Mathematical Programming Computation
https://doi.org/10.1007/s12532-026-00304-7

FULL LENGTH PAPER

Optimal design of resolvent splitting algorithms

Robert L Bassett[1] · **Peter Barkley**[2]

Received: 3 September 2024 / Accepted: 6 January 2026

Abstract
In this paper, we introduce a novel semidefinite programming framework for designing custom frugal resolvent splitting algorithms which find a zero in the sum of n monotone operators. This framework features a number of design choices which facilitate creating resolvent splitting algorithms with specific communication structure. We illustrate these design choices using a variety of constraint sets and objective functions, as well as the use of a mixed-integer SDP to minimize time per iteration or required number of communications between the resolvents which define the splitting. Using the Performance Estimation Problem (PEP) framework, we provide parameter selections, such as step size, which for high dimensional problems provide optimal contraction factors in the algorithms. Among the algorithm design choices we introduce, we provide a characterization of algorithm designs which provide minimal convergence time given structural properties of the monotone operators, resolvent computation times, and communication latencies.

Mathematics Subject Classification 49M37

1 Introduction

In this paper, we introduce a novel method for designing custom resolvent splitting algorithms to find a zero in the sum of n maximal monotone operators. The algorithms can be described using only vector addition, scalar multiplication, and the resolvents of each operator. The flexible framework we introduce allows the creation of decentralized, distributed resolvent splitting algorithms by customizing the flow of information between the subproblems which compute each resolvent. This permits customized

✉ Robert L Bassett
robert.bassett@nps.edu

Peter Barkley
pbarkley@usna.edu

[1] Operations Research and Applied Mathematics Departments, Naval Postgraduate School, 1 University Circle, Monterey, CA 93943, USA

[2] Mathematics Department, United States Naval Academy, 572C Holloway Road, Annapolis, MD 21402, USA

Published online: 26 February 2026

splitting algorithms designed for particular applications where it may be necessary to specify the communication structure of the algorithms to facilitate computational efficiency or respect real-world constraints.

The work presented here builds on recent work in n-operator splittings. Motivated by the success of the Alternating Direction Method of Multipliers (ADMM), a proximal splitting method for minimizing the sum of two convex functions, researchers tried to extend the framework to the sum of n convex functions after it was demonstrated that the direct extension of ADMM to 3 convex functions does not converge [10]. Despite this initial discouraging result, recent work on resolvent splitting methods for monotone inclusion provides reason for optimism. In [31], Ryu showed that Douglas-Rachford Splitting, the resolvent splitting method underlying ADMM, is unique among splittings satisfying a set of mild assumptions unless the original problem is *lifted* by embedding it in a higher dimensional space. Working in this lifted space, Ryu provided a resolvent splitting algorithm for monotone inclusion problems which are the sum of three maximal monotone operators. This algorithm has provably minimal lifting, and is also *frugal*, in that it evaluates the resolvent of each operator only once per iteration. Shortly thereafter, Malitsky and Tam [28] provided a minimally lifted frugal resolvent splitting algorithm for monotone inclusion problems which are the sum of n maximal monotone operators. Tam [34] followed up on this work by generalizing the example provided in [28] to a family of resolvent splitting algorithms with minimal lifting, as do the authors of [7]. All of these works note the potential for decentralized optimization using these methods, and the work in [34] demonstrates their ability to outperform other decentralized methods like P-EXTRA [33].

A number of other recent works also analyze resolvent splitting methods over n maximal monotone operators, including several which allow decentralized execution. Many of these methods ([13, 14, 26]) rely on a product space formulation which lifts beyond the minimal lifting threshold or rely on a single type of communication structure between resolvents ([9, 27]). Both [7] and [2] build algorithms with minimal lifting over connected graphs using the preconditioner analysis in [6], but do not consider the design approach provided in the following analysis. This parametrization is in some ways prefigured by [18], which establishes a class of projective splitting methods for n operators which includes the option of using resolvent values computed in the previous and the current iterations via a lifted positive definite matrix parameter.

This paper shares a common n-operator frugal resolvent splitting theme with these recent articles. We provide a practical resolvent splitting framework capable of designing algorithms for particular applications. Our primary contribution is the introduction of a family of semidefinite programming problems whose solutions give n-operator resolvent splitting algorithms. This family of semidefinite programming problems can be customized for a variety of practical considerations. In practice, desirable characteristics of a resolvent splitting method can depend on many different aspects of a problem—such as communication latency, convergence rate, per-iteration computation, and how amenable the algorithm is to parallelization. We examine each of these in turn, providing theoretical justification and methodological recommendations for design decisions frequently encountered in practice. In addition to providing a framework for creating resolvent splitting algorithms, we provide several theorems which characterize them, extending and generalizing results in [28, 31, 34]. We also apply

the Performance Estimation Problem (PEP) framework from [16], [32], and [22] to provide convergence rate guarantees for these algorithms, and apply our optimization framework to design algorithms which minimize the total time required to converge for a given problem class. We then use the dual of the PEP to determine the optimal step size and other parameters for a given algorithm design and set of assumptions on the monotone operators, in addition to conducting a variety of numerical experiments to provide convergence rates for specific problems, compare formulation choices, and provide a demonstration of algorithm design choices. An accompanying implementation of our contributions can be found at github.com/peterbarkley/oars/.

The rest of this paper proceeds as follows. In the next section, we introduce the required notation and background. In Sect. 3 we formulate the family of semidefinite programming problems which generate custom resolvent splitting algorithms. Sections 4 and 5 then provide a large set of examples for generating resolvent splitting algorithms under various practical considerations using the constraints and objective function of the SDP, respectively. Section 6 then extends the SDP to a mixed integer setting in which we can minimize the required number of communications and the iteration time of the algorithm designs. Section 7 describes a set of PEP formulations and their duals which provide convergence rate results and the ability to optimize step size and a portion of the algorithm for specific problem classes. Section 8 then conducts a set of numerical experiments, in which we use the previous results to determine the rate of convergence under various assumptions, and we compare the resulting splitting algorithms to those proposed by [28] and [34]. Theorems and lemmas not proven in the main document are postponed to the appendix.

2 Preliminaries

Consider n proper closed convex functions $f_1, \dots, f_n$ where each $f_i : \mathcal{H} \to \mathbb{R} \cup \{\infty\}$ for some real Hilbert space $\mathcal{H}$ that contains the problem's decision variable. Our goal is to solve the problem

$$\min_{x \in \mathcal{H}} \sum_{i=1}^{n} f_i(x) \tag{1}$$

using a *proximal splitting algorithm*, i.e. only interacting with each of the f_i through the proximal operators $\text{prox}_{f_1}, \dots, \text{prox}_{f_n}$ and linear combinations thereof, where

$$\text{prox}_{f_i}(x) := \underset{w \in \mathcal{H}}{\text{argmin}} \; f_i(w) + \frac{1}{2}\|w - x\|^2. \tag{2}$$

If $\mathcal{H}$ is finite dimensional and the relative interiors of the domain of each f_i have nonempty intersection, then finding $x \in \mathcal{H}$ such that

$$0 \in \sum_{i=1}^{n} \partial f_i(x), \tag{3}$$

where ∂f_i denotes the subdifferential of the convex function f_i, is necessary and sufficient for minimizing (1) [29, Theorems 23.8 and 23.5]. Similar regularity conditions

over the strong relative interior allow the application of Fermat's Rule for non-finite dimensional Hilbert spaces as given in [13].

Because the subdifferential of a proper closed convex function f_i, considered as a set-valued operator $\partial f : \mathcal{H} \rightrightarrows \mathcal{H}$, is a maximal monotone operator, a generalization of (3) is the monotone inclusion problem: for n maximal monotone operators $A_1, \ldots, A_n$, where each $A_i : \mathcal{H} \rightrightarrows \mathcal{H}$,

$$\text{find } x \in \mathcal{H} \text{ such that } 0 \in \sum_{i=1}^{n} A_i(x). \tag{4}$$

Throughout the rest of this paper, we assume that the solution set of (4), denoted $\operatorname{zer}\left(\sum_{i=1}^{n} A_i\right)$, is nonempty. Though subdifferentials of convex functions are maximally monotone, maximally monotone operators are only the subdifferential of some convex function if they have additional structure (see e.g. [30]). For this reason, problem (4), which we focus on in the remainder of the paper, contains problems which cannot be stated in terms of convex functions in (3), despite the fact that minimizing the sum of convex functions is our primary motivation.

A *resolvent splitting* is a generalization of a proximal splitting which solves (4) and is constructed using only scalar multiplication, addition, and the resolvents of each operator A_i. The resolvent of an operator A is defined as $J_A = (\mathrm{Id} + A)^{-1}$. When A is monotone, J_A is single-valued, and when A is maximal monotone J_A has domain $\mathcal{H}$ [3, Proposition 23.7]. A resolvent splitting is a *proximal* splitting when the monotone operators in (4) are subdifferentials of convex functions, in which case the resolvent of the subdifferential ∂f_i is the proximal operator of the convex function f_i. A resolvent splitting is called *frugal* if each resolvent J_{A_i} is used at most once in each iteration. Many existing splitting algorithms can be framed as frugal resolvent splittings. The oldest is Douglas-Rachford splitting [15, 17], where $x \in \operatorname{zer}(A_1 + A_2)$ can be found by choosing $z^0 \in \mathcal{H}$ and $\gamma \in (0, 2)$ and iterating

$$x_1^k = J_{A_1}(z^k) \tag{5a}$$
$$x_2^k = J_{A_2}(2x_1^k - z^k) \tag{5b}$$
$$z^{k+1} = z^k + \gamma(x_2^k - x_1^k). \tag{5c}$$

At a fixed point z^* of this iteration, $J_{A_1}(z^*) = x_1^* = x_2^*$ provides a zero of the sum.

In [31], Ryu develops a frugal splitting over three maximal monotone operators, where for $z_1^0, z_2^0 \in \mathcal{H}$ and $\gamma \in (0, 1)$, $x \in \operatorname{zer}(A_1 + A_2 + A_3)$ can be found by iterating

$$x_1^k = J_{A_1}(z_1^k) \tag{6a}$$
$$x_2^k = J_{A_2}(x_1^k + z_2^k) \tag{6b}$$
$$x_3^k = J_{A_3}(x_1^k - z_1^k + x_2^k - z_2^k) \tag{6c}$$
$$z_1^{k+1} = z_1^k + \gamma(x_3^k - x_1^k) \tag{6d}$$
$$z_2^{k+1} = z_2^k + \gamma(x_3^k - x_2^k). \tag{6e}$$

Similarly, at a fixed point $\mathbf{z}^*$ of (6), $J_{A_1}(z_1^*) = x_1^* = x_2^* = x_3^*$ provides a zero of the monotone inclusion.

In [28], Malitsky and Tam develop a splitting over n operators given by

$$x_1^k = J_{A_1}\left(z_1^k\right) \tag{7a}$$

$$x_i^k = J_{A_i}\left(x_{i-1}^k + z_i^k - z_{i-1}^k\right) \qquad \forall i \in \{2, \ldots, n-1\} \tag{7b}$$

$$x_n^k = J_{A_n}\left(x_1^k + x_{n-1}^k - z_{n-1}^k\right) \tag{7c}$$

$$z_i^{k+1} = z_i^k + \gamma\left(x_{i+1}^k - x_i^k\right) \qquad \forall i \in \{1, \ldots, n-1\}, \tag{7d}$$

where $z_i^0 \in \mathcal{H}$ and γ is similarly permitted to be any value in $(0, 1)$. Each of these fixed point encodings converges unconditionally [1, 27], with weak convergence for the iterates in infinite dimensional spaces.

This recent flurry of n-operator resolvent splitting methods is a result of an observation of the authors in [31] (for three operators) and [28] (for n operators), which introduces a parametrization of frugal resolvent splitting methods. These authors' primary use of this parametrization is to prove a lower bound on the dimension of the vector $\mathbf{z}^k$, the variable updated in each iteration of the frugal resolvent splitting method. By altering the values used in Ryu's parametrization, subsequent authors were able to quickly generate new frugal resolvent splitting methods. In [34], Tam characterizes a large set of these parameters which yield convergent frugal splitting algorithms. A similar extension is provided in [7]. In this paper, our Theorem 1 provides a generalization which demonstrates convergence of an even larger set of these parameters.

Because it will be useful in the remainder, we describe this parametrization next. Since the resolvent splittings are assumed to be frugal, each resolvent is evaluated at most once per iteration. Assume without loss of generality that, in each iteration, the ordering of resolvent evaluation is $J_{A_1}, J_{A_2}, \ldots, J_{A_n}$, so that when $i < j$ the input to resolvent J_{A_i} does not depend on the evaluation of resolvent J_{A_j}. We let $\mathbf{x} = (x_1, \ldots x_n) \in \mathcal{H}^n$ be the concatenated vector of resolvent outputs, and $\mathbf{z} = (z_1, \ldots, z_d) \in \mathcal{H}^d$ be the vector which is updated in each iteration. The three ingredients of one iteration of a frugal resolvent splitting are (a) form y_i, the input to resolvent of A_i, for all $i \in [n]$, (b) evaluate the resolvent of A_i at y_i and (c) update the vector $\mathbf{z}$. At convergence, all x_i will be equal to a solution of (4). Recall that a resolvent splitting can be described using only scalar multiplication, addition, and resolvent evaluation, so these update steps can be written using matrices $B \in \mathbb{R}^{n\times d}$, $T \in \mathbb{R}^{d\times d}$, $M \in \mathbb{R}^{d\times n}$ and lower triangular $L \in \mathbb{R}^{n\times n}$ as follows.

$$y_i^k = \sum_{j=1}^{d} B_{ij}\, z_j^k + \sum_{j\le i} L_{ij}\, x_j^k \qquad \forall i \in \{1, \ldots, n\} \tag{8a}$$

$$x_i^k = J_{A_i}(y_i^k) \qquad \forall i \in \{1, \ldots, n\} \tag{8b}$$

$$z_i^{k+1} = \sum_{j=1}^{d} T_{ij}\, z_j^k + \gamma \sum_{i=1}^{n} M_{ij}\, x_j^k \qquad \forall i \in \{1, \ldots, d\}. \tag{8c}$$

The updates (8) can be written more concisely using matrices constructed via the Kronecker product, which we denote by $\otimes$. Let Id denote the identity operator on $\mathcal{H}$, and define $\mathbf{B} = B \otimes \mathrm{Id}$, $\mathbf{L} = L \otimes \mathrm{Id}$, $\mathbf{T} = T \otimes \mathrm{Id}$, and $\mathbf{M} = M \otimes \mathrm{Id}$. Denote by $\mathbf{A} : \mathcal{H}^n \rightrightarrows \mathcal{H}^n$ the maximal monotone operator formed by applying the monotone operators A_i elementwise, so

$$\mathbf{Ax} = \begin{pmatrix} A_1(x_1) \\ \vdots \\ A_n(x_n) \end{pmatrix}.$$

Then, eliminating the y_i variables for conciseness, the updates (8) can be written in terms of the concatenated vectors $\mathbf{z}$ and $\mathbf{x}$ as follows.

$$\mathbf{x}^k = J_{\mathbf{A}}(\mathbf{Bz}^k + \mathbf{Lx}^k) \tag{9a}$$

$$\mathbf{z}^{k+1} = \mathbf{Tz}^k + \gamma \mathbf{Mx}^k. \tag{9b}$$

In this paper, we focus on a restriction of the very general parametrization in (9) that allows us to prove convergence under a set of mild conditions. Specifying the matrices L, M, and the value γ, we let $\mathbf{B} = -\mathbf{M}^T$, and $\mathbf{T}$ be the identity. The frugal resolvent splitting algorithm in (9) then becomes

$$\mathbf{x}^k = J_{\mathbf{A}}\left(-\mathbf{M}^T\mathbf{z}^k + \mathbf{Lx}^k\right) \tag{10a}$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \gamma \mathbf{Mx}^k. \tag{10b}$$

Each of the aforementioned resolvent splitting algorithms in (5), (6), and (7) can be written in the form (10) with appropriate choice of L, M, and range of γ.

In [34], the author notes that when $d > n$, the dimension of the variable $\mathbf{z}$ in the iteration (10) can be reduced by performing the substitution $\mathbf{v} = -\mathbf{M}^T\mathbf{z}$, defining $\mathbf{W} = \mathbf{M}^T\mathbf{M}$, and multiplying the equation (10b) by $-\mathbf{M}^T$ to obtain the iteration

$$\mathbf{x}^k = J_{\mathbf{A}}\left(\mathbf{v}^k + \mathbf{Lx}^k\right) \tag{11a}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k - \gamma \mathbf{Wx}^k. \tag{11b}$$

In (11), the variable $\mathbf{v}^k \in \mathcal{H}^n$ is updated each iteration, which requires less memory than $\mathbf{z}^k \in \mathcal{H}^d$ when $d > n$. It is worth noting that this substitution implies that $\mathbf{v}^0$ must be in the range of $\mathbf{M}^T$.

In the remainder of the paper, we will derive convergent algorithms for both iterations (10) and (11) under a variety of practical assumptions. We provide a framework to design new resolvent splitting algorithms that accommodate decentralized computation. We also investigate the performance of these algorithms by providing theoretical results and practical recommendations to guide the generation of these algorithms while prioritizing fast convergence rates, amenability to parallelization, and minimal per-iteration computation time.

Before proceeding, we establish some notation that will be useful in the rest of the paper. $\mathbb{S}^n_+$ will be used to represent the set of positive semi-definite matrices of dimension n and $\mathbb{S}^n$ the set of symmetric matrices of dimension n. $\succeq$ provides the Loewner ordering of the set of symmetric matrices, so $Z \succeq W \implies Z - W \in \mathbb{S}^n_+$. Id is the identity in the appropriate space in which it is used. Given a matrix $K \in \mathbb{S}^n$, $\lambda_1(K), \ldots, \lambda_n(K)$ denotes the eigenvalues of K listed in increasing order. $\mathbb{1}$ is a ones vector in the appropriate space. All bold vectors and matrices are lifted, so bolded vectors contain multiple copies of $\mathcal{H}$, and bolded operators operate on these lifted vectors. For example, $\mathbf{M} : \mathcal{H}^n \to \mathcal{H}^d$ is $\mathbf{M} = M \otimes \mathrm{Id}$ for $M \in \mathbb{R}^{d\times n}$. We denote the adjoint of a lifted linear operator $\mathbf{M}$ as $\mathbf{M}^T$, noting that it is also the lifted operator corresponding with M^T. For a natural number n, we denote by $[n]$ the set of natural numbers $\{1, \ldots, n\}$. We denote by $\mathcal{S}^C$ the complement of a set $\mathcal{S}$. $\mathrm{Tr}(\cdot)$ defines the trace of a matrix. $\|\cdot\|$ denotes the Euclidean norm when applied to vectors and the spectral norm when applied to matrices. e_i refers to the unit vector with 1 in position i, and $L_{\cdot i}$ and $L_{i\cdot}$ refer to the ith column and row of L respectively. We denote by ι_C the indicator function on a set C and for a set-valued operator A we denote by $A^{-\mathbb{V}}$ the operator with $A^{-\mathbb{V}}(x) = -A^{-1}(-x)$.

3 Semidefinite program for resolvent splitting design

In this section we formulate an optimization problem that allows one to solve for the matrices M and L in (10), or the matrices W and L if using iteration (11). We will call a specific choice of the parameters (W, Z) a *design* for algorithms (10) and (11), noting that we can form the matrix M as $M^T M = W$ and a lower triangular matrix L as $Z = 2\mathrm{Id} - L - L^T$. We discuss several methods for forming M in Sect. 3.1. By considering various objective functions and additional constraints, we show that we can recover several of the proximal splitting algorithms from the previous section. Our next theorem is the primary contribution of this paper.

Theorem 1 *Let $\phi : \mathbb{S}^n_+ \times \mathbb{S}^n \to (-\infty, \infty]$ be any proper lower semicontinuous function. Let $\gamma \in (0, 1)$, $\mathcal{C} \subseteq \mathbb{S}^n_+ \times \mathbb{S}^n$, $c > 0$, and $\varepsilon \in [0, 2)$. Consider the following semidefinite programming problem,*

$$
\begin{aligned}
\underset{W,Z}{\text{minimize}} \quad & \phi(W, Z) && \text{(12a)}\\
\text{subject to} \quad & W\mathbb{1} = 0 && \text{(12b)}\\
& \lambda_1(W) + \lambda_2(W) \geq c && \text{(12c)}\\
& Z - W \succeq 0 && \text{(12d)}\\
& \mathbb{1}^T Z \mathbb{1} = 0 && \text{(12e)}\\
& \mathrm{diag}(Z) = Z_{11}\mathbb{1} && \text{(12f)}\\
& 2 - \varepsilon \leq Z_{11} \leq 2 + \varepsilon && \text{(12g)}\\
& (W, Z) \in \mathcal{C} && \text{(12h)}\\
& W \in \mathbb{S}^n_+, \ Z \in \mathbb{S}^n. && \text{(12i)}
\end{aligned}
$$

Any solution W, Z to (12) produces a convergent resolvent splitting algorithm design. In both iterations, the sequence $(\mathbf{x}^k)$ converges weakly to a vector for which $x_i = x^*$ for all $i \in [n]$, where $x^* \in \mathcal{H}$ solves the monotone inclusion (4), and the iterates $(\mathbf{z}^k)$ and $(\mathbf{v}^k)$ in (10) and (11) converge weakly to fixed points $\mathbf{z}^*$ and $\mathbf{v}^*$. Moreover, when the operators A_i in (4) are all μ-strongly monotone for some $\mu > 0$, the valid range of γ can be extended to $(0, 1 + 2\mu/\|W\|)$.

We now discuss the implications of Theorem 1, postponing its proof to the appendix.

Theorem 1 provides particular insight in the case of μ-strongly monotone operators. When all operators are μ-strongly monotone (as is the case if they are the subdifferentials of μ-strongly convex functions), the permissible range of γ extends beyond one (the bound established in [28, 34]). In Sect. 7, where we provide a technique for selecting a γ which yields the best contraction factor for certain classes of monotone operators, we observe that γ should often be chosen to be greater than or equal to one.

Two important aspects of problem (12) worth addressing are its feasibility and its computational difficulty. When the set $\mathcal{C}$ and the function ϕ are convex, problem (12) is convex. Excepting the general purpose objective ϕ and constraint set $\mathcal{C}$, the problem can be written using only affine and semidefinite cone constraints. Constraint (12c) is perhaps the only constraint that is not immediately recognizable as convex, but we recall that the sum of the k smallest eigenvalues of a PSD matrix is a concave function. Constraint (12c) can be written using the semidefinite cone by introducing auxiliary variables $s \geq 0$ and $Y \in \mathbb{S}^n_+$ and using the constraints

$$W + Y - s\mathrm{I} \succeq 0$$
$$2s - \mathrm{Tr}(Y) \geq c,$$

see e.g. [4]. Therefore, given appropriate choice of objective function ϕ and additional constraint set $\mathcal{C}$, problem (12) is a semidefinite program which, when feasible, can be solved at scale using a variety of possible algorithms [35, 36]. When the constraint set $\mathcal{C}$ is $\mathbb{S}^n_+ \times \mathbb{S}^n$ and c is chosen small enough, (12) is feasible because $W = Z = \frac{2}{n}\left(n\mathrm{Id} - \mathbb{1}\mathbb{1}^T\right)$ lies in its feasible region. In general, it is possible to choose a set $\mathcal{C}$ and constant c that makes (12) infeasible. We address this possibility in Sect. 3.2, where we provide a recommendation for a default value of c, and in Lemma 1, where we provide a set of necessary conditions that can be used to guide the selection of the constraint set $\mathcal{C}$.

One crucial aspect of problem (12) is that constraints (12b) and (12c) combine to define the null space of W exactly. Because $W\mathbb{1} = 0$, we have $\mathrm{span}(\mathbb{1}) \subseteq \mathrm{Null}(W)$ and $\lambda_1 = 0$. Constraint (12c) then gives that $\lambda_2 > 0$, so that the dimension of the eigenspace with eigenvalue 0 is 1 and $\mathrm{span}(\mathbb{1}) = \mathrm{Null}(W)$. Constructing W so that $\mathrm{Null}(W) = \mathrm{span}(\mathbb{1})$ is an important part of the proof of Theorem 1. Constraint (12d) implies $Z \succeq 0$. Constraint (12e) ensures $\mathbb{1}^T L \mathbb{1} = n$. Constraints (12b)-(12e) taken together imply that $\mathrm{Null}(Z) = \mathrm{span}(\mathbb{1})$. Constraints (12f)-(12g) give that Z has a constant diagonal with value strictly between 0 and 4.

Utilizing the matrices generated by solving (12) in the iterations (10) or (11) requires solving a fixed point equation in $\mathbf{x}$ in each iteration, equations (10a) and (11a), respectively. The lower triangular nature of L is a critical part of this computation, and makes

solving for $\mathbf{x}$ straightforward. When L is *strictly* lower triangular, $\mathbf{x}$ can be easily found by sequentially computing $\mathbf{x}_i = J_{A_i}(\mathbf{v}_i^k + \sum_{j<i} L_{ij}\mathbf{x}_j)$ for each $i \in [n]$. When L is lower triangular but not strictly so and $L_{ii} \neq 1$, the elementwise fixed point equation can be written

$$\mathbf{x}_i = J_{A_i}\left(\mathbf{v}_i^k + \sum_{j<i} L_{ij}\mathbf{x}_j + L_{ii}\mathbf{x}_i\right). \tag{13}$$

Manipulating this expression using the definition of the resolvent, this fixed point equation is satisfied if and only if

$$\mathbf{x}_i = J_{\frac{1}{1-L_{ii}}A_i}\left(\frac{1}{1-L_{ii}}\mathbf{v}_i^k + \sum_{j<i}\frac{L_{ij}}{1-L_{ii}}\mathbf{x}_j\right). \tag{14}$$

Since (12) requires L_{ii} to be constant and less than 1, $\sum \frac{1}{1-L_{ii}}A_i(x) = 0 \implies \sum A_i(x) = 0$. In this way, the fixed point equation can again be solved directly as in the strictly lower triangular case, with the exception that the scaled resolvent must now be evaluated on a scaled version of the original input. This is equivalent to executing (13) with $\tilde{\mathbf{A}} = \frac{1}{1-L_{ii}}\mathbf{A}$, $\tilde{W} = \frac{1}{1-L_{ii}}W$ and $\tilde{L} = \frac{1}{1-L_{ii}}(L - L_{ii}\mathrm{Id})$. Therefore when L is lower triangular without being strictly so, the updates (10a) and (11a) can still be easily computed.

In addition to the convergence of the $\mathbf{x}$ iterates in (11), the $\mathbf{v}$ iterates can be used to compute the Attouch-Théra dual solution of a lifted version of the problem (4). In the following theorem, we denote by Δ the subspace

$$\Delta = \{\mathbf{x} \in \mathcal{H}^n \,|\, \mathbf{x} = \mathbb{1} \otimes x \text{ for some } x \in \mathcal{H}\}.$$

Theorem 2 *Let* $\mathbf{v}^*$ *and* $\mathbf{x}^*$ *be limits of the algorithm* (11). *Define* $\mathbf{u}^* = \mathbf{v}^* + (\mathbf{L} - \mathrm{Id})\mathbf{x}^*$. *Then* $\mathbf{u}^*$ *is the solution to the Attouch-Théra dual for the problem*

$$\operatorname*{Find}_{\mathbf{x}\in\mathcal{H}^n} 0 \in (\mathbf{A} + \partial\iota_\Delta)\,\mathbf{x}, \tag{15}$$

which is,

$$\operatorname*{Find}_{\mathbf{u}\in\mathcal{H}^n} 0 \in \left(\mathbf{A}^{-1} + (\partial\iota_\Delta)^{-\mathbb{O}}\right)\mathbf{u}. \tag{16}$$

The duality result in Theorem 2 is especially useful in the context of warmstarting. With any prior knowledge of values $\mathbf{x}$ or $\mathbf{u}$ close to $\mathbf{x}^*$ or $\mathbf{u}^*$, one can warm start the algorithm (11) with $\mathbf{v}^0 = \mathbf{u} + (\mathrm{Id} - \mathbf{L})\mathbf{x}$. If $A_i = \partial f_i$ for some set of closed, convex, and proper functions f_i, a similar result holds for Fenchel and Lagrangian duality in iteration (10). In this case, one can show, for $M \in \mathbb{R}^{n-1\times n}$ and $\mathbf{u}^* \in \mathcal{H}^{n-1}$, that $\mathbf{u}^* = \mathbf{z}^* - (\mathbf{M}^T)^\dagger(\mathbf{L} - \mathrm{Id})\mathbf{x}^*$ is the dual solution of the Lagrangian $g(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^n f_i(x_i) + \langle \mathbf{u}, \mathbf{M}\mathbf{x}\rangle$, where $(\mathbf{M}^T)^\dagger$ denotes the lifted Moore-Penrose pseudoinverse of M^T.

3.1 Constructing M from W

The iteration (10) relies on the construction of a matrix $M \in \mathbb{R}^{d\times n}$ from $W \in \mathbb{R}^{n\times n}$ such that $M^T M = W$. Theorem 1 shows that any such M produces an iteration (10) that converges to a solution of (4). How should one construct the matrix M? Different choices produce different values of d and different sparsity patterns in M, which may be beneficial in different scenarios. In this section, we propose three different methods for constructing M, each of which provides a different tradeoff between sparsity and the leading dimension of M.

Our first method prioritizes sparsity. Assume that W is a *Stieltjes matrix*, a symmetric positive semidefinite matrix with nonpositive off-diagonal entries. This can be guaranteed by adding additional nonpositivity constraints for the off-diagonal entries of W to $\mathcal{C}$, which preserves any existing convexity structure in the problem. Empirically, we note that solutions to (12) often return a Stieltjes W without introducing any additional constraints enforcing the condition. Let $\mathcal{N} = \big((i, j) : j > i \text{ and } W_{ij} \neq 0\big)$ be a tuple containing the nonzero entries in the strictly upper triangular part of W, ordered arbitrarily. Let $d = |\mathcal{N}|$. Define $B \in \mathbb{R}^{d\times n}$ as

$$B_{ik} = \begin{cases} -1 & \text{if } \mathcal{N}_k = (i, j) \text{ for some } i \in [n] \\ 1 & \text{if } \mathcal{N}_k = (j, i) \text{ for some } j \in [n] \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Define $V \in \mathbb{R}^{d\times d}$ to be a diagonal matrix with $V_{kk} = -W_{ij}$ where $\mathcal{N}_k = (i, j)$. Note that the Stieltjes assumption on W gives that V has nonnegative entries. We claim that $BVB^T = W$, which implies that setting $M = V^{1/2}B^T$ gives $M^T M = W$.

To show $M^T M = W$, we consider the off-diagonal and diagonal terms of W separately. For off-diagonal entries, the inner product of columns i and j in M is equal to W_{ij} because the index k is the only nonzero entry in both vectors. For diagonal terms, the ith entry along the diagonal of $M^T M$ is the sum of squares of the entries of column i in M, $-\sum_{j\neq i} W_{ij}$. Constraint (12b) gives that $-\sum_{j\neq i} W_{ij} = W_{ii}$, so diagonal entries of $M^T M$ also equal diagonal entries of W. This method aggressively prioritizes sparsity at the cost of increasing d to the number of nonzero entries in the strictly upper triangular part of W. Additionally, it imposes an additional restriction (the Stieltjes condition) on the set of feasible W.

Another option, which prioritizes both sparsity and small value of d, is to construct the matrix M via Cholesky decomposition, where $W = M^T M$ and M is upper triangular. Additional sparsity can be imposed by performing a sparse Cholesky decomposition which results in matrices P, L, and D such that

$$W = PLDL^T P^T.$$

The matrix $P \in \mathbb{R}^{n\times n}$ is a permutation matrix chosen to minimize fill in, L is lower triangular with unit diagonal, and D is a diagonal matrix. Since W is positive semidefinite, the diagonal matrix D has nonnegative entries. Additionally, since $\text{rank}(W) = n - 1$, exactly one of D's diagonal entries is zero. If the zero occurs in the ith entry of the

diagonal, denote by $\tilde{D}$ the $n-1 \times n-1$ matrix with the ith row and column removed. Likewise, denote by B the matrix PL and by $\tilde{B}$ the matrix B with its ith column removed. Then

$$W = \tilde{B}\tilde{D}\tilde{B}^T,$$

so taking $M = \tilde{D}^{1/2}\tilde{B}^T$ results in a sparse $M \in \mathbb{R}^{n-1\times n}$. Note that, in this sparse Cholesky decomposition, the matrix M is upper triangularizable via a permutation of its columns, but may not actually be upper triangular. In this construction the dimension d of the iteration (10) is $n-1$. Relative to the decomposition in the previous paragraph, the sparse Cholesky decomposition has a smaller d value and does not require the Stieltjes condition, but may be less sparse.

A third method for constructing M uses the eigendecomposition of W. Let $U\Lambda U^T$ be an eigendecomposition of W, where Λ is a diagonal matrix containing the (nonnegative) eigenvalues of W. Since $\text{Null}(W) = \text{span}\{\mathbb{1}\}$, exactly one eigenvalue is equal to 0. Let $\tilde{\Lambda} \in \mathbb{R}^{(n-1)\times(n-1)}$ be the matrix Λ with the zero eigenvalue removed, and $\tilde{U} \in \mathbb{R}^{n\times(n-1)}$ be the submatrix of U which removes the column corresponding to the zero eigenvalue. Then $W = \tilde{U}\,\tilde{\Lambda}\,\tilde{U}^T$, so taking $M = \tilde{\Lambda}^{1/2}\,\tilde{U}^T$ yields an M with $d = n-1$. Though this method for constructing M has $d = n-1$ as in the sparse Cholesky method, it does not prioritize sparsity.

Regarding the dimension d of the iteration (10), [28, Theorem 1] shows that $d \geq n-1$ is necessary for convergence of the algorithm (10). For any W, we have provided two different methods (the Cholesky and eigendecomposition methods) which attain that lower bound. This allows us to prove that every sequence of possible x values generated by an iteration of the form (10) can be generated by an iteration of minimal dimension. We formalize this result in Theorem 3, deferring its proof to the appendix.

Theorem 3 *Every frugal resolvent splitting given by iteration (10) has an equivalent frugal resolvent splitting with minimal lifting. That is, for any $M \in \mathbb{R}^{d\times n}$ and $L \in \mathbb{R}^{n\times n}$ for which $W = M^TM$ and $Z = 2I - L - L^T$ are feasible in (12) for some constants c and ϵ and set $\mathcal{C}$, there exists $\tilde{M} \in \mathbb{R}^{n-1\times n}$ such that for any initial point $\mathbf{z}^0$ there is an initial point $\tilde{\mathbf{z}}^0$ for which the iterations*

$$\mathbf{x}^k = J_{\mathbf{A}}\left(-\mathbf{M}^T\mathbf{z}^k + \mathbf{L}\mathbf{x}^k\right) \qquad \textit{and} \qquad \mathbf{x}^k = J_{\mathbf{A}}\left(-\tilde{\mathbf{M}}^T\tilde{\mathbf{z}}^k + \mathbf{L}\mathbf{x}^k\right)$$
$$\mathbf{z}^{k+1} = \mathbf{z}^k + \gamma\mathbf{M}\mathbf{x}^k \qquad\qquad \tilde{\mathbf{z}}^{k+1} = \tilde{\mathbf{z}}^k + \gamma\tilde{\mathbf{M}}\mathbf{x}^k$$

produce the same sequence $(\mathbf{x}^k)$.

3.2 A graph theoretic interpretation

The language of graph theory provides a helpful description of the sparsity of the matrices in problem (12) and information exchange between resolvent operators. In this section, we recall some concepts from graph theory that will be useful in the remainder. For further details we refer the reader to [11].

A graph $G = (\mathcal{N}, \mathcal{E})$ is a set of nodes $\mathcal{N}$ and a set of edges $\mathcal{E}$ connecting those nodes. The degree of a node is the number of edges containing that node, and the degree matrix of a graph is an $|\mathcal{N}| \times |\mathcal{N}|$ diagonal matrix with the degree of each node on the diagonal. The adjacency matrix A of a graph G is an $|\mathcal{N}| \times |\mathcal{N}|$ matrix

with $A_{ij} = 1$ if there is an edge between node i and node j and 0 otherwise. The Laplacian matrix K of a graph G is $K = D - A$, where D is the degree matrix and A the adjacency matrix of G. An orientation of a graph assigns a direction to each edge; given an oriented graph, its edge-incidence matrix is an $|\mathcal{N}| \times |\mathcal{E}|$ matrix B such that

$$B_{ve} = \begin{cases} 1 & \text{if node } v \text{ is the head of edge } e \\ -1 & \text{if node } v \text{ is the tail of edge } e \\ 0 & \text{otherwise.} \end{cases}$$

The oriented edge-incidence matrix of a connected graph has $\text{Null}(B^T) = \text{span}\{\mathbb{1}\}$. The oriented edge-incidence matrix is connected to the Laplacian matrix K because $K = B\,B^T$. Every Laplacian matrix also necessarily satisfies $K\mathbb{1} = 0$, and for $K \in \mathbb{S}^n_+$, $\lambda_2(K) > 0$ implies the graph is connected.

The Laplacian matrix of a connected graph satisfies the conditions on W in (12b) and (12c) for some $c > 0$, and the connection between the oriented edge-incidence matrix and the Laplacian allows one to take $M = B^T$ in (10) if one can pair it with a feasible Z. In general, finding a feasible matrix Z for a given W is a difficult problem and motivates use of the SDP in (12), though [34] notes in a more limited setting that taking $W = Z$, where W is a scalar multiple of the Laplacian of a connected regular graph, yields a convergent resolvent splitting algorithm.

Extending the graph-theoretic interpretation to the setting of directed graphs with weighted edges allows a graph characterization of iteration (11) for any W and Z. If V is an $|\mathcal{E}| \times |\mathcal{E}|$ diagonal matrix with edge weights on the diagonal, the weighted graph Laplacian is defined as $K = BVB^T$ where B is the (unweighted) oriented edge-incidence matrix. In the remainder, we will denote by $G(K)$ the weighted graph induced by K by interpreting K as a weighted graph Laplacian, i.e. the graph with the edge weight between nodes i and j, where $i \neq j$, given by $-K_{ij}$. Interpreting the W returned by (12) as a weighted graph Laplacian motivates the first M construction in Sect. 3.1. Given constraints (12d) and (12e), any feasible Z in (12) can be shown to be the weighted graph Laplacian of a connected graph as well.

A graph-theoretic interpretation of W provides insight into constraint (12c). Since $W\mathbb{1} = 0$ and $W \succeq 0$, we have $\lambda_1(W) = 0$. $\lambda_2(W)$ is called the *Fiedler value* of the graph $G(W)$. For unweighted graphs, the Fiedler value provides a lower bound on the minimum number of edges which must be removed to form a disconnected graph, and in general graphs with larger Fiedler values have greater connectivity. For an unweighted connected graph on n nodes, the minimal Fiedler value is given by $2\left(1 - \cos\frac{\pi}{n}\right)$ [19]. We use this value as the default choice of the parameter c in (12c), since all connected unweighted graphs, i.e. connected graphs with unit edge weights, are included in the set $\left\{G(W) : \lambda_1(W) + \lambda_2(W) \geq 2\left(1 - \cos\frac{\pi}{n}\right)\right\}$.

Viewing W as the Laplacian of a weighted graph also provides an intuitive interpretation of Malitsky and Tam's Minimal Lifting Theorem, [28, Theorem 1], when the matrix W is Stieltjes. This theorem states that d, the dimension of $\mathbf{z}$ in (10), must be greater than or equal to $n - 1$. If $d < n - 1$, then $W = M^T M$ has rank less than $n - 1$ since $M \in \mathbb{R}^{d\times n}$, so $\lambda_2(W) = 0$, and $G(W)$ is disconnected [20]. Clearly, any algorithm with a disconnected graph purporting to reach consensus among all of its nodes may fail to converge.

When W and Z have no zero entries $G(W)$ and $G(Z)$ are complete weighted graphs. When $W = Z = \frac{2}{n-1}(n\mathrm{Id} - \mathbb{1}\mathbb{1}^T)$ so that the edge weights are constant, we refer to this as the *fully connected* design. We demonstrate empirically in Sect. 8 that the fully connected design frequently offers the best convergence rate for a given problem.

4 Constraint sets

We now turn our attention to the potential uses of the constraint set $\mathcal{C}$. The most direct usages of the constraint set lie in forcing adherence to a given communication pattern and imposing parallelism via block structures. We examine the use of the constraint set for communication patterns first.

4.1 Communication patterns

Consider the problem of designing a customized algorithm for solving a convex program $\min_x \sum_{i=1}^6 f_i(x)$ with a compute cluster organized as in Figure 1, where each resolvent calculation $J_{\partial f_i}$ is computed by the ith machine. Two groups of three machines have low latency connections among themselves, with one of the machines in each group possessing a high latency connection to a machine in the other group. Communications between computers which are not directly connected to one another are not permitted. Such an algorithm can be realized in the context of problem (12) by using $\mathcal{C}$ to constrain the sparsity structure of W and L such that $W_{ij} = Z_{ij} = 0$ equals zero whenever there is no connection between machines i and j. One feasible design is given by

$$W = \begin{pmatrix} 1.86 & -0.52 & -0.52 & -0.83 & 0.00 & 0.00 \\ -0.52 & 1.33 & -0.81 & 0.00 & 0.00 & 0.00 \\ -0.52 & -0.81 & 1.33 & 0.00 & 0.00 & 0.00 \\ -0.83 & 0.00 & 0.00 & 1.86 & -0.52 & -0.52 \\ 0.00 & 0.00 & 0.00 & -0.52 & 1.33 & -0.81 \\ 0.00 & 0.00 & 0.00 & -0.52 & -0.81 & 1.33 \end{pmatrix} \quad Z = \begin{pmatrix} 2.00 & -0.56 & -0.56 & -0.88 & 0.00 & 0.00 \\ -0.56 & 2.00 & -1.44 & 0.00 & 0.00 & 0.00 \\ -0.56 & -1.44 & 2.00 & 0.00 & 0.00 & 0.00 \\ -0.88 & 0.00 & 0.00 & 2.00 & -0.56 & -0.56 \\ 0.00 & 0.00 & 0.00 & -0.56 & 2.00 & -1.44 \\ 0.00 & 0.00 & 0.00 & -0.56 & -1.44 & 2.00 \end{pmatrix}. \tag{18}$$

The matrices in (18) yield convergent algorithms in iteration (11) and (when W is factored as $M^T M$) iteration (10). By constraining the sparsity patterns on these matrices, we construct a decentralized optimization algorithm that respects the communication structure among machines dedicated to computing the proximal values of each function f_i. Note that these matrices are not able to avoid communication along the low latency connection, as evidenced by the nonzero values in W_{14} and Z_{14}. Removing these communications is not possible because doing so partitions the machines into two separate groups that do not communicate with each other via application of $\mathbf{W}$ or $\mathbf{Z}$.

4.2 Parallelism

In general, any off-diagonal nonzero value in Z and W implies a required communication between the resolvents corresponding to its row and column. For any $Z_{ij} \neq 0$

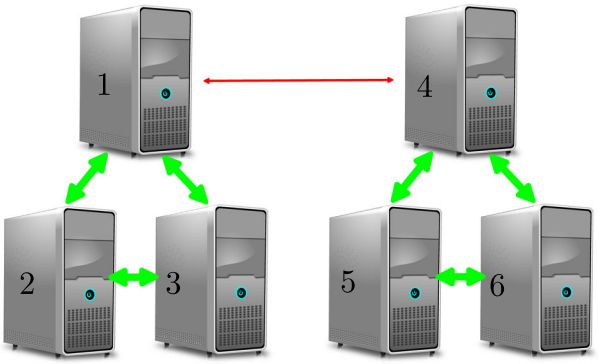


Fig. 1 A hypothetical cluster for which we design a custom splitting algorithm. Red arrows give high latency connections, which should be minimized, and green arrows give low latency connections

where $i < j$ (and $Z_{ji} \neq 0$ by symmetry), resolvent operator i must provide its output to resolvent j within the same iteration. If $W_{ij} \neq 0$, resolvent calculations must be provided by j to i and i to j to supply the information for the subsequent iteration.

By choosing entries of W and Z to be 0, we can design algorithms that allow resolvent calculations to run in parallel, both within a given iteration and across iterations. Within an iteration, if resolvent j does not rely on information from resolvent i ($Z_{ji} = 0$), these resolvents can be computed independently of one another. So, if each has received the required inputs from the other resolvents, resolvents i and j can be executed in parallel. Across iterations, since a given calculation may only depend on a subset of other resolvents to calculate v_i or $\left(\mathbf{M}^T\mathbf{z}\right)_i$, resolvent i can begin the next iteration while resolvent j is still finishing the current iteration if $W_{ij} = 0$.

Figure 2 depicts a Gantt chart which demonstrates the parallel resolvent computation and the execution timeline for the algorithm in (18). For this Gantt chart, we assumed that each resolvent completes in 16 milliseconds, low latency communications (green edges in Figure 1) take .25 milliseconds, and high latency communications (red edges in Figure 1) take 10 milliseconds. In this algorithm, resolvent 4 only depends on resolvent 1 for its input, so as long as it has also received the necessary information from the previous iteration, it can execute its computation in parallel with resolvent 2 or 3. Looking across iterations, resolvent 1 only relies on inputs from resolvents 2, 3, and 4, since W_{15} and W_{16} are 0. Once those resolvents have passed their outputs, resolvent 1 can begin the next iteration, even if resolvents 5 and 6 have not completed their computation.

The parallelism demonstrated in Figure 2 raises the question, can one generate algorithms which achieve maximal parallelism, and therefore minimize the time required to execute some fixed number of iterations? We assume for the moment that all resolvent computation times and communication times are fixed and do not vary across computation nodes, and that each resolvent computation is assigned to a single computation node. In this case, one of the primary approaches to parallelism can be found in designs which execute blocks of calculations in parallel.

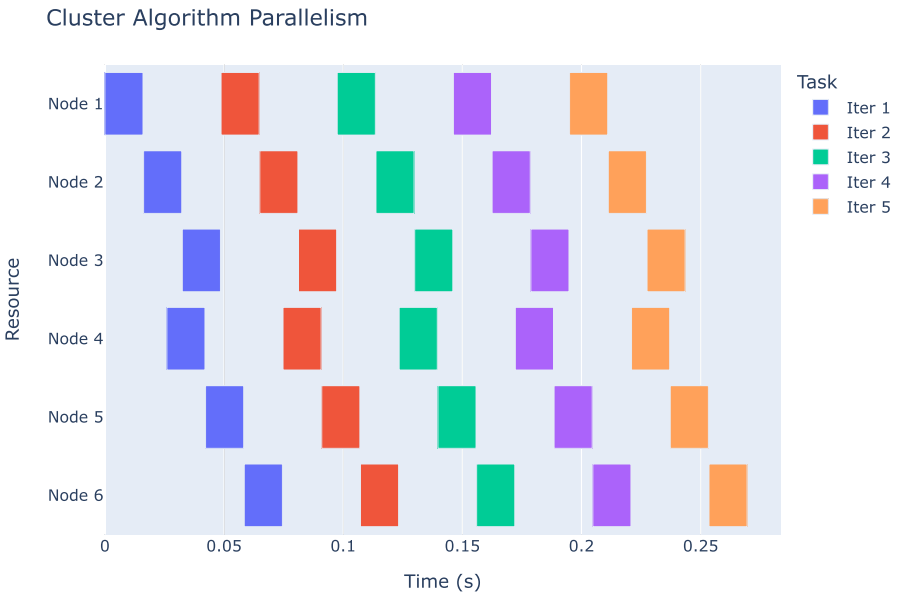


Fig. 2 Execution timeline showing parallelism within and across iterations for the cluster given in Figure 1 using the algorithm specified by (18) and communication and computation times as given in the text

We define a *d-Block* design over n resolvents as follows. Select $d \in \{2, \ldots, n\}$. We construct a partition of the resolvents of size d such that each of the d elements of the partition, which we call a *block*, contains consecutive resolvents. That is, if the blocks are of size $m_1, m_2, \ldots, m_d$, block 1 contains resolvents 1 to m_1, block 2 contains $m_1 + 1$ to $m_1 + m_2$, and so on. For a d-Block design, we prohibit edges in $G(Z)$ connecting two resolvents in the same block and in $G(W)$ we only permit edges between resolvent i in block k and resolvent j in block l if $|k - l| \leq 1$.

Designs which are d-Block impose block matrix structure onto Z and W. Matrices Z and W which form a d-Block design with constant block size have the block matrix form in (19), where each entry is a matrix in $\mathbb{R}^{m\times m}$ where $m = n/d$. Each matrix $D^i \in \mathbb{S}^m$ has diagonals which ensure $W\mathbb{1} = 0$ and off-diagonal elements permitted to be non-zero. 0 and Id are the zero and identity matrices, and $*$ can be varied as long as symmetry is maintained. We present the results corresponding to $\epsilon = 0$ in (12), so each entry on the diagonal of Z equals 2.

$$W = \begin{pmatrix} D^1 & * & 0 & \cdots & 0 & 0 \\ * & D^2 & * & 0 & \ddots & 0 \\ 0 & * & D^3 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & * & 0 \\ 0 & \ddots & 0 & * & D^{n-1} & * \\ 0 & 0 & \cdots & 0 & * & D^n \end{pmatrix}, \quad Z = \begin{pmatrix} 2\mathrm{Id} & * & * & \cdots & * & * \\ * & 2\mathrm{Id} & * & \cdots & * & * \\ * & * & 2\mathrm{Id} & \ddots & * & * \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ * & * & * & \cdots & 2\mathrm{Id} & * \\ * & * & * & \cdots & * & 2\mathrm{Id} \end{pmatrix} \tag{19}$$

In the 2-Block setting with $\epsilon = 0$, we observe an interesting block generalization of Douglas-Rachford splitting. In the Douglas-Rachford splitting for finding a zero in the sum of two monotone operators we have, after rescaling M so that $\gamma \in (0, 1)$,

$$W = Z = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}. \tag{20}$$

The following 2-Block design generalizes Douglas-Rachford for finding a zero in the sum of n operators, where n is divisible by 2, and $m_1 = m_2 = m = n/2$:

$$W = Z = \begin{pmatrix} 2\mathrm{Id} & -\frac{2}{m}\mathbb{1}\mathbb{1}^T \\ -\frac{2}{m}\mathbb{1}\mathbb{1}^T & 2\mathrm{Id} \end{pmatrix}. \tag{21}$$

We also show in Proposition 8 in the appendix that the design in (21) maximizes the sum of second-smallest eigenvalues–the Fiedler values–of Z and W among all 2-block designs with this sparsity pattern. The Douglas-Rachford splitting was extended to n operators in the Malitsky-Tam algorithm [28], and the connection to d-Block designs also extends. When n is divisible by d and blocks are of constant size $m = n/d$, the design

$$Z = \begin{pmatrix} 2\mathrm{Id} & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 0 & \dots & 0 & -\frac{1}{m}\mathbb{1}\mathbb{1}^T \\ -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 2\mathrm{Id} & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & & & 0 \\ 0 & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 2\mathrm{Id} & & & \\ \vdots & & & \ddots & & \vdots \\ 0 & & & & & -\frac{1}{m}\mathbb{1}\mathbb{1}^T \\ -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 0 & & \dots & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 2\mathrm{Id} \end{pmatrix}$$

$$W = \begin{pmatrix} \mathrm{Id} & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 0 & \dots & 0 & 0 \\ -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 2\mathrm{Id} & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & & & 0 \\ 0 & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & 2\mathrm{Id} & & & \\ \vdots & & & \ddots & & \vdots \\ 0 & & & & & -\frac{1}{m}\mathbb{1}\mathbb{1}^T \\ 0 & 0 & & \dots & -\frac{1}{m}\mathbb{1}\mathbb{1}^T & \mathrm{Id} \end{pmatrix}$$

is a d-Block extension of the Malitsky-Tam algorithm. We have observed experimentally that this design also maximizes the sum of the Fiedler values of Z and W among designs which share this sparsity pattern. If instead of fixing the $*$ block matrices in (19) to zero or constant values we allow them to vary, we can generate new algorithms which go beyond the generalized Douglas-Rachford or Malitsky-Tam structure. For example, in Sect. 8, we provide empirical evidence that optimizing for a variety of spectral properties of Z and W across this larger space generates minimum iteration time algorithms with a convergence rate which is invariant to the number of resolvents in some cases. We also find that d-Block maximum Fiedler value designs share the between-block fully connected nature of Z found in the extended Ryu algorithm provided by Tam [34], but with a path graph connecting blocks in W rather than a star.

If we set $d = 2$ for even n, but (unlike the generalization of Douglas-Rachford in (21)) relax the constraint $D_{ij} = 0 \quad \forall i \neq j$, thereby allowing the diagonal blocks of W to vary, we can generate a number of different 2-Block algorithms using objective functions which optimize various spectral properties of W. Section 8 will show that these tend to outperform other block designs with more blocks (and therefore more sparsity).

Care must be taken in the choice of the size of each block to avoid infeasibility in (12). To that end, we note the following lemmas, which provide general guidelines for determining whether a constraint set $\mathcal{C}$ allows a feasible solution to (12).

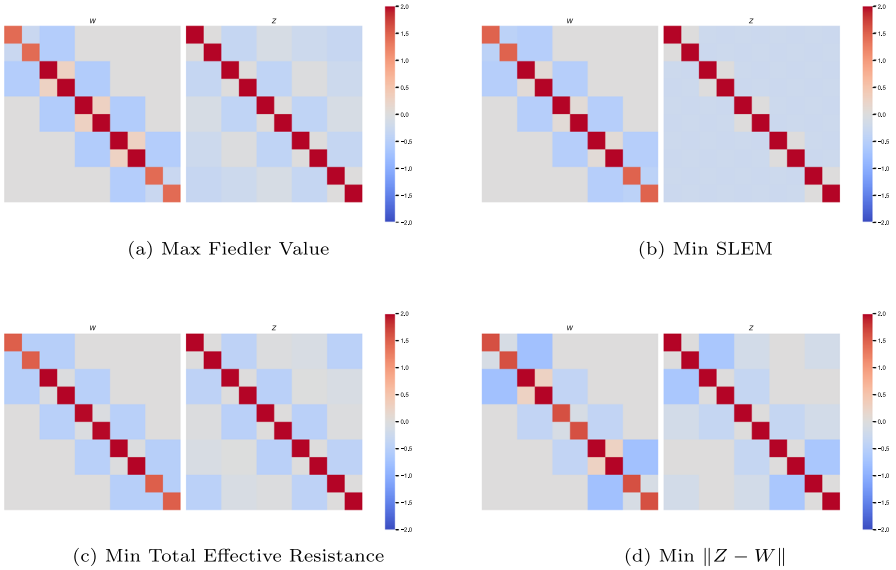
Lemma 1 *For any design* (W, Z) *which satisfies constraints* (12) *we have the following:*

a. All entries in Z *and* W *have magnitude bounded above by* Z_{11}. *Phrased graph-theoretically, edge weights in* $G(Z)$ *and* $G(W)$ *have magnitude bounded above by the (constant) degree of the nodes in* $G(Z)$.
b. $G(W)$ *must be connected.*
c. $G(W)$ *must have at least one edge connected to each node.*
d. $G(W)$ *must have at least* $n - 1$ *edges.*
e. $G(Z)$ *must be connected.*
f. If $n > 2$, $G(Z)$ *must have at least two edges connected to each node.*
g. If $n > 2$, $G(Z)$ *must have at least* n *edges.*
h. For any subset $\mathcal{S}$ *of nodes in* $G(Z)$, *let* $\mathcal{E}_S$ *be the set of edges between nodes in* $\mathcal{S}$. *It then holds that* $||\mathcal{S}| - |\mathcal{S}^C|| \leq 2\left(|\mathcal{E}_S| + |\mathcal{E}_{S^C}|\right)$.

Lemma 1h. is particularly helpful for block design. Note that the definition of a d-Block design means that for any block $\mathcal{S} \subseteq \{1, \ldots, n\}$, $\mathcal{E}_S = \emptyset$ in $G(Z)$. An immediate corollary of Lemma 1h. is that a 2-Block design must have $|\mathcal{S}_1| = |\mathcal{S}_2|$. A 2-Block design is therefore only feasible for even n, and partitions the resolvents into two blocks of size $\frac{n}{2}$. We also note that selecting d which divides n and letting the block size $|\mathcal{S}_i| = \frac{n}{d}$ will always satisfy Lemma 1h. Unless otherwise noted, d-Block designs in the remainder of this work will have a constant block size of $m = \frac{n}{d}$. Given this set of methods for shaping the structure of the algorithm design, we now turn our attention to objective functions which allow us to optimize within a given structure.

5 Objective functions

The objective function in (12a) allows us to select among the feasible matrices defined by constraints (12b)-(12i). We introduce a variety of possible objective functions in this section, beginning with approaches which operate on the spectra of Z and W to target specific graph characteristics and are commonly used as heuristics in the algorithm design process [12]. We then explore a mixed integer framework and a linearization of the problem which allows optimization over the number and placement of edges in addition to edge weights. We use these formulations to design algorithms which minimize the iteration time given any set of computation and communication times.



(a) Max Fiedler Value (b) Min SLEM

(c) Min Total Effective Resistance (d) Min $\|Z - W\|$

Fig. 3 5-Block sparsity pattern over 10 resolvents using different spectral objectives

5.1 Spectral objective functions

Problem (12) admits a number of widely-used convex objective functions which use the spectrum of the graph Laplacian to optimize for specific properties. In this section we introduce several of these objectives, including the algebraic connectivity, the second-largest eigenvalue magnitude (SLEM), the total effective resistance, and the spectral norm of the difference between Z and W. Exact formulations are available in the appendix's Section D.1.

We begin by examining the maximum sum of the Fiedler values for $G(W)$ and $G(Z)$. This can be obtained using the objective function $\phi(W, Z) = -\beta_W(\lambda_1(W) + \lambda_2(W)) - \beta_Z(\lambda_1(Z) + \lambda_2(Z))$, where $\beta_W \geq 0$ and $\beta_Z \geq 0$. In Sect. 3.2, we note that $\lambda_1(W) = 0$, so $\lambda_1(W) + \lambda_2(W)$ gives the Fiedler value of $G(W)$ and provides its algebraic connectivity (and likewise with Z). We also note that this objective function yields a convex problem in (12). In graphs with positive edge weights, selecting edge weights which maximize the Fiedler value has been shown to maximize the mixing rate for a continuous time Markov chain defined over those edges [5]. This has led to its selection as a common heuristic for a number of decentralized algorithms [12]. Figure 3aprovides an example of matrices W and Z formed with the maximum Fiedler value objective function over a 5-Block design on 10 resolvents.

The SLEM provides another popular heuristic for the design of decentralized algorithm mixing matrices. Given a symmetric Markov Chain, the SLEM is known to maximize its rate of convergence to its equilibrium [5]. If $G(W)$ and $G(Z)$ have only positive edge weights, we can form stochastic matrices P_w and P_z by setting $\epsilon = 0$, $P_w = \text{Id} - W/2$ and $P_z = \text{Id} - Z/2$. The SLEM of P_w is then given by $\max\{|1 - \lambda_2(W)|, \lambda_n(W) - 1\}$, and similarly for P_z, giving us $\phi(W, Z) =$

$\beta_Z \max\{|1-\lambda_2(Z)|, \lambda_n(Z)-1\} + \beta_W \max\{|1-\lambda_2(W)|, \lambda_n(W)-1\}$ [5]. This is a convex function of W and Z which can be minimized for either or both, with a trade-off coefficient determining the weight of each. Figure 3bdemonstrates the results of minimizing the SLEM over a 5-Block design on 10 resolvents.

Another popular heuristic is the total effective resistance. For any given weighted graph $G(K)$, the total effective resistance of the graph is defined as $\frac{1}{n}\sum_{i=2}^{n}\frac{1}{\lambda_i(K)}$. For a graph with positive edge weight, the total effective resistance has been shown to minimize the average commute time, over all pairs of vertices, in the random walk on the graph with the transition probability given by the ratio of the edge weights over the node degree for any given node [5]. It is also convex and therefore tractable in (12) with $\phi(W,Z) = \beta_W \frac{1}{n}\sum_{i=2}^{n}\frac{1}{\lambda_i(W)} + \beta_Z \frac{1}{n}\sum_{i=2}^{n}\frac{1}{\lambda_i(Z)}$. Figure 3cdemonstrates the output of this objective. Minimal total effective resistance provides the best convergence rate among the objectives presented here on a wide variety of problems, and demonstrates a convergence rate invariant to problem size in our experiments over any set of n identical l-Lipschitz, μ-strongly monotone operators.

The final spectral objective we consider is $\min \|Z - W\|$ where $\|\cdot\|$ indicates the spectral norm ($\lambda_n(\cdot)$). This objective has the benefit of balancing the inputs ($\mathbf{v}$ or $-\mathbf{M}^T\mathbf{z}$) to a given resolvent with the resolvent inputs ($\mathbf{x}$). For the 2-Block design, $\min \|Z - W\|$ returns $W = Z$, which also minimizes total resistance subject to these constraints. The minimal spectral norm and minimal total resistance designs are not equal in general, however, as seen in Figures 3cand 3d.

These objectives highlight the breadth of design options available in (12). Many more could be developed. Section 8 provides a comparison of the various objectives presented here on different problem classes. The maximum Fiedler value and minimum total effective resistance provide particularly promising worst-case convergence rate results using the PEP framework.

6 Non-convex extensions

A number of valuable algorithmic properties can only be modeled by shifting to a framework which captures whether entries in Z and W are nonzero. We therefore introduce binary variables for each of the off-diagonal entries in Z and W. The sparsity-maximizing formulation we describe in (22) provides the most direct application of these binary variables. The remainder of this subsection describes the use of this mixed integer approach to minimize algorithm iteration time.

Given a particular constraint set $\mathcal{C}$, we can find maximally sparse designs which minimize the number of nonzero entries in W and Z by adding binary variables which track the nonzero entries of these matrices. This can be done with the following mixed integer semidefinite program (MISDP):

$$\begin{aligned} \min_{x,y,Z,W} \quad & \sum_{i=1}^{n}\sum_{j<i} x_{ij} + y_{ij} \\ \text{s.t.} \quad & (2+\epsilon)x_{ij} \geq |Z_{ij}| \quad \forall i, j \in [n] \end{aligned} \tag{22a}$$

$$(2+\epsilon)y_{ij} \geq |W_{ij}| \quad \forall i, j \in [n] \tag{22b}$$

$$\sum_{i=1}^{n} \sum_{j<i} x_{ij} \geq n \tag{22c}$$

$$\sum_{i=1}^{n} \sum_{j<i} y_{ij} \geq n-1 \tag{22d}$$

$$\sum_{j \neq i} x_{ij} \geq 2 \quad \forall i \in [n] \tag{22e}$$

$$\sum_{j \neq i} y_{ij} \geq 1 \quad \forall i \in [n] \tag{22f}$$

$$x \in \mathbb{S}^n, \quad x_{ij} \in \{0,1\} \quad \forall i, j \in [n] \tag{22g}$$

$$y \in \mathbb{S}^n, \quad y_{ij} \in \{0,1\} \quad \forall i, j \in [n] \tag{22h}$$

SDP constraints (12b)-(12i). (22i)

In (22), the objective counts the number of nonzero entries in strictly lower triangular parts of W and Z. Constraints (22a) and (22b) require weights which are not indicated as nonzero to be zero, relying on Lemma 1a. to bound the value of each entry. Constraints (22c)-(22f) utilize Lemma 1 to tighten the formulation; they are optional but reduce the time required by MISDP solvers, e.g. SCIP-SDP [21], to solve the problem. Constraint (22c) implements Lemma 1g., requiring the existence of at least n edges in $G(Z)$. Constraint (22d) implements 1d., requiring the existence of at least $n-1$ edges in $G(W)$. Constraint (22e) ensures each node in $G(Z)$ has at least two edges (required by Lemma 1f.). Lemma 1c. gives that there must be at least one edge to each node in $G(W)$, which is captured in constraint (22f).

6.0.1 Mixed integer formulation

With an additional set of restrictions on Z and W, we can solve this MISDP as a Mixed Integer Linear Program (MILP). We do so by requiring $|Z_{ij}| \geq |W_{ij}|$ and ensuring connectivity in $G(W)$, and by allowing only negative off-diagonal values in W, Z, and $Z-W$. These final conditions are equivalent to requiring only positive edge weights in $G(W)$, $G(Z)$, and $G(Z-W)$. We note that the existing FRS algorithms presented in Sect. 2 all satisfy these restrictions. The following theorem establishes the connection between the SDP and its restricted linear counterpart.

Theorem 4 *Any W and Z satisfying the linear constraints (23a)-(23g) also satisfy the constraints in (12) for some constant $c > 0$ and $\mathcal{C} \subseteq \mathbb{S}^n_+ \times \mathbb{S}^n$.*

$$Z, W \in \mathcal{S}^n \tag{23a}$$

$$Z_{ij} \leq W_{ij} \leq 0 \quad \forall i, j \in [n], i \neq j \tag{23b}$$

$$W\mathbb{1} = 0 \tag{23c}$$

$$2-\epsilon \leq Z_{11} \leq 2+\epsilon \tag{23d}$$

$$\operatorname{diag}(Z) = Z_{11} \tag{23e}$$
$$Z\mathbb{1} = 0 \tag{23f}$$
$$G(W) \text{ is connected.} \tag{23g}$$

There are at least two approaches for linearly constraining W to provide connectivity in $G(W)$ when it has positive edge weights. The first verifies that each subset $\mathcal{S}$ of the graph's nodes has positive edge weight over its cutset (the set of edges with one node in $\mathcal{S}$ and the other in $\mathcal{S}^{\mathcal{C}}$), so that every subset is connected, and the graph is therefore connected. The second verifies a network flow can move from a source node to every other node. The number of constraints in the cutset approach scales exponentially in n, whereas the network flow requires a set of additional continuous variables and constraints which scale quadratically in n. We provide the flow-based formulation here.

The flow-based formulation uses the following additional notation presented below to fix parameters for the source and sink values for the network and define variables for its flow:

$h \in \mathbb{R}^n$: supply/demand parameter placing a supply value of $n-1$ at resolvent 1 ($h_1 = n-1$) and a unit demand value at each of the other resolvents ($h_i = -1$)

$f_{ij} \geq 0 \quad \forall i \neq j$: decision variable for flow from i to j.

We then define the flow-based formulation as

$$\min_{x,y,Z,f} \quad \sum_{i=1}^{n} \sum_{j<i} x_{ij} + y_{ij}$$
$$\text{s.t.} \quad Z_{ij} \leq 0 \quad \forall i \neq j \in [n] \tag{24a}$$
$$y_{ij} \leq x_{ij} \quad \forall i < j \in [n] \tag{24b}$$
$$-Z_{ij} \leq (2+\epsilon) x_{ij} \quad \forall i < j \in [n] \tag{24c}$$
$$-Z_{ij} \geq x_{ij}/(n-1) \quad \forall i < j \in [n] \tag{24d}$$
$$\sum_{j \neq i} f_{ij} - \sum_{j \neq i} f_{ji} = h_i \quad \forall i \in [n] \tag{24e}$$
$$f_{ij} \leq (n-1) y_{ij} \quad \forall i \neq j \in [n] \tag{24f}$$
$$f_{ij} \geq 0 \quad \forall i \neq j \in [n] \tag{24g}$$
$$\text{Cutting Plane Constraints (22c)-(22f)} \tag{24h}$$
$$\text{Binary variable constraints (22g)-(22h)} \tag{24i}$$
$$\text{Linear SDP constraints (12b), (12f).} \tag{24j}$$

This formulation eliminates W as a decision variable, defining it after completion as $W_{ij} = y_{ij} Z_{ij}$ for $i \neq j$, and $W_{ii} = -\sum_{j \neq i} W_{ij}$. Problem (24) uses constraints (24e) and (24f) to require the existence of a path from resolvent 1 to each of the other resolvents by forcing the flow of the $n-1$ value at 1 only via edges allowed in y,

thereby requiring connectivity in y. Constraints (24b) and (24d) link the values of Z_{ij} and x_{ij} to y_{ij}, forcing them to represent connected graphs when y does. The lower bound in (24d) forces an entry of at least $1/(n-1)$ in $-Z_{ij}$ when $x_{ij} = 1$, which is half the value of the weight on the edges of the fully connected design; other small positive values could be used instead. Setting $W_{ij} = y_{ij} Z_{ij}$ then makes $G(W)$ a connected graph. Problem (24) therefore satisfies the requirements of Theorem 4, and guarantees that W and Z satisfy the constraints in (12).

6.0.2 Minimum iteration time

We now consider the use of the MISDP and MIP formulations to generate minimum iteration time algorithms when the computation times for each resolvent and the communication times between computation nodes are arbitrary. We assume in the remainder that all resolvent computation times for a given resolvent are fixed, meaning that they do not vary across iteration or input value. These are given by $t_i > 0$ for each resolvent $i \in [n]$. We similarly assume fixed communication times $l_{ij} = l_{ji} > 0$ for all $i < j \in [n]$. All other calculations in (11) involve scalar arithmetic on co-located data and are assumed negligible. Let s_{ki} be the computation start time of resolvent i in iteration k, s_k be the earliest computation start time across all resolvents in iteration k (the iteration start time), and e_k be the completion time of the final communication in iteration k (the iteration end time). We then have:

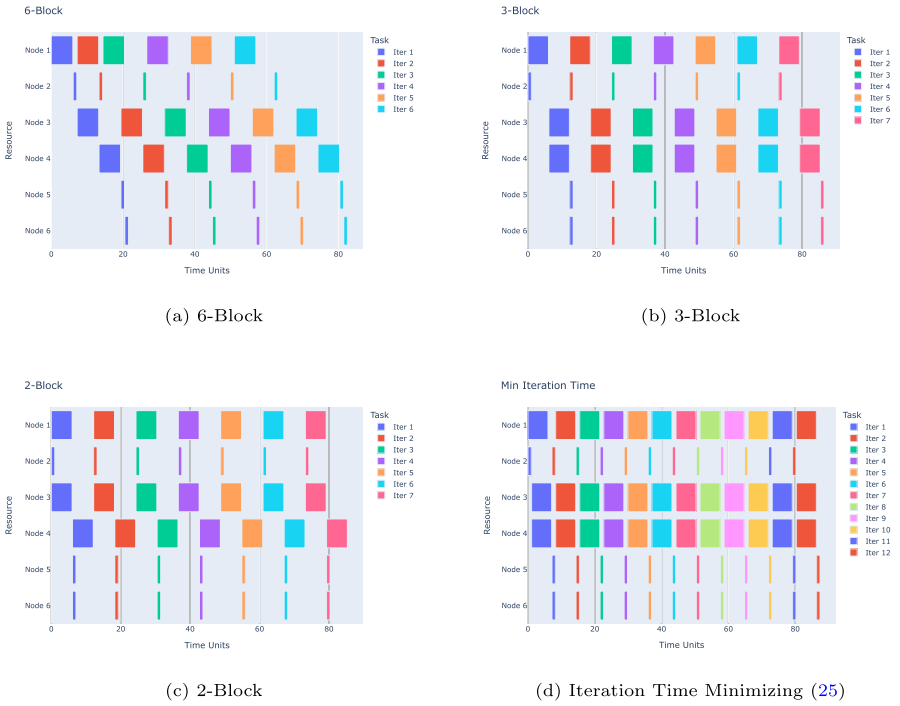
$$
\begin{aligned}
s_{11} &= 0 \\
s_{kj} &= \max_{i | L_{ji} \neq 0} s_{ki} + t_i + l_{ij} && \forall k \in [r],\, j > i \in [n] \\
s_{k+1,i} &= \max_{j | W_{ij} \neq 0} s_{kj} + t_j + l_{ji} && \forall k \in [r],\, i \neq j \in [n] \\
s_k &= \min_{i \in [n]} s_{ki} && \forall k \in [r] \\
e_k &= \max_{i \in [n]} \{ s_{ki} + t_i + \max_{j | W_{ij} \neq 0} l_{ij} \} && \forall k \in [r].
\end{aligned}
$$

We can then define the k-iteration average iteration time as $c^k = e^k/k$. c^1 is the completion time of the first iteration. The following proposition establishes a lower bound on any single iteration time when communication times are constant, which is useful for establishing whether a given design attains this minimum.

Proposition 5 *The minimum single iteration time for algorithm* (11) *with computation time t_i for resolvent $i \in [n]$ and constant communication time l between all resolvents has a lower bound of* $\max_{i \in [n]} t_i + \min_{i \in [n]} t_i + 2l$.

For large numbers of iterations, the limiting behavior of c^k is a useful characterization of the algorithm's performance. We define the *algorithm iteration time*, denoted c^∞, as $c^\infty = \lim_{k \to \infty} c^k$.

The block designs described in Sect. 4 frequently obtain the minimum single iteration time in the limit, but do not do so generally. Consider, for example, the case where communication times are constant, but the resolvent computation times are



(a) 6-Block

(b) 3-Block

(c) 2-Block

(d) Iteration Time Minimizing (25)

Fig. 4 d-Block and Minimum Iteration Time Design Execution Timelines

not. When it is possible to choose the resolvent ordering and block sizes so that the maximum resolvent computation time across one of the two sets of parallel blocks equals $\min_{i\in[n]} t_i$, the minimum iteration time bound in Proposition 5 is attained by this design. One can show that this condition holds in the case of constant communication time among all resolvents and resolvent computation times which are constant except for a single maximal outlier. In this situation, the minimum iteration time bound is attained regardless of the ordering of the resolvents whenever the d-Block design is feasible in (12).

In the general case, however, minimizing the iteration time over arbitrary computation and communication times requires the use of the binary edge constraints provided in (22). We do so with an extension of the critical path method found in project scheduling [25] which minimizes the longest stretch of dependent resolvent calculations over r iterations of the algorithm. Figure 4 demonstrates the utility of this formulation relative to d-Block designs. In this example, we show the algorithm execution time for 6-Block, 3-Block, 2-Block, and minimum iteration designs for a problem with a mix of long and short resolvent computation times. In each of the block designs, the alternating sets of adjacent blocks each have a least one long resolvent computation time, resulting in average iteration times which are almost double the minimum possible. The design produced by our MISDP algorithm clearly outperforms the d-Block designs, completing 12 iterations in the displayed time interval.

We require a set of additional parameters and decision variables given by

$r \geq n$: integer parameter for number of iterations over which to minimize
$t_i \geq 0$: parameter for time to compute resolvent i
$l_{ij} \geq 0$: parameter for time to communicate from i to j
$a > 0$: parameter larger than any possible time difference between resolvent start times to relax inactive constraints
$q > 0$: parameter for a lower bound on the single iteration time
$b \geq 0$: decision variable tracking difference between the run time over r iterations and the lower bound
$s_{ki} \geq 0$: decision variable for time from algorithm start until completion of resolvent calculation i in iteration k.

Our minimum iteration time MISDP is then given by

$$\min_{x,y,b,s,Z,W} \quad b$$

$$\text{s.t.} \quad s_{kj} - s_{ki} \geq (t_i + l_{ij} + a)x_{ij} - a \quad \forall k \in [r], i < j \in [n] \tag{25a}$$

$$s_{k+1,i} - s_{kj} \geq (t_j + l_{ji} + a)y_{ij} - a \quad \forall k \in [r], i < j \in [n] \tag{25b}$$

$$s_{k+1,j} - s_{ki} \geq (t_i + l_{ij} + a)y_{ij} - a \quad \forall k \in [r], i < j \in [n] \tag{25c}$$

$$b \geq \max_{i<j\in[n]} \left\{ s_{ri} + t_i + l_{ij}y_{ij} \right\} - rq \tag{25d}$$

SDP constraints (12b)-(12h) (25e)

Integer bounding constraints (22a)-(22b) (25f)

Binary variable constraints (22g)-(22h) (25g)

Optional Cutting Plane Constraints (22c)-(22f). (25h)

Constraint (25a) forces the start time s_{kj} of resolvent j in iteration k to fall on or after the completion time $(s_{ki} + t_i + l_{ij})$ of resolvent i in iteration k when $x_{ij} = 1$. Constraints (25b) and (25c) ensure that resolvents i and j cannot start iteration $k+1$ until after the completion of each in iteration k when $y_{ij} = 1$. Constraint (25d) sets b as the difference between the run time bound for r iterations (rq) and the actual end time for iteration r. The SDP, integer, and cutting plane constraints are all inherited from (12) and (24j). For relatively small problems ($n < 10$), this approach can identify iteration time minimizing designs which improve upon the block designs presented in Sect. 4.

As shown in Figure 4a, early iterations of the algorithm often have different patterns of completion than later iterations, due to the initial availability of all initialization values, so we recommend selecting $r \geq n$ iterations to model in the critical path. If we have constant communication times, we can also use the lower bound provided in Proposition 5 to tighten the formulation, which provides a significant performance improvement in the MISDP solution time. In the general case we can tighten the formulation with a lower bound of $\max_i \{t_i + \min_j \{l_{ij}\}\} + \min_i \{t_i + \min_j \{l_{ij}\}\}$. We also note that a restricted version of the MISDP in (25) can be solved as an MILP with the replacement of the PSD cone constraints in (25) with those in (24).

7 Convergence rates and optimal tuning

We now present a set of related SDPs which use the Performance Estimation Problem (PEP) approach described in [16] and [32] to optimize the step size γ and matrix W with respect to convergence rates of algorithms designed with (12) across a variety of problem classes. These classes include sums of strongly monotone operators, Lipschitz operators, and the subdifferentials of closed, convex, and proper functions, including indicator functions over convex sets, as long as the sum over the operators is guaranteed to have a zero. We note that throughout this section, as in Theorem 1, the modulus of strong monotonicity of an operator is permitted to be 0.

We begin first with an analysis of iteration (10), which provides both the worst-case contraction factor for a fixed L, M, and γ over all initializations and all monotone operators satisfying certain assumptions, as well as the ability to optimize the step size with respect to the contraction factor of $\mathbf{z}$. We then present an analysis of iteration (11) which allows us to determine the contraction factor of $\mathbf{v}$ and to optimize W and/or γ with respect to this contraction factor.

7.1 Optimal step size

In this section we build a PEP which upper bounds the worst-case contraction factor, $\tau_z = \left\|\mathbf{z}_1^1 - \mathbf{z}_2^1\right\|^2 / \left\|\mathbf{z}_1^0 - \mathbf{z}_2^0\right\|^2$, of (10) over all initializations and operators $(A_i)_{i=1}^n$ which are strongly monotone and Lipschitz with respect to some constants μ_i and l_i, respectively. The first step is to build a PEP which has an optimal value which provides an upper bound for τ_z. When $\dim(\mathcal{H}) \geq d + n$ we show the bound is tight. We then dualize this PEP and note that the primal and dual problems enjoy strong duality. Finally, we note that the dual problem can be parametrized with respect to the step size γ, allowing us to optimize over γ in the dual problem, resulting in a step size which minimizes our bound on the worst-case contraction factor.

For fixed $M \in \mathbb{R}^{d\times n}$ and $L \in \mathbb{R}^{n\times n}$, define block matrices K_I and (for i in $[n]$) K_{μ_i} and $K_{l_i} \in \mathcal{S}^{d+n}$ as

$$K_I = \begin{bmatrix} \mathrm{Id} & 0 \\ 0 & 0 \end{bmatrix} \tag{26}$$

$$K_{\mu_i} = \begin{bmatrix} 0 & -\frac{1}{2} M_{\cdot i} e_i^T \\ -\frac{1}{2} e_i M_{\cdot i}^T & \frac{1}{2}\left(e_i L_{i\cdot} + (e_i L_{i\cdot})^T\right) - (1+\mu_i) e_i e_i^T \end{bmatrix} \tag{27}$$

$$K_{l_i} = \begin{bmatrix} -M_{\cdot i} M_{\cdot i}^T & M_{\cdot i}\left(L_{i\cdot} - e_i^T\right) \\ \left(L_{i\cdot} - e_i^T\right)^T M_{\cdot i}^T & -\left(L_{i\cdot} - e_i^T\right)^T \left(L_{i\cdot} - e_i^T\right) + e_i e_i^T l_i^2 \end{bmatrix}. \tag{28}$$

We also define a function $\tilde{S} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \to \mathbb{S}^{d+n+d}$ as

$$\tilde{S}(\phi, \lambda, \psi, \gamma) = \begin{bmatrix} -\sum_i \phi_i K_{\mu_i} - \sum_i \lambda_i K_{l_i} + \psi K_I & \begin{bmatrix} \mathrm{Id} \\ \gamma M^T \end{bmatrix} \\ \begin{bmatrix} \mathrm{Id} & \gamma M \end{bmatrix} & \mathrm{Id} \end{bmatrix}.$$

This notation allows us to state Theorem 6, which establishes the use of the dual PEP to find the contraction factor minimizing step size and the minimal contraction factor.

Theorem 6 *For $M \in \mathbb{R}^{d\times n}$ and $L \in \mathbb{R}^{n\times n}$ derived from a valid design in (12) and $(\mu_i)_{i=1}^n$ and $(l_i)_{i=1}^n$ satisfying $0 \le \mu_i < l_i$ for all i in $[n]$, consider the problem*

$$\begin{aligned} \min_{\phi,\lambda,\psi,\gamma} \quad & \psi && \text{(29a)}\\ \text{subject to} \quad & \tilde{S}(\phi,\lambda,\psi,\gamma) \succeq 0 && \text{(29b)}\\ & \phi \ge 0,\ \lambda \ge 0 && \text{(29c)}\\ & \phi,\lambda \in \mathbb{R}^n && \text{(29d)}\\ & \psi,\gamma \in \mathbb{R}. && \text{(29e)} \end{aligned}$$

A solution to (29) provides a step size γ^* for (10) which minimizes the upper bound of the worst-case contraction factor τ_z, provided by ψ^*, over all initial values $\mathbf{z}_1^0$ and $\mathbf{z}_2^0$ and all possible μ_i-strongly monotone l_i-Lipschitz operators $(A_i)_{i\in[n]}$. When $\dim(\mathcal{H}) \ge d+n$, this bound is tight.

We note that $\gamma > 0$ is required in the proof of Theorem 1 for fixed points of (10) to yield a solution of (4). One can enforce this by adding the constraint $\gamma \ge c$ to (6) for some small $c > 0$. We also note that, if the step size γ is fixed in (29), the solution value ψ^* provides the worst case contraction rate τ_z for (10) for that step size.

7.2 Optimal matrix selection

We now turn to iteration (11). Given a feasible matrix Z in (12), we provide a method for finding γ and/or W which have the optimal contraction factor $\tau_v = \left\|\mathbf{v}_1^1 - \mathbf{v}_2^1\right\|^2 / \left\|\mathbf{v}_1^0 - \mathbf{v}_2^0\right\|^2$. This optimal contraction factor is valid over a set of operators defining (4) which have certain structural properties that can be characterized using the PEP framework [16].

For fixed $L \in \mathbb{R}^{n\times n}$, we define K_I, $K_{\mathbb{1}}$ and, for i in $[n]$, K_{μ_i} and K_{l_i}, each of which is in $\mathbb{S}^{2n}$, as

$$\begin{aligned} K_I &= \begin{bmatrix} \mathrm{Id} & 0 \\ 0 & 0 \end{bmatrix},\\ K_{\mathbb{1}} &= \begin{bmatrix} \mathbb{1}\mathbb{1}^T & 0 \\ 0 & 0 \end{bmatrix},\\ K_{\mu_i} &= \begin{bmatrix} 0 & \frac{1}{2}e_i e_i^T \\ \frac{1}{2}e_i e_i^T & \frac{1}{2}\left(e_i L_{i\cdot} + L_{i\cdot}^T e_i^T\right) - (1+\mu_i)\, e_i e_i^T \end{bmatrix}, \text{ and}\\ K_{l_i} &= \begin{bmatrix} -e_i e_i^T & -e_i\left(L_{i\cdot} - e_i\right) \\ -\left(L_{i\cdot} - e_i\right)^T e_i^T & -\left(L_{i\cdot} - e_i\right)^T\left(L_{i\cdot} - e_i\right) + e_i e_i^T l_i^2 \end{bmatrix}. \end{aligned}$$

We also define a function $\tilde{S} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \times \mathbb{S}^n_+ \to \mathbb{S}^{3n}$ as

$$\tilde{S}(\phi, \lambda, \psi, \omega, \tilde{W}) = \begin{bmatrix} -\sum_i \phi_i K_{\mu_i} - \sum_i \lambda_i K_{l_i} + \psi K_I + \omega K_{\mathbb{1}} & \begin{bmatrix} \mathrm{Id} \\ -\tilde{W} \end{bmatrix} \\ \begin{bmatrix} \mathrm{Id} & -\tilde{W}^T \end{bmatrix} & \mathrm{Id} \end{bmatrix}. \tag{30}$$

Theorem 7 *Let $L \in \mathbb{R}^{n \times n}$, $c > 0$, and $(\mu_i)_{i=1}^n$ and $(l_i)_{i=1}^n$ satisfy $0 \leq \mu_i < l_i$ for all i in $[n]$. Consider the problem*

$$\min_{\phi, \lambda, \psi, \omega, \tilde{W}} \quad \psi \tag{31a}$$

$$\text{subject to} \quad \tilde{S}\left(\phi, \lambda, \psi, \omega, \tilde{W}\right) \succeq 0 \tag{31b}$$

$$\phi \geq 0, \ \lambda \geq 0 \tag{31c}$$

$$\tilde{W}\mathbb{1} = 0 \tag{31d}$$

$$\lambda_1(\tilde{W}) + \lambda_2(\tilde{W}) \geq c \tag{31e}$$

$$\phi, \lambda \in \mathbb{R}^n \tag{31f}$$

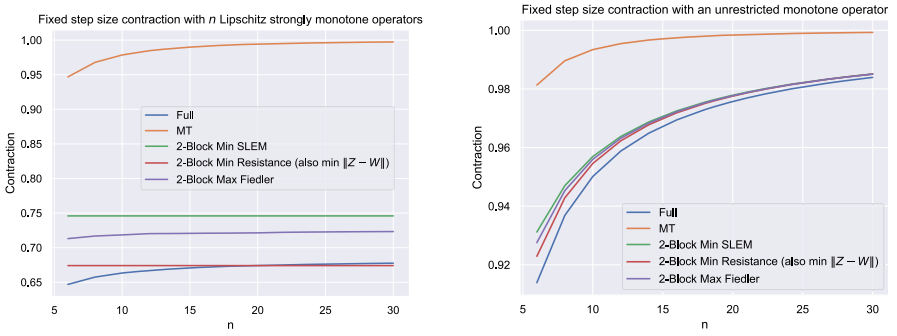
$$\psi, \omega \in \mathbb{R} \tag{31g}$$

$$\tilde{W} \in \mathbb{S}^n_+. \tag{31h}$$

A solution to (31) provides a matrix parameter $\tilde{W}^*$ for (11) which minimizes the upper bound of the worst-case contraction factor τ_v, provided by ψ^*, over all valid initial values $\mathbf{v}_1^0$ and $\mathbf{v}_2^0$ and all possible μ_i-strongly monotone l_i-Lipschitz operators $(A_i)_{i \in [n]}$. When $\dim(\mathcal{H}) \geq 2n$, this bound is tight.

8 Numerical experiments

This section presents numerical experiments which utilize and validate our contributions from the previous sections. We conduct three sets of experiments, choosing parameters for l_i and μ_i to minimize the impact of numerical solver tolerances on the results, which can be significant at high condition numbers. The first compares the contraction factor of d-Block designs, which compromise between a high degree of parallelism and high resolvent connectivity, with a fully connected design and the Malitsky-Tam algorithm introduced in [28]. We conduct this comparison with and without the optimal step size selection in Theorem 7, while also examining the impact of resolvent evaluation ordering and the choice of d in d-Block designs. Additionally, we numerically investigate the ideal amount of sparsity to balance between parallelism and rapid rate of convergence, demonstrating that d-Block designs make the most progress towards a solution during a fixed running time. A final experiment compares d-Block designs to minimum iteration time designs from Sect. 6.0.2 when the resolvent computation times and communication times are known. Throughout, in order to compare our contributions to previous results [28, 34], which are formulated in terms of iteration (10), we quantify the contraction factor in iteration (10) using



(a) Class 1 (b) Class 2

Fig. 5 Contraction factors with fixed $\gamma = 0.5$

Theorem 6. For d-Block designs and minimum iteration time designs from Sect. 6.0.2, we apply the sparse Cholesky decomposition from Sect. 3.1 to construct an M for use in Theorem 6. For the sake of reproducibility, all experiments from this section can be found in our accompanying software package github.com/peterbarkley/oars/, along with a number of additional examples analyzing the convergence as the condition number of the operators increases.

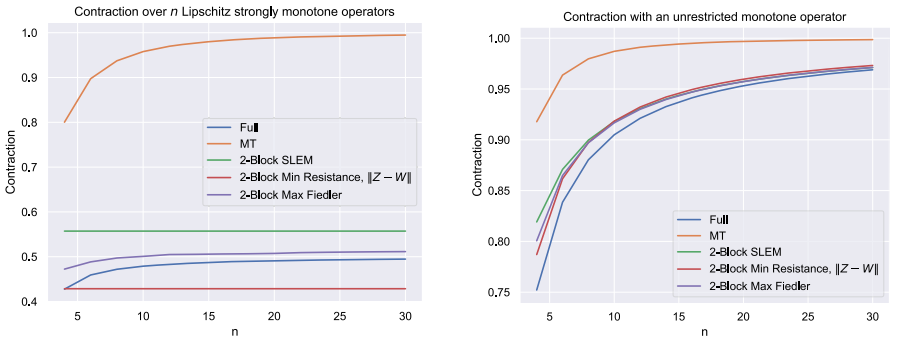
8.1 Comparing contraction factors of various designs

In our first experiment, we characterize the contraction factor τ from Theorem 6 for some of the design choices presented in Sects. 4 and 5. We do so for problems in two problem classes: n identical maximal l-Lipschitz μ-strongly monotone operators (which we call Class 1) and $n-1$ identical maximal l-Lipschitz μ-strongly monotone operators with one unrestricted maximal monotone operator (which we call Class 2).

We begin with a comparison of the spectral objective functions in Sect. 5 using a Class 1 problem with $l = 2$ and $\mu = 1$. We restrict the SDP to the 2-Block design. For comparison, we consider the Malitsky-Tam (MT) [28] and a fully connected design. For the fully connected design, we take the matrix Z to have 2 on its diagonal and $-\frac{2}{n-1}$ in all entries off the diagonal, and we set $W = Z$. For each design, we take the step size $\gamma = 0.5$.

Figure 5 displays the results. We note first that the maximally dense fully connected design provides the best contraction factor, and the maximally sparse MT design provides the worst. The minimum resistance objective (72) and minimizing $\|Z - W\|$ return the same matrices and provide the fastest convergence rate among the 2-Block designs.

We next conduct a similar test using optimal step sizes. We compare the various objective functions over the 2-Block design with MT and the fully connected design, both for Class 1 (with $l = 2$ and $\mu = 1$) and for Class 2 with the same parameters. We use the dual PEP formulation in Theorem 6 to calculate the optimal step sizes and contraction factor.



(a) Class 1 (b) Class 2

Fig. 6 Contraction factors with optimal γ from Theorem 7

Figure 6 provides the results. We see a dramatic improvement in convergence rates when using optimal step size in all cases. In the Class 1 case of identical operators, the 2-Block minimum resistance design (72) beats the fully connected algorithm for $n \geq 5$, and the 2-Block maximum Fiedler value algorithm (70) comes quite close to it as well. Both minimum resistance and minimum SLEM show invariance to the number of operators, n, in this scenario.

In Class 2, the invariance disappears, and the number of monotone operators strongly impacts the contraction factor for all designs. The fully connected design provides the best convergence rate, with the spectral objectives approaching the performance of the fully connected design as n increases. Below $n = 10$, the minimum resistance design (72) is slightly better than the other spectral designs. Above $n = 10$, the maximum Fiedler value (70) and minimum SLEM designs (71) begin to outperform minimum resistance by a small margin.

The matrices returned by the spectral objectives, whether restricted to a d-Block design or not, exhibit a high degree of symmetry. Tam's extension of the Ryu Algorithm [34], however, has a strong lack of symmetry, concentrating significant value in W onto the last resolvent. We therefore examine the impact of operator ordering on the MT, full, and block designs, and Tam's extension of the Ryu algorithm. In this test, we use the optimal step size for each algorithm design, and shift an unrestricted monotone operator between the last position and the first position. The other $n - 1$ operators are 2-Lipschitz 1-strongly monotone as before.

Figure 7 provides these results. For most algorithms, the convergence remains the same regardless of ordering, but for the extended Ryu algorithm, the impact is quite strong. With the unrestricted operator in the last position, the extended Ryu design performs similarly to the fully connected algorithm and the 2-Block minimum resistance design (72), which has the best contraction factor among the 2-Block designs. But when the unrestricted operator is moved from the last resolvent to the first resolvent becomes poor relative to its competitors, which are invariant to this shift in resolvent evaluation order.

Finally, we test the impact of block size on the contraction factor. As above, we test over Class 1 and Class 2 with $l = 2$ and $\mu = 1$. We use the optimal step size in all cases,

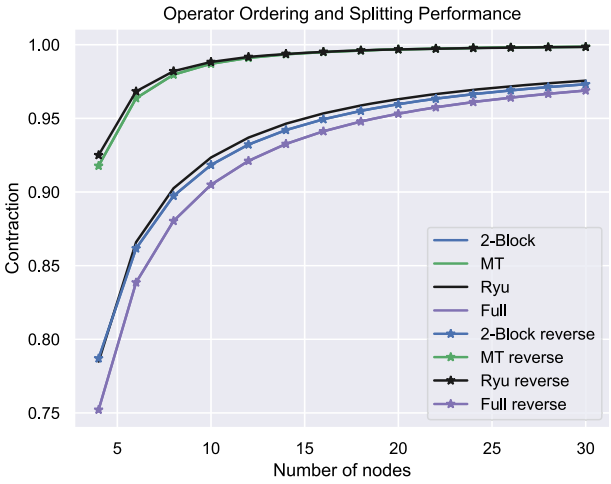
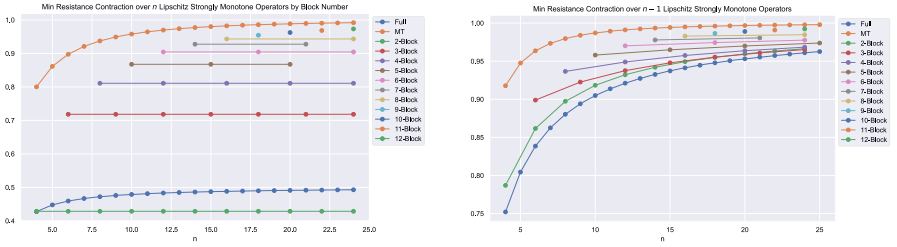


Fig. 7 Impact of unrestricted operator placement on convergence rate in Class 2 problems. All designs are invariant to swapping the role of A_1 and A_n except the extended Ryu algorithm, which has degraded performance when the restricted operator is evaluated first



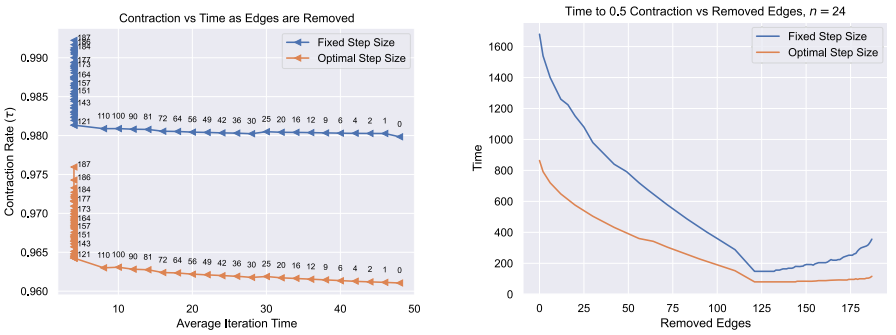
(a) Class 1 (b) Class 2

Fig. 8 Convergence rate by block size

and compare the minimum resistance design (72) over all possible constant-size block counts for a given n. We include the fully connected and MT designs for reference, which provide 1 and n blocks, respectively. Figure 8 displays the block size results. We observe that increasing to three or more blocks has a strong negative impact on the contraction factor, particularly in the identical operator case. For Class 2, the 2-Block design contraction factor approaches that of the 3-Block design near $n = 15$.

8.2 The ideal amount of sparsity

In our next experiment, we analyze the total time required for a contraction of 0.5, given a fixed amount of algorithm running time. This combines the average iteration time of the algorithm designs with their contraction factor in order to determine overall performance. We test across Class 2 with $l = 2$ and $\mu = 1$, using both fixed and optimal step size, and the maximum Fiedler value design (70). We progressively increase matrix sparsity in Z toward the 2-Block design, removing edges alternately from



(a) Convergence vs iteration time

(b) Time to half contraction vs sparsity

Fig. 9 Impact of convergence and average iteration time on total algorithm time across $(n-1)$ Lipschitz strongly monotone operators with one monotone operator as sparsity increases to 2-Block design and beyond

within the first and second block. Figure 10 illustrates the edge removal ordering. After reaching the 2-Block design, we remove edges progressively from the remaining block matrices beginning near the diagonal. All resolvent computations and communications are treated as a constant single time unit throughout.

Figure 9 displays the results for $n = 24$. In Figure 9a, moving from right to left we see the gradual decrease in average iteration time as edges are removed toward the 2-Block design. Once the 2-Block design is reached, continuing to remove edges shifts the convergence rate upwards without any further decrease in average iteration time. This point is the phase transition of each of the parametrized curves in Figure 9. Figure 9bshows the decrease in total convergence time as edges are removed. We observe a strong inflection point at the 2-Block design, with total time rising gradually thereafter as more edges are pruned. These results demonstrate the utility of constructing designs within the d-Block constraints.

8.3 Minimum iteration time and *d*-block designs

We now turn our attention to problems with arbitrary resolvent computation and communication times. Although the d-Block designs for small d consistently show strong performance in terms of contraction factor and average iteration time, it is possible in some cases to reduce the total time required for convergence by using the minimum iteration time formulations (25) and (24).

Achieving competitive performance of formulations (25) and (24) with d-Block designs requires the addition of another constraint—a minimum number of nonzero entries for each row in W. From a graph-theoretic perspective, this provides a minimum number of edges for each node in $G(W)$. Experiments with various constraint levels have shown that requiring at least $\lfloor \frac{n}{2} \rfloor$ entries in each row preserves sufficient density in W to keep the convergence rate comparable with the 2-Block design, so we subject our time minimizing designs to an additional constraint requiring $\lfloor \frac{n}{2} \rfloor$ nonzero entries in each row throughout this experiment. For even n, we compare the performance of a time minimizing design with a 2-Block design, and for odd n we use a 3-Block design with block sizes $\left(\frac{n-1}{2}, \frac{n-1}{2}, 1\right)$, which we found to perform best across the set of size

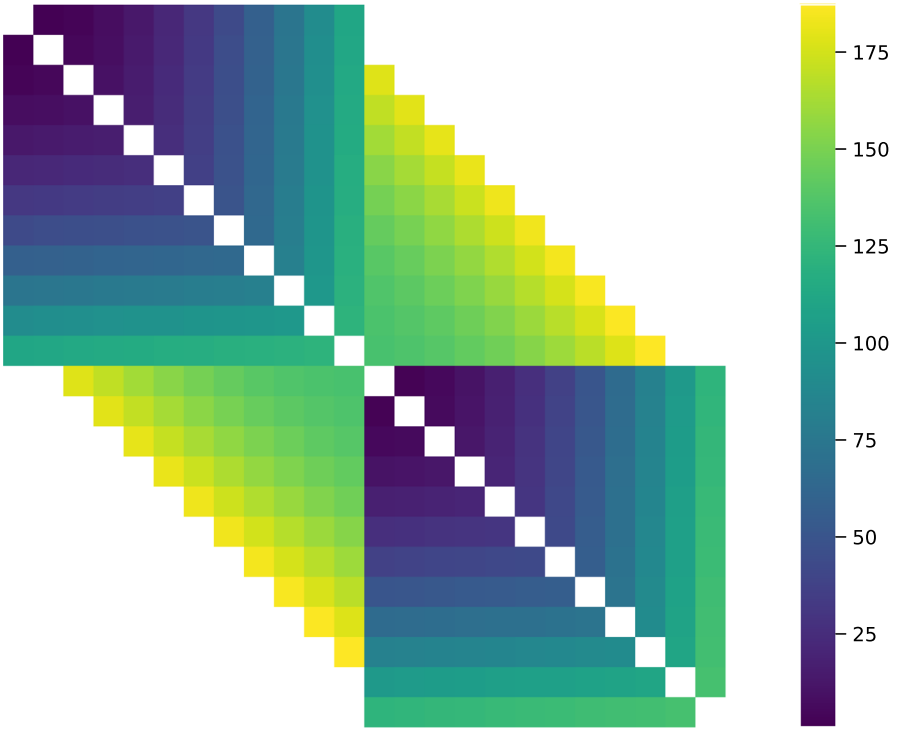


Fig. 10 In Figure 9, edges are alternately removed from block matrices (1,1) and (2,2) starting in the upper left off-diagonal positions and progressively removing edges to the right and down. Once block matrices (1, 1) and (2, 2) are restricted to identities, we then prune within the remaining blocks ((1, 2) and (2, 1), which are equivalent by symmetry), and proceed in the top right of block matrix (2, 1) from the diagonal of the entire matrix working outward

3 partitions. We construct the d-Block designs using the minimum total resistance objective function (72), and we construct the minimum iteration time design using the scalable MILP formulation in problem (24), which we formulate in Pyomo and solve with Gurobi [8, 23, 24].

We assess the performance of each design by computing the total time required to achieve a 0.01 contraction factor using the single-iteration contraction factor τ from Theorem 6 with optimal step size. We use a Class 2 problem set with $l = 2$ and $\mu = 1$ as before, and conduct 40 trials each for $n \in \{6, 7, 8, 9\}$, selecting the resolvent computation times uniformly from [0.5,2.0] and the communication times uniformly from [1,11]. Figure 11 provides the results. While the 2-Block design remains quite good for even n, for odd n the minimum iteration time results provide consistently better performance.

9 Conclusion

This work presents a novel framework for designing frugal resolvent splittings suitable for particular applications. We prove the convergence of these algorithms over

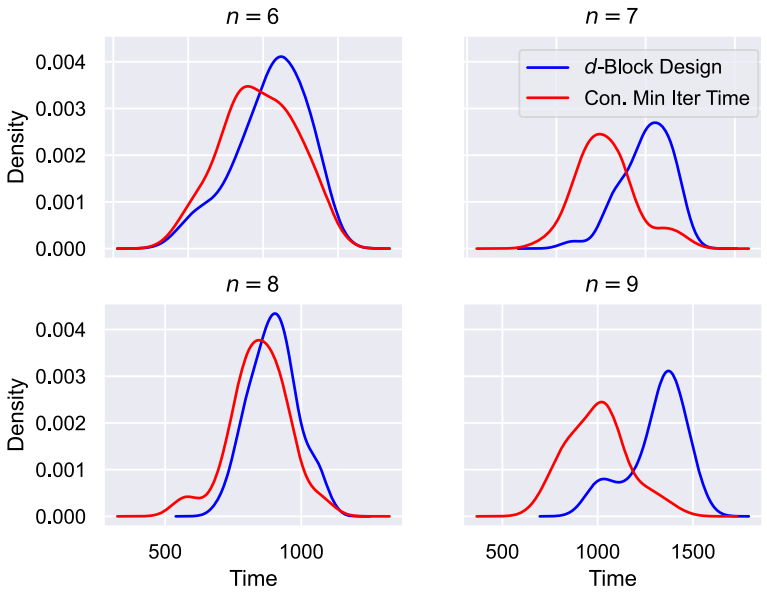


Fig. 11 Comparison of d-Block and the constrained minimum iteration time design from (24) on randomly generated resolvent computation and communication times. d-Block designs are 2-Block for even n and $\left(\frac{n-1}{2}, \frac{n-1}{2}, 1\right)$ for odd n and use the minimum total effective resistance objective (72). Constrained Minimum Iteration Time designs use the MILP formulation (24) with the additional edge constraint described in the text

n maximal monotone operators whose sum has a zero, offering an enlarged range of valid step sizes and an approach which provides an optimized step size given a characterization of the problem class. We establish the equivalence of all algorithms in (10) with minimal lifting designs. We also further develop the connection between the algorithm matrices and the weighted graph Laplacian, using the connection to inform the set of feasible constraints for (12).

We demonstrate the utility of the framework with a number of possible constraint sets and objective functions. These allow minimization of iteration time for both uniform and arbitrary computation and communication times, as well as designs which optimize over a wide variety of common heuristics. We also demonstrate the use of the PEP framework and its dual with these designs, optimizing W and γ, and providing contraction factors under various structural assumptions on the monotone operators.

Numerically, we validate the performance of our contributions by comparing their performance in settings with different computation times, communication times, and problem classes. We show that d-Block designs, for small values of d, outperform competitors from the literature. We compare the designs optimized by the various spectral objectives, with and without the optimal step size, and find that minimizing the total effective resistance performs consistently well. We analyze algorithm design performance when the monotone operators are re-ordered, and find consistent performance across most designs, but note that the asymmetric extended Ryu design can perform poorly in this setting. Finally, we demonstrate that when resolvent computa-

tion times and communications times are known or can be accurately estimated, the minimal iteration time designs we propose in Sect. 6.0.2 outperform d-Block designs, but the marginal improvement suggests that d-Block designs still can be expected to exhibit good performance when these times are not available.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12532-026-00304-7.

Acknowledgements Both authors acknowledge support from the Office of Naval Research. The views expressed in this article are those of the authors and do not reflect the official policy or position of the U.S. Naval Academy, Naval Postgraduate School, Department of the Navy, the Department of Defense, or the U.S. Government

Funding The authors acknowledge support from the Office of Naval Research under grants O2501017017073407 and O2512017017177746.

Data availability No data were used for the research described in this article.

Code Availability Source code for the contributions and numerical experiments in this article can be found at https://github.com/peterbarkley/oars.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aragón Artacho, F.J., Campoy, R., Tam, M.K.: Strengthened splitting methods for computing resolvents. Comput. Optim. Appl. **80**(2), 549–585 (2021). https://doi.org/10.1007/s10589-021-00291-6
2. Aragón-Artacho, F.J., Campoy, R., López-Pastor, C.: Forward-backward algorithms devised by graphs. arXiv preprint arXiv:2406.03309 (2024)
3. Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in Hilbert spaces, vol. 408. Springer (2011)
4. Ben-Tal, A., Nemirovski, A.: Lectures on modern convex optimization: analysis, algorithms, and engineering applications. SIAM (2001)
5. Boyd, S.: Convex optimization of graph Laplacian eigenvalues. pp. 1311–1319 (2006)
6. Bredies, K., Chenchene, E., Lorenz, D.A., et al.: Degenerate preconditioned proximal point algorithms. SIAM J. Optim. **32**(3), 2376–2401 (2022). https://doi.org/10.1137/21M1448112
7. Bredies, K., Chenchene, E., Naldi, E.: Graph and distributed extensions of the Douglas-Rachford method. SIAM J. Optim. **34**(2), 1569–1594 (2024)
8. Bynum, M.L., Hackebeil, G.A., Hart, W.E., et al.: Pyomo-optimization modeling in Python, vol. 67, 3rd edn. Springer Science & Business Media (2021)
9. Campoy, R.: A product space reformulation with reduced dimension for splitting algorithms. Comput. Optim. Appl. **83**(1), 319–348 (2022)
10. Chen, C., He, B., Ye, Y., et al.: The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Math. Program. **155**(1–2), 57–79 (2016)
11. Chung, F.R.: Spectral graph theory, vol. 92. American Mathematical Society (1997)
12. Colla, S., Hendrickx, J.M.: On the optimal communication weights in distributed optimization algorithms. arXiv preprint arXiv:2402.05705 (2024)
13. Combettes, P.L., Pesquet, J.C.: A proximal decomposition method for solving convex variational inverse problems. Inverse Prob. **24**(6), 065014 (2008)
14. Condat, L., Kitahara, D., Contreras, A., et al.: Proximal splitting algorithms for convex optimization: A tour of recent advances, with new twists. SIAM Rev. **65**(2), 375–435 (2023)

15. Douglas, J., Rachford, H.H.: On the numerical solution of heat conduction problems in two and three space variables. Trans. Am. Math. Soc. **82**(2), 421–439 (1956)
16. Drori, Y., Teboulle, M.: Performance of first-order methods for smooth convex minimization: a novel approach. Math. Program. **145**(1), 451–482 (2014)
17. Eckstein, J., Bertsekas, D.P.: On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**, 293–318 (1992)
18. Eckstein, J., Svaiter, B.F.: General projective splitting methods for sums of maximal monotone operators. SIAM J. Control. Optim. **48**(2), 787–811 (2009)
19. Fiedler, M.: Algebraic connectivity of graphs. Czechoslov. Math. J. **23**(2), 298–305 (1973)
20. Fiedler, M.: A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. Czechoslov. Math. J. **25**(4), 619–633 (1975)
21. Gally, T., Pfetsch, M.E., Ulbrich, S.: A framework for solving mixed-integer semidefinite programs. Optimization Methods and Software **33**(3), 594–632 (2018)
22. Goujaud, B., Moucer, C., Glineur, F., et al.: PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python. Math. Program. Comput. **16**(3), 337–367 (2024)
23. Gurobi Optimization, LLC (2023) Gurobi Optimizer Reference Manual. https://www.gurobi.com
24. Hart, W.E., Watson, J.P., Woodruff, D.L.: Pyomo: modeling and solving mathematical programs in Python. Math. Program. Comput. **3**(3), 219–260 (2011)
25. Kelley, J.E., Jr.: Critical-path planning and scheduling: mathematical basis. Oper. Res. **9**(3), 296–320 (1961)
26. Li, Z., Shi, W., Yan, M.: A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. IEEE Trans. Signal Process. **67**(17), 4494–4506 (2019)
27. Malitsky, Y., Tam, M.K.: A first-order algorithm for decentralised min-max problems. arXiv preprint arXiv:2308.11876 (2023a)
28. Malitsky, Y., Tam, M.K.: Resolvent splitting for sums of monotone operators with minimal lifting. Math. Program. **201**(1–2), 231–262 (2023)
29. Rockafellar, R.T.: Convex analysis. Princeton Math Series 28 (1970a)
30. Rockafellar, R.T.: On the maximal monotonicity of subdifferential mappings. Pac. J. Math. **33**(1), 209–216 (1970)
31. Ryu, E.K.: Uniqueness of DRS as the 2 operator resolvent-splitting and impossibility of 3 operator resolvent-splitting. Math. Program. **182**(1–2), 233–273 (2020)
32. Ryu, E.K., Taylor, A.B., Bergeling, C., et al.: Operator splitting performance estimation: Tight contraction factors and optimal parameter selection. SIAM J. Optim. **30**(3), 2251–2271 (2020)
33. Shi, W., Ling, Q., Wu, G., et al.: A proximal gradient algorithm for decentralized composite optimization. IEEE Trans. Signal Process. **63**(22), 6013–6023 (2015). https://doi.org/10.1109/TSP.2015.2461520
34. Tam, M.K.: Frugal and decentralised resolvent splittings defined by nonexpansive operators. Optimization Letters pp 1–19 (2023)
35. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM review **38**(1), 49–95 (1996)
36. Yurtsever, A., Tropp, J.A., Fercoq, O., et al.: Scalable semidefinite programming. SIAM Journal on Mathematics of Data Science **3**(1), 171–200 (2021)