λ_{best} orbit (d) λ_{best} error

ExParP



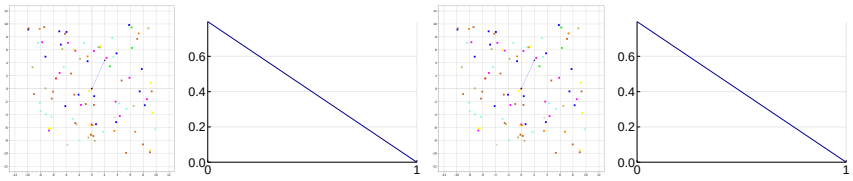
(e) λ_{default} orbit (f) λ_{default} error (g) λ_{best} orbit (h) λ_{best} error

DR



(i) λ_{default} orbit (j) λ_{default} error (k) λ_{best} orbit (l) λ_{best} error

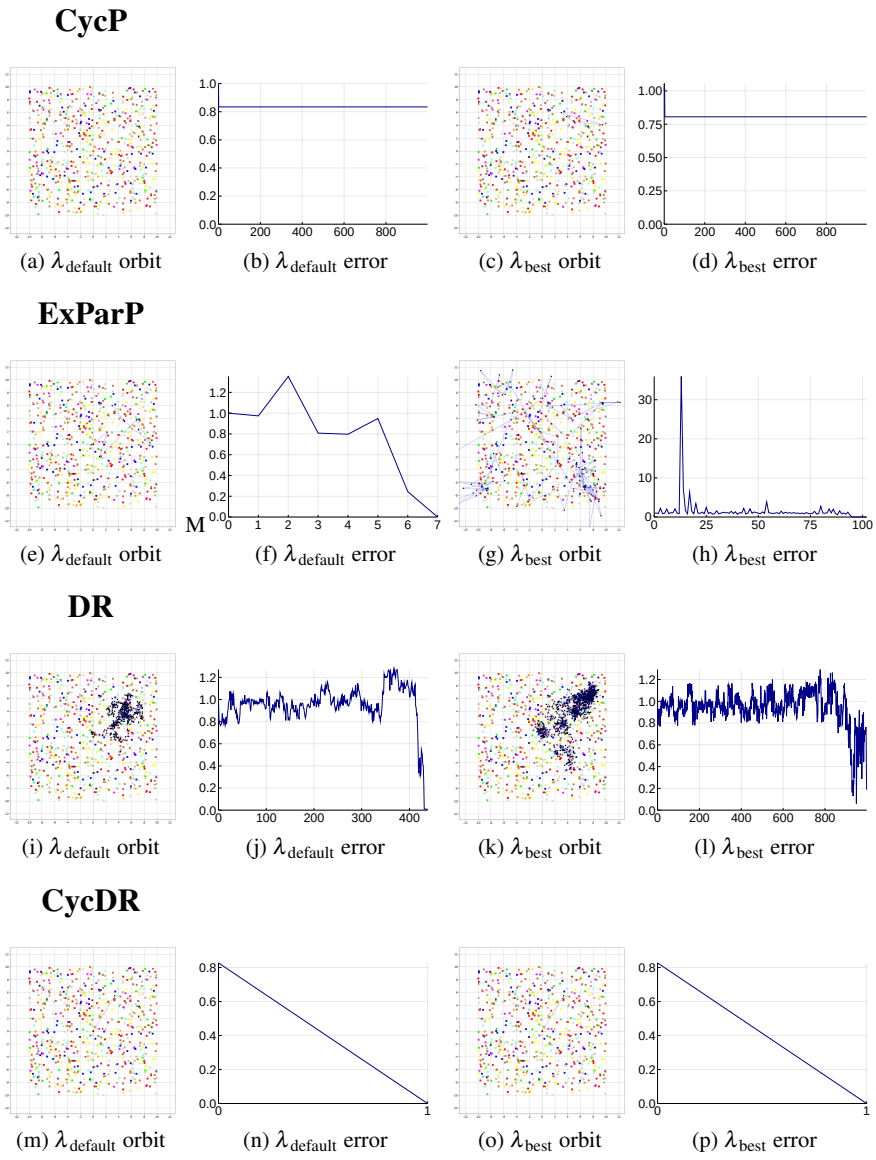
CycDR



(m) λ_{default} orbit (n) λ_{default} error (o) λ_{best} orbit (p) λ_{best} error

Fig. 1.10: Orbits and errors for CycP, ExParP, DR, and CycDR in the many sets with few points constellation

1.6.4 Many sets with many points



(a) λ_{default} orbit (b) λ_{default} error (c) λ_{best} orbit (d) λ_{best} error

(e) λ_{default} orbit (f) λ_{default} error (g) λ_{best} orbit (h) λ_{best} error

(i) λ_{default} orbit (j) λ_{default} error (k) λ_{best} orbit (l) λ_{best} error

(m) λ_{default} orbit (n) λ_{default} error (o) λ_{best} orbit (p) λ_{best} error

Fig. 1.11: Orbits and errors for CycP, ExParP, DR, and CycDR in the many sets with many points constellation

1.6.5 Discussion

The numerical results in this subsection suggest the following: The most challenging constellation is the one with few sets and many points. The least challenging constellation is the one with many sets and few points for which all algorithms are successful.

The worst algorithm is CycP. ExParP with λ_{best} solves all four constellations. DR solves all constellations but λ has to be chosen appropriately. CycDR works well with λ_{default} in terms of number of iterations required; however, it was not able to solve the constellation with few sets and many points.

The experiments in this section suggest that (i) ExParP, DR, and CycDR are algorithms worthwhile exploring and that (ii) experimenting with λ may lead to improved convergence.

Because the results in this section feature a *fixed* starting point, we will explore in the next section the four constellations for a *multitude* of starting points.

1.7 Local and global behaviour

In this section, we continue to consider our four constellations (see Section 1.2) which our four algorithms attempt to solve (see Section 1.3).

In contrast to Section 1.6 where we tracked a single orbit, we here illustrate *local* and *global* behaviour of the algorithms for a multitude of starting points, sampled from $[-10,10]\times[-10,10]$ and $[-100,100]\times[-100,100]$, respectively. We do this for $\lambda_{\text{default}}=1$ and for λ_{best} (see the table in Figure 1.7 in Section 1.5);

For each starting point in the given range, these plots display as gray levels the number of iterations the algorithm needed in its attempt to solve the problem represented by the given constellation. Black corresponds to the minimum number of iterations (zero), and white to the maximum number of iterations (1000). The latter is obtained when the algorithm is unsuccessful. Therefore, the darker the image, the better the performance.

To quantitatively assess the performance of each algorithm, success rates are also provided. These are obtained by dividing the number of times the algorithm is successful by the number of starting points used.

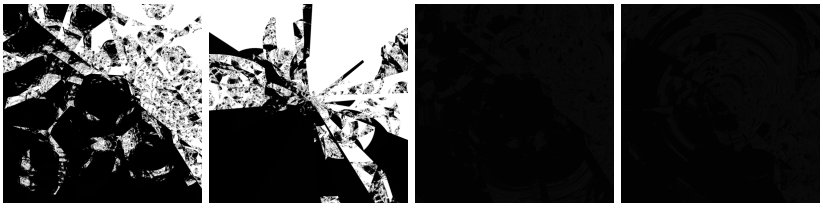
Each of these images was generated using at least 15 million starting points. Depending on the constellation, the time required to generate these pictures ranged between a few minutes to about 3 hours using a quad-core computer.

1.7.1 Few sets with few points

CycP

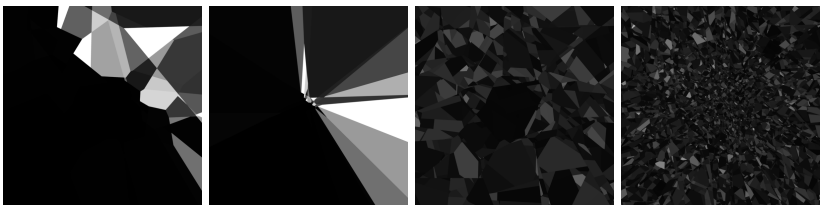
(a) λ_{default} local (57%) (b) λ_{default} global (57%) (c) λ_{best} local (95%) (d) λ_{best} global (98%)

ExParP



(e) λ_{default} local (68%) (f) λ_{default} global (52%) (g) λ_{best} local (100%) (h) λ_{best} global (100%)

DR



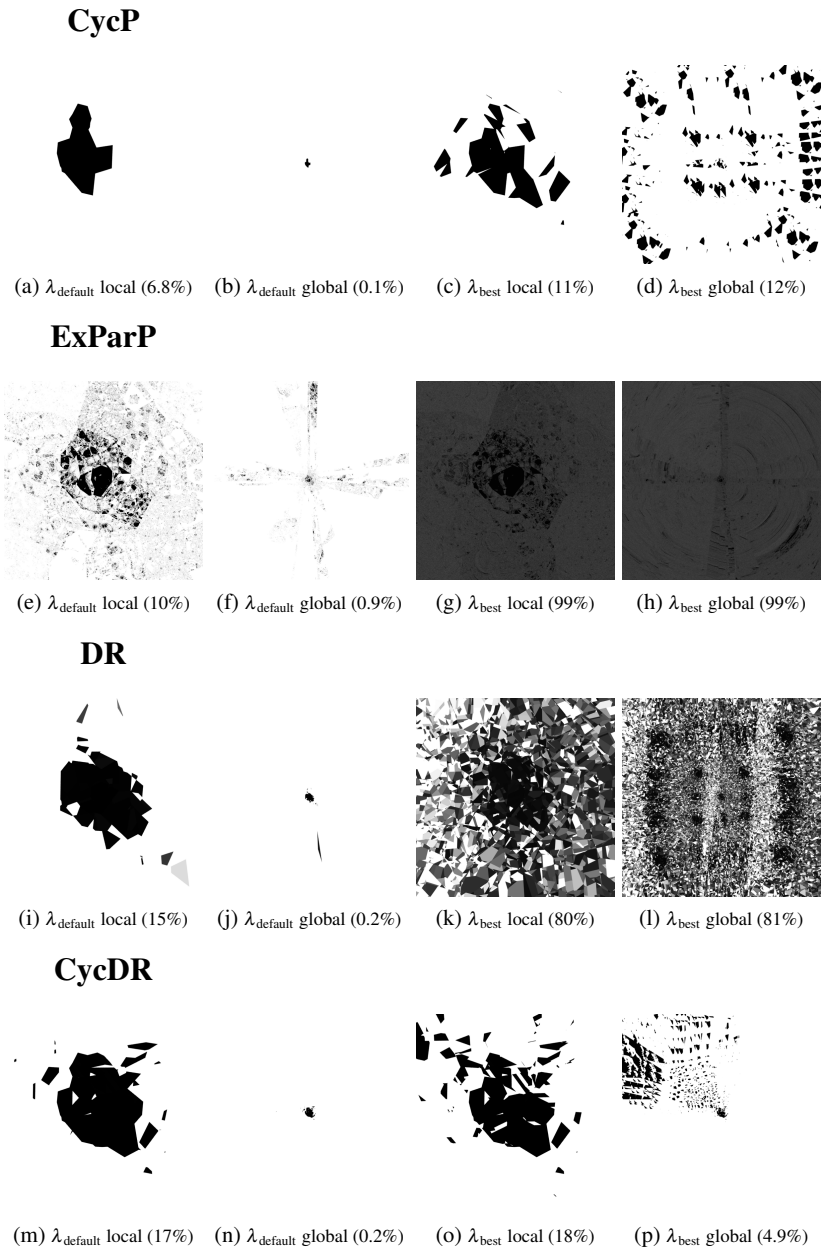
(i) λ_{default} local (96%) (j) λ_{default} global (94%) (k) λ_{best} local (100%) (l) λ_{best} global (100%)

CycDR

(m) λ_{default} local (93%) (n) λ_{default} global (91%) (o) λ_{best} local (100%) (p) λ_{best} global (100%)

Fig. 1.12: Behaviour of CycP, ExParP, DR, and CycDR for the few sets with few points constellation (success rates indicated in parentheses)

1.7.2 Few sets with many points



(a) λ_{default} local (6.8%) (b) λ_{default} global (0.1%) (c) λ_{best} local (11%) (d) λ_{best} global (12%)

(e) λ_{default} local (10%) (f) λ_{default} global (0.9%) (g) λ_{best} local (99%) (h) λ_{best} global (99%)

(i) λ_{default} local (15%) (j) λ_{default} global (0.2%) (k) λ_{best} local (80%) (l) λ_{best} global (81%)

(m) λ_{default} local (17%) (n) λ_{default} global (0.2%) (o) λ_{best} local (18%) (p) λ_{best} global (4.9%)

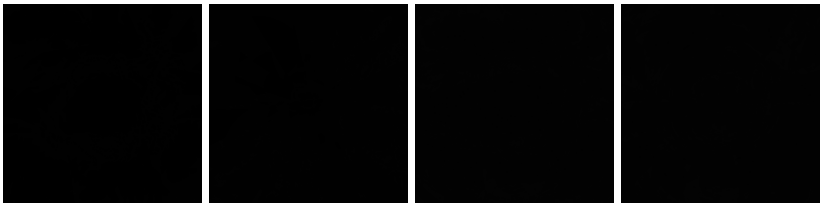
Fig. 1.13: Behaviour of CycP, ExParP, DR, and CycDR for the few sets with many points constellation (success rates indicated in parentheses)

1.7.3 Many sets with few points

CycP

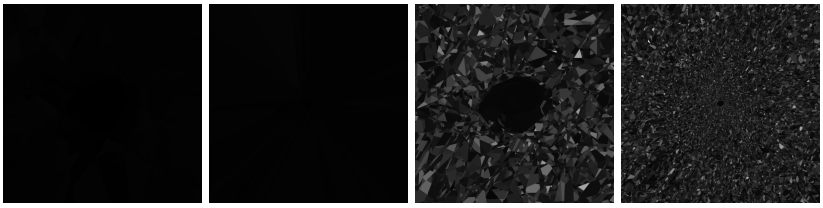
(a) λ_{default} local (100%) (b) λ_{default} global (100%) (c) λ_{best} local (100%) (d) λ_{best} global (100%)

ExParP



(e) λ_{default} local (100%) (f) λ_{default} global (100%) (g) λ_{best} local (100%) (h) λ_{best} global (100%)

DR



(i) λ_{default} local (100%) (j) λ_{default} global (100%) (k) λ_{best} local (100%) (l) λ_{best} global (100%)

CycDR

(m) λ_{default} local (100%) (n) λ_{default} global (100%) (o) λ_{best} local (100%) (p) λ_{best} global (100%)

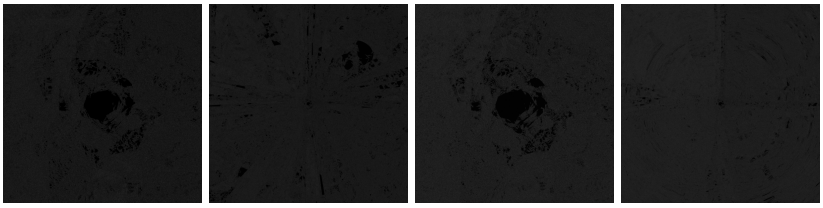
Fig. 1.14: Behaviour of CycP, ExParP, DR, and CycDR for the many sets with few points constellation (success rates indicated in parentheses)

1.7.4 Many sets with many points

CycP

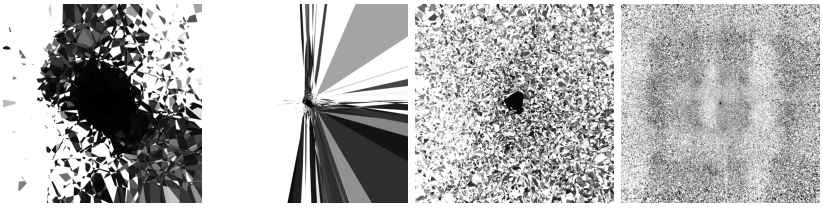
(a) λ_{default} local (24%) (b) λ_{default} global (2.2%) (c) λ_{best} local (38%) (d) λ_{best} global (47%)

ExParP



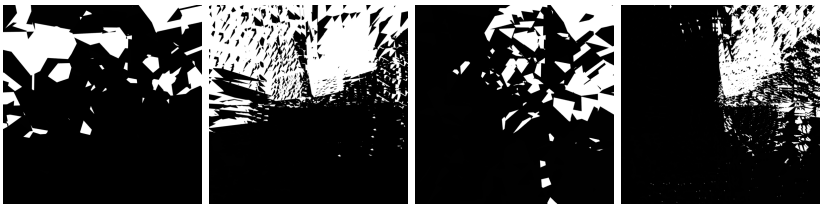
(e) λ_{default} local (100%) (f) λ_{default} global (100%) (g) λ_{best} local (100%) (h) λ_{best} global (100%)

DR



(i) λ_{default} local (53%) (j) λ_{default} global (40%) (k) λ_{best} local (56%) (l) λ_{best} global (57%)

CycDR



(m) λ_{default} local (83%) (n) λ_{default} global (66%) (o) λ_{best} local (84%) (p) λ_{best} global (82%)

Fig. 1.15: Behaviour of CycP, ExParP, DR, and CycDR for the many sets with many points constellation (success rates indicated in parentheses)

1.7.5 Discussion

Comparing the success rates reported in the figures above, it appears that ExParP, DR, and CycDR are good choices; we recommend that CycP be not used. The effect of the tuning parameter λ is very striking for most algorithms when comparing performance of λ_{default} with λ_{best}.

1.8 Divertissements

We experimented also with other constellations and encountered some interesting behaviour of ExParP. This algorithm seems to exhibit fractal-like behaviour for some constellations — whether they are created randomly or not. In the following, we present three images that we found particularly delightful in Figure 1.16 and Figure 1.17.

1.9 Concluding remarks

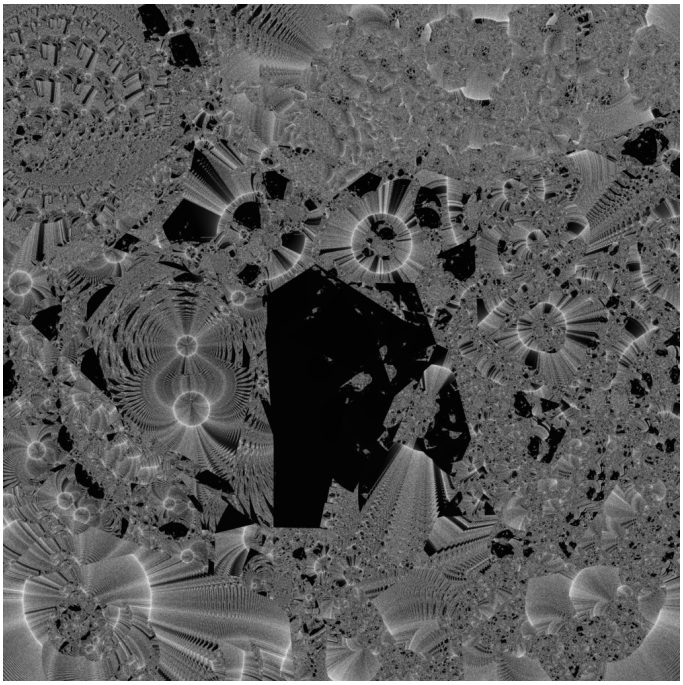
We encountered a somewhat surprising complexity in the behaviour of four algorithms for solving feasibility problems in a simple nonconvex case. The importance of the tuning parameter λ is apparent as is the proximity to solutions (local vs global behaviour). Further studies are needed to find effective guidelines for users in terms of choice of algorithms and the choice of the parameter λ. Finally, and similarly to [8], we encountered *beauty* in our numerical explorations. It is our hope that others will join us and explore theoretically and numerically this fascinating universe of constellations.

Acknowledgements

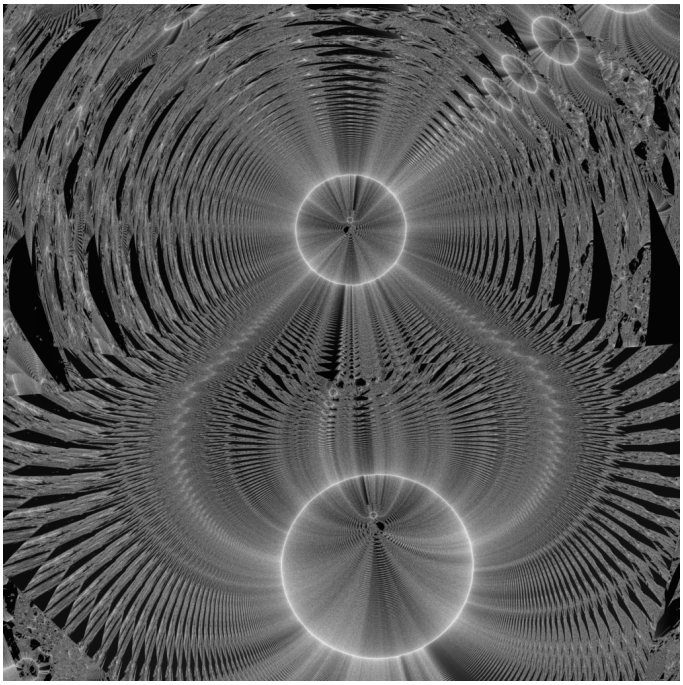
We thank the referee for constructive comments and suggestions. The research of HHB was partially supported by NSERC.

References

1. Bauschke, H.H., Borwein, J.M.: On projection algorithms for solving convex feasibility problems. *SIAM Rev.* **38**, 367–426 (1996)
2. Bauschke, H.H., Combettes, P.L.: *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, second edition. Springer, Cham (2017)



(a)



(b)

Fig. 1.16: Shown in (a) is the performance of ExParP on a constellation consisting of 3 sets with 20 points each, with $\lambda = 0.998$, within the region $[-10,10] \times [-10,10]$. A close-up of the centre-left region of (a) is presented in (b).

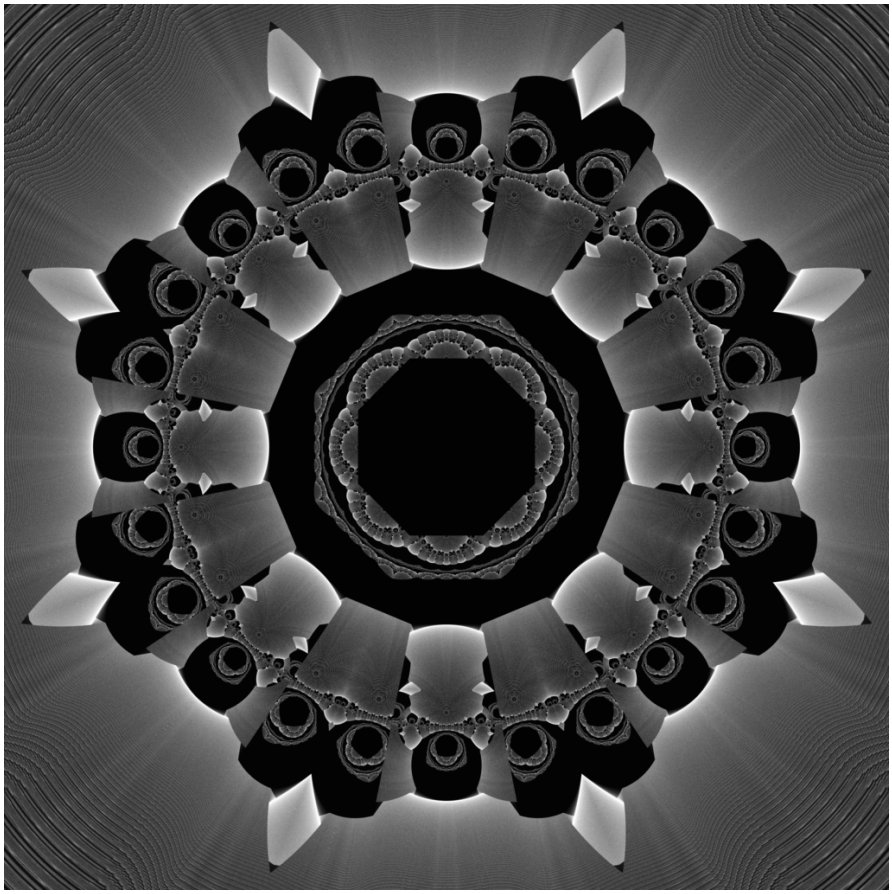


Fig. 1.17: ExParP for a constellation with $\lambda = 0.995$, consisting of 2 subsets of concentric circles centred at the origin, with radii 4 and 8, containing 8 and 16 equispaced points, respectively.

3. Bauschke, H.H., Combettes, P.L., Kruk, S.G.: Extrapolation algorithm for affine-convex feasibility problems. *Numer. Algorithms* **41**, 239–274 (2006)
4. Bauschke, H.H., Koch, V.R.: Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces. *Contemp. Math.* **636**, 1–40 (2015) doi: 10.1090/conm/636/12726
5. Bauschke, H.H., Lindstrom, S.B., Dao, M.N.: The Douglas–Rachford algorithm for a hyperplane and a doubleton. *J. Glob. Optim.*, to appear. `https://arxiv.org/abs/1804.08880` [math.OC] (2018)
6. Bauschke, H.H., Moursi, W.M.: On the Douglas–Rachford algorithm. *Math. Prog. (Ser. A)* **164**, 263–284 (2017)
7. Bauschke, H.H., Noll, D., Phan, H.M.: Linear and strong convergence of algorithms involving averaged nonexpansive operators. *J. Math. Anal. Appl.* **421**, 1–20 (2015)
8. Borwein, J.M., Lindstrom, S.B., Sims, B., Schneider, A., Skerritt, M.P.: Dynamics of the Douglas–Rachford method for ellipses and p-spheres. *Set-Valued Var. Anal.* **26**, 385–403

(2018)
9. Borwein, J.M., Tam, M.K.: A cyclic Douglas–Rachford iteration scheme. *J. Optim. Th. Appl.* **160**, 1–29 (2014)
10. Cegielski, A.: *Iterative methods for fixed point problems in Hilbert spaces*. Springer, Heidelberg (2012)
11. Censor, Y., Chen, W., Combettes, P.L., Davidi, R., Herman, G.T.: On the effectiveness of projection methods for convex feasibility problems Extrapolation algorithm for affine-convex feasibility problems. *Numer. Algorithms* **41**, 239–274 (2006)
12. Censor, Y., Zaknoon, M.: Algorithms and convergence results of projection methods for inconsistent feasibility problems: a review. *Pure Appl. Funct. Anal.*, to appear. https://arxiv.org/abs/1802.07529 [math.OC] (2018)
13. Censor, Y., Zenios, S.A.: *Parallel Optimization*. Oxford University Press (1997)
14. Combettes, P.L.: Convex set theoretic image recovery by extrapolated iterations of parallel subgradient projections. *IEEE Trans. Image Process.* **6**, 493–506 (1997)
15. Combettes, P.L.: Hilbertian convex feasibility problems: convergence of projection methods. *Appl. Math. Optim.* **35**, 311–330 (1997)
16. Dao, M., Phan, H.M.: Linear convergence of the generalized Douglas–Rachford algorithm for feasibility problems. *J. Glob. Optim.* https://doi.org/10.1007/s10898-018-0654-x (2018)
17. Eckstein, J., Bertsekas, D.P.: On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Prog. (Ser. A)* **55**, 293–318 (1992)
18. Elser, V., Rankenburg, I., Thibault, P.: Searching with iterated maps. *Proc. Natl. Acad. Sci. U.S.A.* **104**, 418–423 (2007)
19. Lions, P.-L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.* **16**, 964–979 (1979)
20. Luke, D.R.: Finding best approximation pairs relative to a convex and a prox-regular set in a Hilbert space. *SIAM J. Optim.* **19**, 714–739 (2008)
21. Luke, D.R., Sabach, S., Teboulle, M.: Optimization on spheres: models and proximal algorithms with computational performance comparisons. https://arxiv.org/abs/1810.02893 [math.OC] (2018)