# Exploiting Locality and Structure for Distributed Optimization in Multi-Agent Systems

Robin Brown[1], Federico Rossi[2], Kiril Solovey[3], Michael T. Wolf[2], and Marco Pavone[3]

***Abstract*— A number of prototypical optimization problems in multi-agent systems (e.g. task allocation and network load-sharing) exhibit a highly local structure: that is, each agent's decision variables are only directly coupled to few other agent's variables through the objective function or the constraints. Nevertheless, existing algorithms for distributed optimization generally do not exploit the locality structure of the problem, requiring all agents to compute or exchange the full set of decision variables. In this paper, we develop a rigorous notion of "locality" that relates the structural properties of a linearly-constrained convex optimization problem (in particular, the sparsity structure of the constraint matrix and the objective function) to the amount of information that agents should exchange to compute an arbitrarily high-quality approximation to the problem from a cold-start. We leverage the notion of locality to develop a *locality-aware* distributed optimization algorithm, and we show that, for problems where individual agents only require to know a small portion of the optimal solution, the algorithm requires very limited inter-agent communication. Numerical results show that the convergence rate of our algorithm is directly explained by the locality parameter proposed, and that the proposed theoretical bounds are remarkably tight.**

## I. Introduction

Many problems in multi-agent control are naturally posed as a large-scale optimization problem, where knowledge of the problem is distributed among agents, and the collective actions of the network are summarized by a global decision variable. Concerns about communication overhead and privacy in such settings have motivated the need for distributed solution algorithms—chiefly those that preclude explicitly gathering all of the problem data in one location. This is strikingly similar to a prominent setting in the literature on distributed optimization where knowledge of the objective function is distributed, i.e., can be expressed as the sum of privately known functions, and agents must reach a consensus on the optimal decision vector despite limited inter-agent communication. We refer the reader to [1] for a recent survey on distributed optimization.

For many settings, this problem formulation is appropriate, such at rendezvous or flocking, where all agents' actions depend of the global decision variables (meeting time and location for the former, and speed and heading for the latter). However, when the global decision variable represents a concatenation of individual actions, the network can still act optimally without ever coming to a consensus. Take, for example, a centralized solution where the problem data is collected by a single computation node, who solves the problem, and passes the appropriate solution components to each agent.

Many existing distributed optimization algorithms leverage consensus as a core building block. In general, many of them can be abstracted as the interleaving of descent steps, to drive the solution to the optimum, and averaging of information from neighbors to enforce consistency. The main features differentiating these algorithms from each other are the centralized algorithm from which they are derived, and details regarding the communication structure such as synchronous or asynchronous, and directed or undirected communication links, with the broad overarching categories being consensus-based (sub)gradient ([2], [3], [4], [5], [6]), (sub)gradient push ([7], [8]), dual-averaging ([9], [10]), and distributed second-order schemes ([11], [12]).

Our main objective of this paper is to show that under reasonable assumptions about problem instances for multi-agent systems, the large communication overhead incurred by such algorithms is unnecessary. Specifically, we will take advantage of sparsity structure in the constraints and objective to develop a notion of "locality", namely the property that solution components can be computed with high accuracy without full knowledge of the problem. Under such assumptions, we will illustrate that an algorithm that restricts information exchange to "where it matters most" is significantly more efficient than those which rely on eventually disseminating information through the entire network.

Our approach builds on the work of Rebschini and Tatikonda [13], who introduced a notion of correlation in purely-deterministic settings as a structural property of optimization problems. The authors in [13] characterize the "locality" of network-flow problems, and show that the notion of locality can be applied to develop computationally-efficient algorithms for "warm-start" optimization, i.e., re-optimizing a problem when the problem is perturbed. Moallemi and Van Roy [14] have also explored similar notions of correlation, but solely as a tool to prove convergence of min-sum message passing algorithm for unconstrained convex optimization. To the best of our knowledge, [13] is the only prior work to advocate for a general theory of locality in the context of multi-agent systems.

*Statement of Contribution:* The contributions of this paper are twofold. First, we formalize a notion of "locality" in linearly constrained convex optimization problems, and devise a rigorous metric that captures the locality of an optimization problem and explicitly takes into account the problem structure. Second, we leverage the locality metrics proposed to design a novel distributed approximation algorithm that exploits locality to drastically reduce the communication needed to approximate the optimal solution over existing "locality-agnostic" distributed algorithms.

[1]R. Brown is with the Institute for Computational & Mathematical Engineering, Stanford University, Stanford, CA, 94305, rabrown1@stanford.edu.

[2] K. Solovey and M. Pavone are with the Department of Aeronautics & Astronautics, Stanford University, Stanford, CA, 94305, {kirilsol,pavone}@stanford.edu.

[3] F. Rossi, and M. T. Wolf are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, {federico.rossi, michael.t.wolf}@jpl.nasa.gov.

*Organization:* The paper is organized as follows. In Section II we introduce our notation and terminology, and formally provide the problem statement. In Section III, we motivate and propose a rigorous metric of the "locality" of an optimization problem. In Section IV we show that locality can be exploited to design communication-efficient distributed optimization algorithms. In Section V, we validate our theoretical results on a network load-sharing example. Section VI contains concluding remarks and highlights future directions. The extended version of this paper [15] contains additional discussion and numerical results and a full proof of all theorems.

## II. Preliminaries

### A. Notation

We use $[N] := \{1,...,N\}$ to denote the $1-N$ index set, and $e_i$ denotes the canonical $i$th basis vector i.e., the vector with 1 in position $i$ and zero elsewhere, where the size of the vector will be clear from context. For a given matrix $A$, $A_{ij}$ denotes the element in the $i$th row and $j$th column of $A$. Similarly let $A_{i,*}$ and $A_{*,j}$ denote the $i$th row and $j$th column of $A$ respectively. Let $A^T$ be the transpose, and $A^{-1}$ be the inverse of matrix $A$. Given subsets $I \subseteq M, J \subseteq N$, let $A_{I,J} \in \mathbb{R}^{|I|\times|J|}$ denote the submatrix of $A$ corresponding to the rows and columns of $A$ indexed by $I$ and $J$, respectively. Similarly, let $A_{-I,-J}$ denote the submatrix of $A$ obtained by removing rows $I$ and columns $J$. We define a *partition* of a set $C$ to be a collection of subsets $C_i, i = 1,..,k$ such that $\bigcup_{i=1,...k} C_i = C$ and $C_i \cap C_j = \emptyset$ for all $i \neq j$.

We define an undirected graph $G = (V,E)$ by its vertex set $V$ and edge set $E$, where elements $(u,v) \in E$ are unordered tuples with $u,v \in V$. We let $\mathcal{N}_G(v) := \{u \in V | (u,v) \in E\}$ be the neighbors of node $v$. For a subset $S \subseteq V$, we let $\mathcal{N}_G(S) = \bigcup_{v\in S} \mathcal{N}_G(v)$. Wedefine the graph distance $d_G(u,v)$ to be the length of the shortest path between vertices $u$ and $v$ in graph $G$.

### B. Problem Setup

We consider a network of $N$ agents collectively solving the following linearly constrained optimization problem

$$\begin{aligned} &\underset{x\in\mathbb{R}^N}{\text{minimize}} && f(x) \\ &\text{subject to} && Ax = b \end{aligned} \tag{1}$$

where knowledge of the constraints is distributed, and the decision vector represents a concatenation of the decisions of individual agents. Specifically, we assume that $f$ is known by all of the agents, $A_{*j}$ is initially known by agent $j$ only, and agent $j$ knows $b_i$ if $A_{ij} \neq 0$. As a departure from a large body of the existing literature on distributed optimization, we allow any solution where each agent $j$ knows $x_j^*$—that is, we do not require every agent to know the entire optimal decision variable. With some abuse of notation, we conflate each agent with its associated primal variable.

As a motivating example, consider a scenario where a fleet of agents need to collectively complete tasks at various locations, while minimizing the cost of completing such tasks. In this setting, the constraints ensure completion of the tasks, while the entries of the constraint matrix may encode the portion of a task that an agent can complete, or efficiency when completing tasks, thus, constituting private knowledge.

While, in this paper, each agent is only associated with a scalar variable for illustrative purposes, one can readily extend the results in this paper to the setting where each agent is associated with a vector. Additionally, the case where multiple agents' actions depend on shared variables can be addressed by creating local copies of those variables and enforcing consistency between agents who share that variable through a coupling constraint.

Additionally, we assume that $A \in \mathbb{R}^{M\times N}$ is full rank, and the function $f : \mathbb{R}^N \to \mathbb{R}$ is strongly convex, and twice continuously differentiable. In problem 1, let $V^{(p)} = [N]$ denote the set of primal variables, $V^{(d)} = [M]$ the set of dual variables, and $S_j = \{i \in V^{(p)} | A_{ji} \neq 0\}$ the set of agents participating in the $j$th constraint. Throughout this paper, we fix the objective function $f$ and the constraint matrix $A$, and write $x^*(b)$ as a function of the constraint vector, $b$.

At this point, we make no assumptions on the communication structure of the network. In the literature on distributed optimization, the underlying communication graph is typically assumed to be fixed, however, in practice it can be modulated, and should be co-designed with the solution algorithm. Our analysis of locality gives a method of quantifying the importance of problem information to solution components, and assessing the value of communicating certain pieces of information.

## III. Closed-form Locality Metric

In this section, we propose a rigorous metric of the "locality" of an optimization problem. Specifically, in Section III-A, we provide a method to track the cascading effects from perturbations of the constraint vector, $b$, and quantify its effect on the optimal solution. Then in III-B, we provide conditions under which this cascading effect is damped as it propagates; the damping allows to solve for components of the optimal solution with only partial information of the problem—the rate of damping naturally characterizes the trade-off between solution accuracy and the quantity of problem information used. Our method quantifies the structural amount of information that each computation node needs to receive from other nodes in the network to yield an approximate solution with a certain accuracy. These results are *global*—they hold true on the entire space of feasible instances of problem 1, not just in a neighborhood of the optimal solution. This distinction will later (in Section IV) allow us apply these results to design distributed optimization algorithms that can converge from a cold-start (i.e., with no prior knowledge of a "good" solution).

Our results for this section build on the analysis in [13] on the sensitivity of optimal points to finite perturbations of the constraint vector for linearly constrained convex optimization problems. We first review the relevant results, and then present our contribution.

**Theorem III.1** (Sensitivity of Optimal Points - Theorem 1 of [13])**.** Let $f : \mathbb{R}^N \to \mathbb{R}$ be strongly convex and twice continuously differentiable, and $A \in \mathbb{R}^{M\times N}$ have full row rank. For $b \in \mathrm{Im}(A)$, let $\Sigma(x^*(b)) := \nabla^2 f(x^*(b))^{-1}$ . Then $A\Sigma(x^*(b))A^T$ is invertible, $x^*(b)$ at all $b \in \mathbb{R}^m$, and

$$\frac{\mathrm{d}x^*(b)}{\mathrm{d}b} = D(b) = \Sigma(x^*(b))A^T(A\Sigma(x^*(b))A^T)^{-1}. \tag{2}$$

The above theorem relates the gradient of the optimal solution, $x^*(b)$, to the constraint matrix and the objective function. This result will be the building block for under-

standing, structurally, how the optimal solution depends on problem information.

### A. Sparsity of Multi-Agent Convex Optimization Problems

In many practical problems, both $A$ and $\nabla^2 f(x)$ are sparse and highly structured. That is, agents only participate in a small subset of the constraints, and the objective function is only loosely coupled across agents. For example, constraints might encode collision avoidance within a fleet of robots where one robot has no chance of colliding with a robot that is far away [16], or consumption of shared resources where each individual resource may only be accessed by a small fraction of the network, as is the case for transmission link bandwidth in the setting of network utility maximization [17]. Additionally, a common objective function is one where the global cost function is simply the sum of individual agents' cost functions.

One might hope that such problems would be more amenable to a distributed solutions compared to ones where the constraints and objective are densely coupled among agents. However, the cascading effects that one decision has on the remainder of the network makes the analysis of such problems challenging. We use the sensitivity expression in Equation (2) to reason about this structural coupling, but the terms $(A\Sigma(x^*(b))A^T)^{-1}$ and $\Sigma(x^*(b)) = \nabla^2 f(x^*(b))^{-1}$ require careful treatment. Specifically, the inverse of sparse matrices is not guaranteed to be sparse, and in fact, is typically dense (corresponding to the aforementioned cascading effects). While the structure of the original problem is obfuscated when we take the inverses, $(A\Sigma(x^*(b))A^T)^{-1}$ and $\Sigma(x^*(b)) = \nabla^2 f(x^*(b))^{-1}$, it can still be recovered by exploiting the following key insights:

1) $A\Sigma(x)A^T$ can be expressed as $(L(x)^{-1}A^T)^T(L(x)^{-1}A^T)$, where $L(x)$ is the Cholesky factorization of $\Sigma(x)$. Moreover, the sparsity pattern of $L(x)$ can be characterized in closed-form.
2) The columns of $L(x)^{-1}A^T$ are solutions to sparse linear systems. Explicitly,
$$L(x) \times [L(x)^{-1}A^T]_{*i} = [A^T]_{*i}.$$
3) Under the appropriate spectral conditions, $(A\Sigma(x^*(b))A^T)^{-1}$ can be expressed as the Neumann series $\sum_{i=0}^{\infty}(I - A\Sigma(x)A^T)^i$.

We refer the reader to the Supplementary Material for discussion on how to determine the sparsity patterns of the Cholesky factorization and the solution of sparse linear systems. [1] This will naturally give rise to a distance metric between primal and dual variables, and will allow us to identify conditions under which sensitivity of components of the optimal solution decays as a function of distance to perturbations in the constraint vector. This corresponds to the notion that one constraint will not have an out-sized effect on the remainder of the network. We now define several graphs that will allow us to reason about the numerical structure of the sensitivity expression using the sparsity patterns of the terms composing the expression. For fixed $x \in D$, define the following undirected graphs:

- $G_{\text{obj}}(x) = (V^{(p)}, E_{\text{obj}}(x))$, with $E_{\text{obj}}(x) = \{(i,j)|[\nabla^2 f(x)]_{ij} \neq 0\}$. Informally, $G_{\text{obj}}(x)$ encodes direct links between primal variables through the Hessian of the objective function.
- $G_{\text{eff}}(x) = (V^{(d)}, E_{\text{eff}}(x))$, with $E_{\text{eff}} = \{(i,j)|[A\Sigma(x)A^T]_{ij} \neq 0\}$. Informally, $G_{\text{eff}}(x)$ encodes direct links between dual variables, by tracing through shared primal variables and the Hessian of the objective function.
- $G_{\text{opt}} = (V^{(p)} \cup V^{(d)}, E_{\text{opt}}(x))$, with $E_{\text{opt}} = \{(v_j^{(p)}, v_i^{(d)})|A_{ij} \neq 0\}$. Informally, $G_{\text{opt}}$ is the bipartite graph showing the primal variables involved in each constraint.

We define $G_{\text{obj}} := (V^{(p)}, \bigcup_{x\in D} E_{\text{obj}}(x))$ and $G_{\text{eff}} := (V^{(d)}, \bigcup_{x\in D} E_{\text{eff}}(x))$ to eliminate dependence on the specific value of $x$ where these graphs are evaluated.

Using these graphs, the following theorem and corollary will allow us to derive the sparsity pattern of the terms in the previously mentioned Neumann series.

**Theorem III.2** (Sparsity Structure of Matrix Powers)**.** For $k \in \mathbb{Z}_+$, neglecting numerical cancellation[2],
$$\begin{aligned}\text{supp}((I - A\Sigma(x)A^T)^k) &= \{(i,j)|d_{G_{\text{eff}}(x)}(v_i, v_j) \leq k\} \\ &\subseteq \{(i,j)|d_{G_{\text{eff}}}(v_i, v_j) \leq k\}.\end{aligned}$$

The previous theorem establishes that the sparsity pattern of a symmetric matrix to the $k$th power is determined by the $k$-hop neighbors in the graph representing the sparsity pattern of the original matrix. This allows us the characterize the sparsity pattern of each term in the Neumann series $\sum_{i=0}^{\infty}(I - A\Sigma(x)A^T)^i$. This, in turn, can be used to derive the sparsity pattern when each term of the series is pre-multiplied by $\Sigma(x)A^T$. We define the set $\mathcal{N}_k^{(d)}(i) := \{v \in V^{(d)}|d_{G_{\text{eff}}}(v_i^{(d)}, v) \leq k\}$ to be the set of vertices of distance at most $k$ from vertex $i$ in $G_{\text{eff}}$

**Corollary III.2.1** (Sparsity Structure of the Sensitivity Expression)**.** For $k \in \mathbb{Z}_+$ and $i \in [M]$
$$\begin{aligned}\text{supp}\left(\Sigma(x)A^T(I - A\Sigma(x)A^T)^k\right) e_i \\ \subseteq \mathcal{N}_{G_{\text{obj}}}(\mathcal{N}_{G_{\text{opt}}}(\mathcal{N}_k^{(d)}(i))).\end{aligned} \quad (3)$$

The proofs of Theorem III.2 and Corollary III.2.1 rely on combinatorially deducing the entries that must be zero in the above expressions. The full proofs and all subsequent proofs are included in the Supplementary Material. Intuitively, $\mathcal{N}_{G_{\text{obj}}}(\mathcal{N}_{G_{\text{opt}}}(\mathcal{N}_k^{(d)}(i)))$ represents the components of $\left(\Sigma(x)A^T(I - A\Sigma(x)A^T)^k\right)e_i$ that can be nonzero based on combinatorial analysis of the terms of $\left(\Sigma(x)A^T(I - A\Sigma(x)A^T)^k\right)e_i$. We will later consider a truncated approximation of the sensitivity expression. The consequence of Corollary III.2.1 is that we know which components of the approximation are guaranteed to be zero i.e., are invariant to locally supported perturbations in the constraint vector.

Based on the previous theorem and its corollary, we define a measure of distance between primal variables and dual variables that characterizes the indirect path, through coupling in the constraints and the objective function, by which a perturbation in the constraint propagates to primal

[1] We use the term "sparsity pattern" to refer to the pattern of non-zero entries of a matrix.

[2] When characterizing the sparsity pattern of a matrix, "numerical cancellation" refers to when entries that are zeroed out due to the exact values of entries in the matrix, cannot be deduced to be zero from the combinatorial structure of the matrix alone.

variables.

$$d(v_i^{(p)}, v_j^{(d)}) := \min\{k | \mathcal{N}_{\text{obj}}(\mathcal{N}_{\text{opt}}(V_k^{(d)}(i)))\}.$$

We also define the distance between sets of primal and dual variables as

$$d(I,J) = \min\{d(v_i^{(p)}, v_j^{(d)}) | v_i^{(p)} \in I, v_j^{(d)} \in J\}.$$

### B. Spectral Conditions for Locality

We are now in a position to define our metric of locality, and provide conditions under which a problem exhibits locality.

**Definition III.1** (Exponential Locality)**.** We say a linearly constrained convex optimization problem is *exponentially local under distance metric d, with parameter* $\lambda$ if there exists nonnegative $\lambda < 1$, a distance metric, $d$, between primal and dual variables, and constant $c$, such that for all subsets $S \subseteq V^{(p)}$ and $b, \Delta \in \text{Im}(A)$

$$\|(x^*(b+\Delta) - x^*(b))_S\| \leq c\,\|\Delta\|\,\lambda^{d(S,\text{supp}(\Delta))} \tag{4}$$

Intuitively, exponential locality is the condition that perturbations in the constraint vector result in perturbations in the optimal solution that decay exponentially as a function of distance between the solution components and the perturbation.

The next theorem provides conditions for which an optimization problem is exponentially local.

**Theorem III.3** (Spectral Conditions for Exponential Locality of Linearly Constrained Convex Optimization Problems)**.** A linearly constrained convex optimization problem is exponentially local with parameter $\lambda$ if $\sup_x \rho(I - A\Sigma(x)A^T) = \lambda < 1$, where $\rho(M)$ denotes the largest singular value of the matrix $M$.

*Proof Sketch.* Under the spectral conditions specified, we can express $(A\Sigma(x^*(b))A^T)^{-1}$ as the Neumann series $\sum_{i=0}^{\infty}(I - A\Sigma(x)A^T)^k$. We split the sensitivity expression

$$\begin{aligned}
&(x^*(b+\Delta) - x^*(b))_S \\
&= \sum_{i=0}^{d(S,\text{supp}(\Delta))-1} \left(\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^k \mathrm{d}\theta\right)\Delta \\
&+ \sum_{i=d(S,\text{supp}(\Delta))}^{\infty} \left(\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^k \mathrm{d}\theta\right)\Delta
\end{aligned}$$

where the first terms is zero from Corollary III.2.1, and the second term converges to zero exponentially as $d(S, \text{supp})$ approaches infinity. □

We are now in a position to provide our metric of locality.

**Definition III.2** (Locality)**.** For an optimization problem of the form of problem 1, we define the locality of the problem as

$$\lambda(f, A) = \sup_x \rho(I - A\Sigma(x)A^T). \tag{5}$$

The definition of locality also extends to classes of problems. Explicitly, if it is known that $f \in F$, and $A \in \mathscr{A}$, we define the locality of the class of problems as

$$\lambda(F, \mathscr{A}) = \sup_{f \in F, A \in \mathscr{A}} \lambda(f, A). \tag{6}$$

For example, in network flow problems the class of constraint matrices, $\mathscr{A}$, are those representing flow conservation constraints. The flow conservation constraint at a given node only affects variables for flows departing or arriving at that node; accordingly, if the objective function is separable function of the flow on each edge, the distance metric $d$ corresponds to the shortest-path distance in the network flow graph. As shown in [13], the expression $I - A\Sigma(x)A^T$ reduces to the appropriately weighted graph Laplacian, and $\lambda(F, \mathscr{A})$ can be shown in closed form to equal one [18].

### C. Discussion

The locality of a problem is characterized by $\lambda$ and by the distance metric, $d$. The value of $\lambda$ characterizes the impact of the constraints on components of the optimal solution as a function of the previously defined distance metric. If $\lambda < 1$, this impact decays exponentially at rate $\lambda$, and the problem is said to be *exponentially local*. The locality of a problem naturally characterizes the quantity of problem information necessary to solve for *components* of the optimal solution. The distance metric, $d$, may seem esoteric as it measures the distance between primal variables, which are inherently tied to agents in the networks, and dual variables, which may not have an immediate physical interpretation. However, for many problems of interest, this distance metric can very naturally be translated to one that is physical, primarily if being involved in the same constraints indicates physical proximity. For example, if constraints represent tasks that need to be completed at some location, and only agents within range of that location can complete the task, then the graph distance metric is closely related to geographical distance.

The metrics proposed in this section characterize the locality of a specific instance of an optimization problem. However, in most practical applications, the specific instance of the optimization problem to be solved is *not* known in advance, but is determined at run time by the agents' states and observations, and by the environment itself. Indeed, if the specific optimization problem to be solved was known in advance, there would be no need to solve it in a distributed fashion. Nevertheless, a priori knowledge of the *exact* problem the network will face at the time of execution is often not necessary to take advantage of locality; we can still exploit knowledge of the *class* of problem the system is designed to solve to estimate their locality. As a concrete example, in Section V we validate our theoretical results to an network load-sharing problem where the constraint vector $b$ is determined at run-time. In such a scenario, because the constraint matrix and objective function are fixed, the metric proposed in this section can be directly applied. Even when the constraint matrix or objective function is determined on the fly, the system designer has full knowledge of the map from environment to optimization problems. If all possible problem instances exhibit locality, again, the metric proposed in this section can be directly applied. In the case that enumerating all of these problems is intractable, a sampling-based approach can still be leveraged to estimate the locality of a family of problems. Similarly, computing the locality parameter of a problem comes down to checking the spectral conditions for the entire decision space, which may be infeasible to do so in closed form. In this case, we suggest a sampling-based approach for estimating the locality parameter.

## IV. Exploiting Locality for Cold-Start Optimization

We are now in a position to exploit the locality metrics proposed in Section III to design communication-efficient *distributed optimization algorithms* to approximately solve Problem 1. Specifically, we show that the original optimization problem can be partitioned into independent sub-problems, with the sub-problem size specified by a predetermined bound (chosen as a function of the problem's degree of locality and error tolerance). Locality guarantees that, for a specified error tolerance, solution components of the sub-problems can then be "patched" together to approximately recover the globally optimal solution by computing a correction factor only for components that are near the broken constraints (in the sense of the distance metric of Section III). This gives rise to a two-phase optimization algorithm where the problem is partitioned into sub-problems and solved in the first phase, and the sub-problems are patched together in the second phase.

This section is organized as follows. Sections IV-A and IV-B will be dedicated to the first and second phases of the algorithm respectively. Each of the subsections will begin with motivation and proof of correctness from a *centralized* standpoint (for clarity and ease of notation). We will then comment on how each phase can be implemented in a distributed manner.

We must first define how the quality of an approximate solution to an optimization problem will be quantified. While approximate solutions are typically assessed by the sub-optimality of the objective function, often in practice, the decision variable represents an action to be taken. Motivated by this, we want a handle of the error in the decision variable, and the constraint violations. We say a solution $\hat{x}$ is an $(\varepsilon_x, \varepsilon_C)$ approximation if $\|A\hat{x}-b\|_\infty < \varepsilon_C$, and $\|x^*-\hat{x}\|_2 < \varepsilon_x$. We note that by strict convexity of the objective, the optimal solution is guaranteed to be unique. This not only ensures that our notion of an approximate solution is well-defined, but rules out the case of "jumps" to other optimal solutions.

### A. Phase I: Partitioning and Solving the sub-problems

In this section we show that by ignoring an appropriate subset of the constraints, our original problem can be partitioned into independent sub-problems. We will also show that the solution obtained from solving the sub-problems is an optimal solution for a perturbed version of the original problem. This interpretation is key for making the connection between the "warm-start" scenario presented in [13] (where the algorithm needs to compute $x^*(b+p)$ given the solution $x^*(b)$) to the "cold-start" scenario (where the algorithm must compute $x^*(b)$ from scratch).

Precisely, the following lemma shows that, if removing a subset of the constraints, $C \subseteq V^{(d)}$, and its adjacent edges partitions $G_{\text{eff}}$ into multiple connected components, then the original problem can be partitioned into independent sub-problems.

**Lemma IV.1.** Let $C \subseteq V^{(d)}$ be a set of constraints such that removing the vertices $C$ and its adjacent edges partitions $G_{\text{eff}}$ in connected components, [3] $G_1, \ldots, G_p$. Then,

1) There are no shared primal variables between the connected components: $S_{G_i} \cap S_{G_j} = \emptyset$ if $i \neq j$;
2) We can write the objective function as a sum of additively separable functions: $f(x) = \sum_{i=1}^{p} f_i(x_{S_{G_i}})$.

*Proof Sketch.* The result follows by relating the graph structure of $G_{\text{eff}}$ to the sparsity patterns of $A$ and $\Sigma(x)$, specifically by showing that separability in $G_{\text{eff}}$ implies that the appropriate components of these objects are zero. □

It follows from the previous lemma that the original problem can be partitioned into independent sub-problems given by

$$\begin{aligned} &\underset{x \in \mathbb{R}^N}{\text{minimize}} && f_i(x_{S_{G_i}}) \\ &\text{subject to} && A_{C_i, S_{G_i}} x_{S_{G_i}} = b_{G_i} \end{aligned} \tag{7}$$

We relate the solution of this problem to our original problem by showing that the solution obtained is the solution to a perturbed variant of our original problem.

**Lemma IV.2** (Implicit Constraints)**.** Let $C \subseteq V^{(d)}$ and let $\hat{x}^*$ be the minimizer of

$$\begin{aligned} &\underset{x \in \mathbb{R}^N}{\text{minimize}} && f(x) \\ &\text{subject to} && A_{-C,*} x = b_{-C} \end{aligned} \tag{8}$$

If $\hat{b} = A\hat{x}^*$, then $\hat{x}^*$ is the minimizer of

$$\begin{aligned} &\underset{x \in \mathbb{R}^N}{\text{minimize}} && f(x) \\ &\text{subject to} && Ax = \hat{b} \end{aligned} \tag{9}$$

*Proof Sketch.* The result follows from showing that the feasible set of Problem (9) is a subset of the feasible set of Problem (8). □

Full proofs of Lemmas IV.1 and IV.2 are reported in the Supplementary Material.

*1) Distributed Implementation of Phase I:* To implement phase I in a distributed manner we need to generate the constraint cut-set, $C$, in a distributed manner. This can be accomplished using the algorithm of Linial and Saks [19] for weak-diameter graph decomposition as a subroutine to cluster the constraints and elect leaders for each cluster. An overview of the algorithm is included in the Supplementary Material for completeness. The algorithm of Linial and Saks generates a subset of the constraints $\tilde{C} \subset V^{(d)}$ and assigns a leader $l(u) \in V^{(d)}$ for each $u \in \tilde{C}$, such that, if $u, v \in V(d)$ are assigned to different leaders, then there is no edge between them in $G_{\text{eff}}$. The connected components $G_1, \ldots, G_p$ as referenced in Section IV are simply the constraints that have been assigned the same leader, and $C = V^{(d)} \setminus \tilde{C}$ is implicitly the set of constraints that are not assigned to any cluster. The leader for each cluster solves the cluster's sub-problem and informs agents in its cluster of their own optimal solution values.

We let $x^{(k)}$ be the aggregate of privately known solution components at iteration $k$, and initialize $x^{(0)}$ with the solution of the sub-problems. We define $b^{(0)} = Ax^{(0)}$ to be the implicit constraints in the first phase.

### B. Phase II: Patching

In the previous section, we obtained the solution to a perturbed variant of our original problem. In this phase, we will need to provide a correction factor to drive the solution of the partitioned problem to that of the target

[3] We say $v_i^{(d)} \in V^{(d)}$, and $v_j^{(d)} \in V^{(d)}$ are in different connected components of $\tilde{G}_{\text{eff}}$ if there is no path from one to the other

problem. Critically, we will show that this correction factor can be computed efficiently and in a distributed manner if our problem is exponentially local.

It follows from Lemma IV.2 that $x^{(0)}$ is the minimizer of

$$\begin{aligned} &\underset{x\in\mathbb{R}^N}{\text{minimize}} && f(x) \\ &\text{subject to} && Ax = b^{(0)} \end{aligned} \tag{10}$$

If the problem exhibits locality, we can then express the optimal solution as the solution of the partitioned problem plus a correction factor. Explicitly,

$$x^*(b) = x^*(\hat{b}) + \sum_{i=0}^{\infty}\left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^i d\theta\right)\Delta \tag{11}$$

where $\Delta = b-\hat{b}$, and $x_\theta := x^*(b+\theta\Delta)$. Thanks to locality, we can approximate $\hat{x}^*(b) \approx x^*(b)$ as

$$\hat{x}^*(\hat{b}+\Delta) = x^*(b) + \sum_{i=0}^{K}\left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^i d\theta\right)\Delta \tag{12}$$

where $K$ represents the number of terms of the Neumann sum used.

**Theorem IV.3.** The solution generated from taking the $K$-term truncation of $\hat{x}^*(b)$ is an $\left(\frac{\sup_x\|\Sigma(x)A^T\|}{1-\lambda}\lambda^{K+1}\|\Delta\|, \frac{1+\lambda}{(1-\lambda)\sqrt{m}}\lambda^{K+1}\|\Delta\|\right)$ approximation.

*Proof Sketch.* The proof of the error term in $x$ follows directly from applying the definition of locality to the truncated approximation of $\hat{x}^*(b)$. A similar error analysis can be applied to $Ax^*(b)$ to recover the error in the constraints. □

From here on, we will refer to the number of terms in the truncation, $K$, as the "radius of repair", corresponding to the distance around $\Delta$ that we compute corrections for.

Theorem IV.3 quantifies the trade-off between a number of algorithmic design choices and solution accuracy, primarily the size of the sub-problems chosen in the first phase, through $\|\Delta\|$, and communication in the second phase, through $K$. Both the relationship between $\|\Delta\|$ and the size of sub-problems, and the dependence of communication volume on $K$ are problem specific. In Section V, we provide an example assessing both of these trade-offs for a network load-sharing problem.

*1) Distributed Implementation of Phase II:* The key insight that will allow us to implement the correction step in a distributed manner is that correction in Equation (11) can be represented as a series of sequential updates rather than a single update; these updates will be the target values for each iteration of the algorithm. By continuity of $\Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^k$, the path over which we integrate does not matter, and the sequence of updates

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} + \sum_{i=0}^{\infty}\left(\int_0^1 \Sigma(\hat{x}_\theta^{(k)})A^T(I-A\Sigma(\hat{x}_\theta^{(k)})A^T)^i d\theta\right)\Delta^{(k)} \tag{13}$$

converges to $x^*(b)$ within $|\text{supp}(\Delta)|$ iterations, where $\hat{x}_\theta^{(k)} = \hat{x}^*(b^{(k)}+\theta\Delta^{(k)})$, $b^{(k)} = A\hat{x}^{(k)}$, $\Delta^{(k)} = (b-b^{(0)})_{S^{(k)}}$, and $\{S^{(k)}\}$ partitions the support, $\text{supp}(b-b^{(0)}) = C$. We let $x^{(k)}$ be the approximation to $\hat{x}^{(k)}$ by truncating the Neumann series to $R_k$ terms. These updates can be interpreted as traversing the optimal surface to iteratively drive $b^{(0)}$ to $b$. While the support of $\sum_{i=0}^{K}\left(\int_0^1 \Sigma(\hat{x}_\theta)A^T(I-A\Sigma(\hat{x}_\theta)A^T)^i d\theta\right)\Delta$ may span the entire network, the sequential updates circumvent this by localizing the support of each of the truncated updates $\sum_{i=0}^{R_k}\left(\int_0^1 \Sigma(\hat{x}_\theta^{(k)})A^T(I-A\Sigma(\hat{x}_\theta^{(k)})A^T)^i d\theta\right)\Delta^{(k)}$ around $\Delta^{(k)}$.

For ease of presentation, in the remainder of this section we will assume $f$ is quadratic so $\Sigma(x) = \Sigma$ is constant. Truncating the approximation at each iteration introduces drift away from the optimal surface. Because this also introduces error in the integrand, the total error at the end is not simply the sum of the errors made at each step. Straightforward, but tedious, modifications to our analysis can be made to account for this drift. The distributed implementation of the second phase is presented in Algorithn 1. It circumvents the fact that we cannot use a priori knowledge of $\|\Delta\|$ to compute the number of terms to include in the truncation. The algorithm operates by iteratively including more terms and testing for constraint violations. While an upper bound on $\left\|\Sigma A^T\right\|$ can be computed along with $\lambda$, $|I^{(0)}|$ will need to be estimated, and in general will depend on how the cut-set, $C$ is generated. We refer the reader to the Supplementary Material for an example illustrating the estimation of $|I^{(0)}|$ based on the mechanism for partitioning in phase I.

**Algorithm 1:** Phase II—Patching Phase

**input:** $I^{(0)}, x^{(0)}, b^{(0)}$ from Phase I
**while** $I^{(k)} \neq \emptyset$ **do**
  $R \leftarrow \min\left\{r : \frac{\left\|\Sigma A^T\right\|}{1-\lambda}\lambda^{r+1}\left\|\Delta^{(k)}\right\| < \frac{\varepsilon_x}{|I^{(0)}|}\right\}$;
  $\tilde{x}^{(k+1)} = x^{(k)} + \sum_{i=0}^{R}\left(\int_0^1 \Sigma A^T(I-A\Sigma A^T)^k d\theta\right)\Delta^{(k)}$ ;
  **if** $\{i \in V^{(d)} : |(A\tilde{x}^{(k+1)}-b)_i| \geq \varepsilon\} \subsetneq I^{(k)}$ **then**
    $x^{(k+1)} \leftarrow \tilde{x}^{(k+1)}$, $k \leftarrow k+1$;
  **else**
    $R \leftarrow R+1$, go to 3;
  **end**
**end**

**Theorem IV.4** (Suboptimality Bounds of the Patching Phase)**.** If $|I^{(0)}|$ is known, Algorithm 1 is guaranteed to generate an $(\varepsilon_x, \varepsilon_C)$ approximate solution within $|I^{(0)}|$ outer iterations.

*Proof Sketch.* The proof follows from the observation that that the algorithm will not terminate until the constraint bound is satisfied. The $x$ error bound follows from lower-bounding the radius of repair for each iteration and applying Theorem IV.3. □

By applying Corollary III.2.1, we can characterize the support of $\sum_{i=0}^{R}\left(\int_0^1 \Sigma A^T(I-A\Sigma A^T)^k d\theta\right)\Delta^{(k)}$. This will allow us to efficiently implement Algorithm 1 by only exchanging necessary information for each update. Specifically, note that $[\tilde{x}^{(k+1)}-x^{(k)}]_j = 0$ if $d(v_j^{(p)}, \text{supp}(\Delta^{(k)})) > R$ so these solution components do not need to be updated. Similarly, because $A\tilde{x}^{(k+1)} =: \tilde{b}^{(k+1)}$,

$$\tilde{b}^{(k+1)} = Ax^{(k)} + A\sum_{i=0}^{R}\left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^i d\theta\right)\Delta^{(k)}.$$

This implies that if $d_{G_{\text{eff}}}(v_i^{(d)}, \text{supp}(\Delta)) \geq K+2$ then $\tilde{b}_i^{(k+1)} = b_i^{(k)}$. Consequently, it suffices to calculate $A_{B(\text{supp}(\Delta),K+1),*}x$

where $B(C,k) := \{v_i^{(d)} | d_{G_{\text{eff}}}(v_i^{(d)}, C) \leq k\}$, where only the values of $\{v_i^{(p)} | d(v_i^{(p)}, \text{supp}(\Delta)) \leq K+1\}$ are necessary for calculating $A_{B(\text{supp}(\Delta),K+1),*}x$. Finally, the entirety of $\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^i d\theta$ does not need to be explicitly computed; instead $\left(\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^i d\theta\right)\Delta^{(k)}$ should be computed via a series of sparse matrix vector products that only require information from nodes $v_i^{(p)}$ where $d(v_i^{(p)}, \text{supp}(\Delta^{(k)})) \leq R$.

We presented a leader-election based algorithm for the first phase of the algorithm, however, other methods can also be applied. In the Supplementary Material, we discuss scaling of the distributed subgradient method with network size for our numerical example, and comment on how locality can be used to reduce the communication volumne needed by the distributed subgradient method.

## V. Experiments

In this section, we use a simplified example of network load-sharing to validate our theoretical results. We simulate the distributed algorithm of Section IV on problem instances of varying locality to demonstrate the effect of locality on algorithm convergence and to assess the tightness of our bounds. We also demonstrate how some of the algorithm design trade-offs discussed in Section IV can be assessed for this problem setting.

### A. Problem Setting

We consider a setting where agents are positioned in a grid of size $M \times N$, and the 4 agents bordering each grid cell need to share the load generated in their grid cell $\mathscr{L}(i)$. The loads could represent a joint task that needs to be completed, or a resource the needs to be stored. There is a preference for equitably distributing each load, as well as not overloading any one agent. We encode these preferences in the objective function $f(x) = \frac{\alpha}{2}\sum_i \left(\sum_{j\in\mathscr{N}(i)} x_{i,j}\right)^2 + \frac{\beta}{2}\sum_i \sum_{j\in\mathscr{N}(i)} x_{i,j}^2$, where we let $i$ index the agents, and $j$ the loads, and the variable $x_{i,j}$ captures the assignment. With some abuse of notation we let $\{j \in \mathscr{N}(i)\}$ be the set of loads that agent $i$ can service, and $\{i \in \mathscr{N}(j)\}$ be the set of agents who can service load $j$. The following optimization problem describes the system objectives and constraints

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{\alpha}{2}\sum_i \left(\sum_{j\in\mathscr{N}(i)} x_{i,j}\right)^2 + \frac{\beta}{2}\sum_i \sum_{j\in\mathscr{N}(i)} x_{i,j}^2 \\ \text{subject to} \quad & \sum_{i\in\mathscr{N}(j)} x_{i,j} = \mathscr{L}_j, \forall j \end{aligned} \tag{14}$$

For a fixed maximum sub-problem size, $m \times n$, we partition the original problem into sub-problems by tiling the $(M-1)\times(N-1)$ grid of constraints with sub-grids of size $m \times n$, and ignoring any constraints between tiles. The resulting sub-problems are fully decoupled and can be solved in parallel by an elected leader in each partition. Note that only the constraints that were originally ignored are violated at this step and consequently are the only ones that need to be repaired. Also note that, if $\alpha = 0$, the problem fully decouples and the optimal solution is given by splitting each load evenly between its servicing agents. The parameters $\alpha$ and $\beta$ allow us to tune the locality of the problem and investigate the performance of our algorithm for various rates of locality.

### B. Effect of Locality on Convergence

In this example, we fixed the dimension of the global problem to be $36 \times 36$ and allowed maximum sub-problems of size $5 \times 5$. This results in the original problem being partitioned into 36 sub-problems. We fixed $\beta = 3$ and let $\alpha$ range from 0.5 to 3.5 by increments of 0.1. The locality parameter was empirically calculated and was found to range from 0.33 to 0.76. Figure 1 plots error in the optimization variable versus radius of repair for varying locality parameters. In addition to showing convergence as the radius grows to encompass all agents, the plot indicates that solution accuracy depends heavily on the locality parameter—in other words, the amount of communication required to solve a distributed optimization problem is directly related to the locality metric we have proposed.

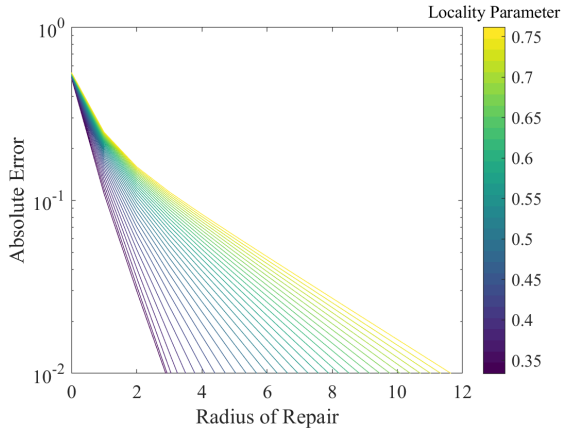


Fig. 1: Effect of locality parameter on convergence in the patching phase of the optimization algorithm.

In Figure 2, we compare our theoretical predictions to the true behavior of the locality aware algorithm for a representative sample of problem instances. In all cases, our theoretical bounds on convergence are tight.

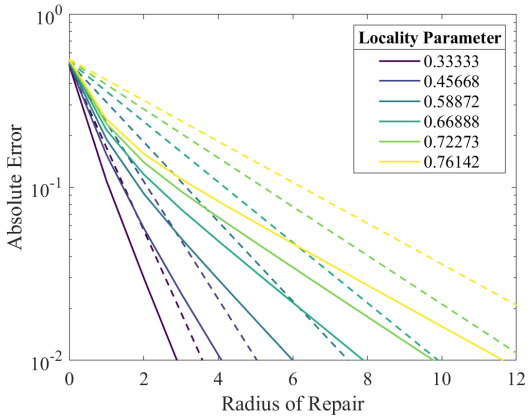


Fig. 2: The theoretical convergence rate is plotted as a dotted line and the corresponding true convergence rate is plotted as a solid line in the same color.

### C. Phase I and Phase II Trade-offs

We now illustrate how one can assess the algorithmic trade-off between cluster size and required number of patching iterations presented in Section IV, so as to give the reader a sense for how similar analysis can be carried out on their

problem. We consider the network load-sharing problem on both a $36 \times 36$ grid (which we will refer to as the "square grid") and a $36 \times 2$ grid (the "long grid"). For both problems, we fix $\alpha = 1, \beta = 3$, and sweep the maximum sub-problem size from $2 \times 2$ to $35 \times 35$. Figure 3 shows the convergence of the patching phase for both the square and the long grid.

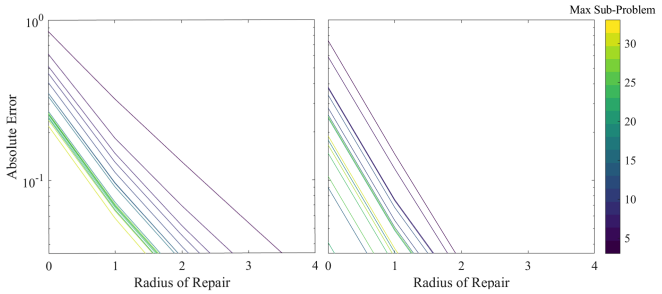


Fig. 3: Convergence of the patching phase on the square grid (left) and long grid (right).

Notably, the plot shows that solution accuracy in the first phase is not monotonic with the maximum sub-problem size. This is a direct consequence of the fact that how "tight" a constraint is dictates how solution accuracy is affected by cutting it. While there are methods for generating sparse cuts in a distributed manner [20], to the best of our knowledge, generating "loose" cuts is an unexplored problem. Such an algorithm, however, could dramatically improve our locality-aware algorithm, and we highlight it as a potential future direction.

Crucially, the same radius of repair for the square grid and long grid do not equal the same volume of communication. On the square grid, a one unit increase in the radius of repair results in a factor of 4 *multiplicative* increase in the communication volume. In contrast, on the long grid, a one unit increase in the radius of repair results in an *additive* increase of 4 times the number of broken constraints. The interpretation of the radius of repair depends closely on the structure of the underlying problem. Such structure should be carefully evaluated when assessing the trade-off between the size of sub-problems solved in the first phase and the communication volume needed in the second phase.

## VI. Conclusion

We have studied the structure of linearly constrained convex optimization problems and provided a method of tracking the cascading effects of a perturbation of the remainder of the network. This gave rise to a notion of locality suggesting that certain global optimization problem with "local" structure can be solved on much smaller scales. We applied this notion of locality to design a distributed optimization algorithm that explicitly takes advantage of this fact. We validated our results on a network load-sharing problem, and provided an example of how one could assess the trade-offs between some of the free parameters in the algorithm.

The framework of locality presented in this paper motivates further investigation for a number of interesting questions:

- Once we have quantified the importance of problem information to solution components, how can we use such knowledge to systematically design optimal communication protocols?
- How can we optimally, and in a distributed manner, choose the constraint cut-sets to balance the size of sub-problems solved in the first phase of the algorithm and the amount of communication needed in the second phase?
- How can locality be used to improve the efficiency of existing distributed optimization algorithms?

## Acknowledgments

Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## References

[1] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization," *arXiv e-prints*, p. arXiv:1709.08765, Sep 2017.

[2] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, April 2010.

[3] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra," *SIAM J. on Optimization*, vol. 25, no. 2, pp. 944–966, May 2015. [Online]. Available: https://doi.org/10.1137/14096668X

[4] A. Chen and A. Ozdaglar, "A fast distributed proximal-gradient method," *CoRR*, vol. abs/1210.2289, 2012. [Online]. Available: http://arxiv.org/abs/1210.2289

[5] D. Jakovetić, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[6] K. Srivastava and A. Nedić, "Distributed asynchronous constrained stochastic optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 772–790, Aug 2011.

[7] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *2013 IEEE 52nd Annual Conference on Decision and Control, CDC 2013*. United States: Institute of Electrical and Electronics Engineers Inc., 2013, pp. 6855–6860.

[8] K. I. Tsianos and M. G. Rabbat, "Distributed consensus and optimization under communication delays," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2011, pp. 974–982.

[9] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, pp. 5453–5458.

[10] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, March 2012.

[11] D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-raphson consensus for distributed convex optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 994–1009, April 2016.

[12] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, Jan 2017.

[13] P. Rebeschini and S. Tatikonda, "Locality in network optimization," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 487–500, June 2019.

[14] C. C. Moallemi and B. Van Roy, "Convergence of min-sum message-passing for convex optimization," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 2041–2050, April 2010.

[15] R. A. Brown, F. Rossi, K. Solovey, M. T. Wolf, and M. Pavone. (2020) Exploiting locality and structure for distributed optimization in multi-agent systems (extended version). Available at http://asl.stanford.edu/wp-content/papercite-data/pdf/Brown.Rossi.ea.ECC20.pdf.

[16] C. A. Bererton, "Multi-robot coordination and competition using mixed integer and linear programs," Ph.D. dissertation, 2004. [Online]. Available: https://search.proquest.com/docview/305204035?accountid=14026

[17] S. H. Low and D. E. Lapsley, "Optimization flow control. i. basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, Dec 1999.

[18] D. Spielman, "Spectral graph theory," *Lecture Notes, Yale University*, pp. 740–0776, 2009.

[19] N. Linial and M. Saks, "Low diameter graph decompositions," *Combinatorica*, vol. 13, no. 4, p. 441–454, 1993.

[20] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 1–26, 2013. [Online]. Available: https://doi.org/10.1137/080744888

[21] T. Davis, *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718881
[22] J.-J. Climent, N. Thome, and Y. Wei, "A geometrical approach on generalized inverses by neumann-type series," *Linear Algebra and Its Applications - LINEAR ALGEBRA APPL*, vol. 332, pp. 533–540, 08 2001.

## VII. Appendix

### A. Sparse Cholesky Factorization and the Solution to Sparse Linear Systems

The sparsity pattern of $A\Sigma(x^*(b))A^T$ can be characterized in closed form by leveraging techniques typically used to accelerate sparse linear solvers.

By assumption, $f$ is strongly convex and twice continuously differentiable so $\nabla^2 f(x)$ is positive definite and has a unique Cholesky factorization $L(x)L(x)^T = \nabla^2 f(x)$, where $L(x)$ is a lower triangular matrix with real and positive diagonal entries. Noting that $A\Sigma(x)A^T = (L(x)^{-1}A^T)^T(L(x)^{-1}A^T)$.

**Lemma VII.1** (Sparsity Structure of the Cholesky Factorization)**.** For a Cholesky factorization $L(x)L(x)^T = \nabla^2 f(x)$, neglecting numerical cancellation,

- If $[\nabla^2 f(x)]_{ij} \neq 0$ then $L(x)_{ij} \neq 0$ ;
- For indices $i < j < k$, $L(x)_{ji} \neq 0$, and $L(x)_{ki} \neq 0$ then $L(x)_{kj} \neq 0$.

Figure 4 depicts the fill-in structure of the sparse Cholesky factorization.

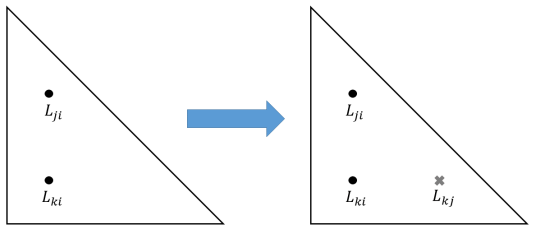


Fig. 4: Fill-in structure of the sparse Cholesky factorization

For a given matrix lower triangular matrix $L \in \mathbb{R}^{N\times N}$, we define the directed graph $G_L(V_L, E_L)$ with nodes $V_L = [N]$ and edges $E_L = \{(j,i) : l_{ij} \neq 0\}$. Let $\text{Reach}_L(i)$ denotes the set of nodes reachable from node i via paths in $G_L$, and let $\text{Reach}_L(S)$ for subset $S \subseteq [N]$ be defined as $\text{Reach}_L(S) = \bigcup_{i\in S} \text{Reach}_L(i)$

**Lemma VII.2** (Support of the Solution to a Sparse Linear System)**.** The support, $\text{supp}(x) := \{j : x_j \neq 0\}$, of the solution $x$ to the sparse linear system $Lx = b$ is given by $\text{supp}(x) = \text{Reach}_L(\text{supp}(b))$

We refer the reader to [21] for the proofs of Lemmas VII.1 - VII.2. The sparsity patterns of both $L(x)^{-1}A^T$ and $A\Sigma(x)A^T$ can be derived as immediate consequences of Lemma VII.2.

**Lemma VII.3.** Neglecting numerical cancellation, $\text{supp}([L(x)^{-1}A^T]_{*,i}) = \text{Reach}_{L(x)}(S_i)$. Furthermore, $[A\Sigma(x)A^T]_{ij} \neq 0$ if $\text{Reach}_{L(x)}(S_i) \cap \text{Reach}_{L(x)}(S_j) \neq \emptyset$

### B. Section III Proofs

**Theorem** (III.2)**.** For $k \in \mathbb{Z}_+$, neglecting numerical cancellation,

$$\text{supp}((I - A\Sigma(x)A^T)^k) = \{(i,j) | d_{G_{\text{eff}}(x)}(v_i, v_j) \leq k\} \subseteq \{(i,j) | d_{G_{\text{eff}}}(v_i, v_j) \leq k\}.$$

*Proof.* For ease of notation, we let $M = I - A\Sigma(x)A^T$, so $M_{uv} \neq 0$ if and only if $(u,v) \in E_{\text{eff}}(x)$. An edge constitutes a path of length one so the result hold true for $k = 1$.

We now proceed by induction. Suppose for all $j \leq k$, $\text{supp}((I - A\Sigma(x)A^T)^j) = \{(u,v) | d_{G_{\text{eff}}(x)}(u,v) \leq j\}$ Then $M^{k+1} = MM^k$, and $M^{k+1}_{uv} = M_{u*}M^k_{*v}$. Thus, if $M^{k+1}_{uv} \neq 0$, there exists $w$ such that $M_{uw} \neq 0$ and $M^k_{wv} \neq 0$. Consequently, $(u,w) \in E_{\text{eff}}(x)$ and there is a path of length at most $k$ from $w$ to $v$. We concatenate these paths to find a path of length at most $k+1$ from $u$ to $v$. Similarly, if there is a path from $u$ to $v$ of length at most $k+1$ in $G_{\text{eff}}$ then letting $w$ be the first vertex after $u$ along this path, there is an edge from $u$ to $w$ and there is a path of length at most $k$ from $w$ to $v$. Thus, $M_{uw} \neq 0$ and $M^k_{wv} \neq 0$, so neglecting numerical cancellation $M_{u*}M^k_{*v} = M^{k+1}_{uv} \neq 0$. Taking the the union over all $x$, $G_{\text{eff}} := (V^{(d)}, \bigcup_{x\in D} E_{\text{eff}}(x))$, yields our result. □

**Theorem** (III.3)**.** A linearly constrained convex optimization problem is exponentially local with parameter $\lambda$ if $\sup_x \rho(I - A\Sigma(x)A^T) = \lambda < 1$, where $\rho(M)$ denotes the largest singular value of the matrix $M$.

*Proof.* If $\rho(I - A\Sigma(x)A^T) < 1$ then $\sum_{i=0}^{\infty}(I - A\Sigma(x)A^T)^k$ converges [22]. Furthermore, $A\Sigma(x)A^T$ is invertible with inverse $(A\Sigma(x)A^T)^{-1} = \sum_{i=0}^{\infty}(I - A\Sigma(x)A^T)^k$. We can rewrite Equation (2)

$$\begin{aligned}\frac{dx^*(b)}{db} &= \Sigma(x^*(b))A^T(A\Sigma(x^*(b))A^T)^{-1} \\ &= \Sigma(x^*(b))A^T \sum_{i=0}^{\infty}(I - A\Sigma(x^*(b))A^T)^i\end{aligned}$$

It is important that Equation (2) is based on Hadamard's global inverse function theorem (rather than the implicit function theorem, which holds only locally). The implication is that the derivative of the optimal point is continuous *everywhere* along the subspace $\text{Im}(A)$. This allows us to apply the fundamental theorem of calculus allows us to integrate through this expression to determine sensitivity of the optimal point of finite perturbations in the constraint vector. Formally,

$$\begin{aligned}x^*(b+\Delta) - x^*(b) &= \left(\int_0^1 \frac{dx^*(b+\theta\Delta)}{d\theta} d\theta\right)\Delta \\ &= \left(\int_0^1 \Sigma(x_\theta)A^T(A\Sigma(x_\theta)A^T)^{-1} d\theta\right)\Delta \\ &= \left(\int_0^1 \Sigma(x_\theta)A^T \sum_{i=0}^{\infty}(I - A\Sigma(x_\theta)A^T)^k d\theta\right)\Delta \\ &= \sum_{i=0}^{\infty}\left(\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^k d\theta\right)\Delta\end{aligned}$$

where $x_\theta := x^*(b+\theta\Delta)$.

$$\begin{aligned}&\left\|\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^k d\theta\right\| \\ &\leq \sum_{i=0}^{\infty}\left\|\int_0^1 \Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^k d\theta\right\| \\ &\leq \int_0^1 \left\|\Sigma(x_\theta)A^T(I - A\Sigma(x_\theta)A^T)^k\right\| d\theta \\ &\leq \int_0^1 \|\Sigma(x_\theta)A^T\| \left\|(I - A\Sigma(x_\theta)A^T)^k\right\| d\theta\end{aligned}$$

$$
\begin{aligned}
&\leq \int_0^1 \left\|\Sigma(x_\theta)A^T\right\| \rho(I-A\Sigma(x_\theta)A^T)^k d\theta \\
&\leq \sup_x \left\|\Sigma(x)A^T\right\| \int_0^1 \rho(I-A\Sigma(x_\theta)A^T)^k d\theta \\
&\leq \sup_x \left\|\Sigma(x)A^T\right\| \int_0^1 \sup_x \rho(I-A\Sigma(x)A^T)^k d\theta \\
&\leq \sup_x \left\|\Sigma(x)A^T\right\| \sup_x \rho(I-A\Sigma(x)A^T)^k \\
&\leq \sup_x \left\|\Sigma(x)A^T\right\| \sup_x \rho(I-A\Sigma(x)A^T)^k \\
&\leq \sup_x \left\|\Sigma(x)A^T\right\| \lambda^k
\end{aligned}
$$

By strong convexity of $f$, $\sup_x \left\|\Sigma(x)A^T\right\|$ exists, and is finite. From Corollary III.2.1,

$$
\begin{aligned}
&(x^*(b+\Delta)-x^*(b))_S \\
&= \sum_{i=d(S,\mathrm{supp}(\Delta))}^{\infty} \left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^k d\theta\right)\Delta \quad (15)
\end{aligned}
$$

We take the norm of the perturbed solution to conclude the proof.

$$
\begin{aligned}
&\|(x^*(b+\Delta)-x^*(b))_S\| \\
&= \left\| \sum_{i=d(S,\mathrm{supp}(\Delta))}^{\infty} \left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^k d\theta\right)\Delta \right\| \\
&\leq \left\| \sum_{i=d(S,\mathrm{supp}(\Delta))}^{\infty} \left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^k d\theta\right) \right\| \|\Delta\| \\
&\leq \sum_{i=d(S,\mathrm{supp}(\Delta))}^{\infty} \left\| \left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^k d\theta\right) \right\| \|\Delta\| \\
&\leq \sum_{i=d(S,\mathrm{supp}(\Delta))}^{\infty} \sup_x \left\|\Sigma(x)A^T\right\| \lambda^k \|\Delta\| \\
&= \sup_x \left\|\Sigma(x)A^T\right\| \|\Delta\| \sum_{i=d(S,\mathrm{supp}(\Delta))}^{\infty} \lambda^k \\
&= \frac{\sup_x \left\|\Sigma(x)A^T\right\|}{1-\lambda} \|\Delta\| \lambda^{d(S,\mathrm{supp}(\Delta))}
\end{aligned}
$$

□

### C. Section IV Proofs

**Lemma** (IV.1)**.** Let $C \subseteq V^{(d)}$ be a set of constraints such that removing the vertices $C$ and its adjacent edges partitions $G_{\text{eff}}$ in connected components [4] $G_1, \ldots, G_p$.

1) There are no shared primal variables between the connected components: $S_{G_i} \cap S_{G_j} = \emptyset$ if $i \neq j$
2) We can write the objective function as a sum of additively separable functions: $f(x) = \sum_{i=1}^p f_i(x_{S_{G_i}})$

*Proof.* Note that if $v_i^{(d)} \in G_i$ and $v_j^{(d)} \in G_j$ are in different connected components of $\tilde{G}_{\text{eff}}$ then there cannot be an edge between $v_i^{(d)}$ and $v_j^{(d)}$, so $[A_{-C,*}\Sigma(x)A_{-C,*}^T]_{ij} = 0$. Because the diagonal elements of $\nabla^2 f(x)$ are strictly positive, if $S_{G_i} \cap S_{G_j} \neq \emptyset$ then $[A_{-C,*}\Sigma(x)A_{-C,*}^T]_{ij} \neq 0$. This holds true for all $i \neq j$ and for all $v_i^{(d)} \in G_i$ and $v_j^{(d)} \in G_j$ so $S_{G_i} \cap S_{G_j} = \emptyset$ if $i \neq j$.

[4] We say $v_i^{(d)} \in V^{(d)}$, and $v_j^{(d)} \in V^{(d)}$ are in different connected components of $\tilde{G}_{\text{eff}}$ if there does not exist a path from one to the other

Without loss of generality, let $i > j$. We will show that if $[\nabla^2 f(x)]_{ij} \neq 0$ then there is an edge between $v_i^{(d)}$ and $v_j^{(d)}$ in $\tilde{G}_{\text{eff}}$. Note that if $[\nabla^2 f(x)]_{ij} \neq 0$ then $L(x)_{ij} \neq 0$. Additionally, $L_{ii} > 0$ and $L_{jj} > 0$. Let $v_i^{(d)} \in G_i$ and $v_j^{(d)} \in G_j$ be such that $v_i^{(p)} \in S_{v_i^{(d)}}$ and $v_j^{(p)} \in S_{v_j^{(d)}}$. If $L(x)_{ij} \neq 0$ then $[L^{-1}A^T_{v_j^{(d)},*}]_i \neq 0$. Note that $[L^{-1}A^T_{v_i^{(d)},*}]_i \neq 0$ as well so $[A_{-C,*}\Sigma(x)A^T_{-C,*}]_{ij} \neq 0$ and there is an edge between $v_i^{(d)}$ and $v_j^{(d)}$ in $\tilde{G}_{\text{eff}}$. If $v_i^{(d)}$ and $v_j^{(d)}$ are in different connected components in $\tilde{G}_{\text{eff}}$, there cannot be an edge between them. Consequently, $[\nabla^2 f(x)]_{ij} = 0$ and we can separate $f(x) = f(S_1) + f(S_2)$ where $S_1 \cap S_2 = \emptyset$ and $S_{v_i^{(d)}} \subseteq S_1, S_{v_j^{(d)}} \subseteq S_2$. Inducting on all pairs of $G_i$ and $G_j$ proves our result. □

**Lemma** (IV.2)**.** Let $C \subseteq V^{(d)}$ and let $\hat{x}^*$ be the minimizer of

$$
\begin{aligned}
&\text{minimize} \quad f(x) \\
&x \in \mathbb{R}^N \\
&\text{subject to} \quad A_{-C,*}x = b_{-C}
\end{aligned} \quad (16)
$$

and $\hat{b} = A\hat{x}^*$. Then, $\hat{x}^*$ is the minimizer of

$$
\begin{aligned}
&\text{minimize} \quad f(x) \\
&x \in \mathbb{R}^N \\
&\text{subject to} \quad Ax = \hat{b}
\end{aligned} \quad (17)
$$

*Proof.* Note that on $V^{(d)} \setminus C$, the implicit constraints are equal to the true constraints. Precisely, $b_{-C} = \hat{b}_{-C}$.

The constraints in Problem (8) are a subset of the constraints in Problem (9). Therefore, the feasible set of Problem (9) is contained in the feasible set of Problem (8). Explicitly,

$$
\begin{aligned}
\{x | Ax = \hat{b}\} &= \{x | A_{-C,*}x = b_{-C}, A_{C,*}x = \hat{b}_C\} \\
&\subseteq \{x | A_{-C,*}x = b_{-C}.\}
\end{aligned}
$$

Therefore, if $\hat{x}^*$ is not optimal for Problem (9), it is also suboptimal for Problem (8). □

**Theorem** (IV.3)**.** The solution generated from taking the $K$-term approximation of $\hat{x}^*(b)$ is an $\left(\frac{\sup_x \left\|\Sigma(x)A^T\right\|}{1-\lambda}\lambda^{K+1}\|\Delta\|, \frac{1+\lambda}{(1-\lambda)\sqrt{m}}\lambda^{K+1}\|\Delta\|\right)$ approximation.

*Proof.* The error induced by truncating the sum in Equation (11) can be expressed as

$$
\begin{aligned}
\delta &= x^*(b+\Delta) - \hat{x}^*(b+\Delta) \\
&= \sum_{i=K+1}^{\infty} \left(\int_0^1 \Sigma(x_\theta)A^T(I-A\Sigma(x_\theta)A^T)^i d\theta\right)\Delta.
\end{aligned}
$$

The magnitude of the error in the optimal solution can be bounded as

$$
\|\delta\| \leq \frac{\sup_x \left\|\Sigma(x)A^T\right\|}{1-\lambda}\lambda^{K+1}\|\Delta\| \quad (18)
$$

It follows that the truncation error converges exponentially to zero at a rate of $\lambda$; the errors in the constraints and optimization variable both converge to zero exponentially as well. Similarly, the constraint error is given by $A\delta$. Taking the norm,

$$
\begin{aligned}
\|A\delta\|_\infty &\leq \frac{\sup_x \left\|A\Sigma(x)A^T\right\|_\infty}{1-\lambda}\lambda^{K+1}\|\Delta\| \\
&\leq \frac{\sup_x \left\|A\Sigma(x)A^T\right\|_2}{(1-\lambda)\sqrt{m}}\lambda^{K+1}\|\Delta\| \\
&\leq \frac{1+\lambda}{(1-\lambda)\sqrt{m}}\lambda^{K+1}\|\Delta\|
\end{aligned}
$$

yields the result. □

**Theorem** (IV.4)**.** Algorithn 1 is guaranteed to generate an $(\varepsilon_x, \varepsilon_C)$ approximate solution within $|I^{(0)}|$ outer iterations.

*Proof.* At iteration $k$, we define $x^{(k)}$ to be the aggregate of privately known solution components, $b^{(k)} = Ax^{(k)}$ to be the implicit constraints, and $I^{(k)} = \{i \in [M] : |(b^{(k)} - b)_i| \geq \varepsilon\}$ to be the violation set. Picking any element of our violation set $i^{(k)} \in I^{(k)}$, our goal is to drive $b^{(k+1)}_{i^{(k)}}$ to $b_{i^{(k)}}$, so we let $\Delta^{(k)} = (b_i - b^{(k)}_{i^{(k)}})e_{i^{(k)}}$, and define the following

$$\hat{x}^{(k+1)} = x^{(k)} + \sum_{i=0}^{\infty} \left( \int_0^1 \Sigma(x_\theta^{(k)}) A^T (I - A\Sigma(x_\theta^{(k)})A^T)^i d\theta \right) \Delta^{(k)}$$

$$x^{(k+1)} = x^{(k)} + \sum_{i=0}^{K} \left( \int_0^1 \Sigma(x_\theta^{(k)}) A^T (I - A\Sigma(x_\theta^{(k)})A^T)^i d\theta \right) \Delta^{(k)}$$

$$\delta^{(k+1)} = \sum_{i=K+1}^{\infty} \left( \int_0^1 \Sigma(x_\theta^{(k)}) A^T (I - A\Sigma(x_\theta^{(k)})A^T)^i d\theta \right) \Delta^{(k)}$$

where $x_\theta^{(k)} := x^*(b^{(k)} + \theta\Delta^{(k)})$. Intuitively, $\hat{x}^{(k+1)}$ is our target value for iteration $k+1$, $x^{(k+1)}$ is our approximation of $\hat{x}^{(k+1)}$, and $\delta^{(k+1)}$ is the error in our approximation.

The implicit constraints at each iteration iteration can be expressed recursively as the sum of the implicit constraints at the previous iteration, the target correction factor, and the truncation error. Explicitly,

$$b^{(k+1)} = b^{(k)} + \Delta^{(k)} - A\delta^{(k+1)}$$

By definition of $I^{(k)}$, if $i \notin I^{(k)}$ then $|(b^{(k)} - b)_i| < \varepsilon$ so for each $i \notin I^{(k)}$ there exists $\varepsilon_i > 0$ such that $|(b^{(k)} - b)_i| + \varepsilon_i < \varepsilon$. Let $\varepsilon^{(k)} = \min_{i \notin I^{(k)}} \varepsilon_i$. Then there exists $R_C$ such that

$$\frac{1+\lambda}{(1-\lambda)\sqrt{m}} \lambda^{R_C+1} \|\Delta\| < \varepsilon^{(k)}. \tag{19}$$

Similarly, for each $\Delta^{(k)}$ there exists $R_x$ such that

$$\frac{\left\|\Sigma A^T\right\|}{1-\lambda} \lambda^{R_x+1} \left\|\Delta^{(k)}\right\| < \frac{\varepsilon_x}{|I^{(0)}|} \tag{20}$$

Taking $R = \max(R_C, R_x)$ allows both equations to be satisfied. So, for each outer iteration, there is some $R$ that allows the inner loop to terminate. In particular, for each iteration $R \geq R_x$. We can express our total error in $x$ as the sum of the errors made in each other iteration, so by the triangle inequality,

$$\begin{aligned}
\left\| \sum_{k=1}^{|I^{(0)}|} \delta^{(k)} \right\| &\leq \sum_{k=1}^{|I^{(0)}|} \left\| \delta^{(k)} \right\| \\
&\leq \sum_{k=1}^{|I^{(0)}|} \frac{\left\|\Sigma A^T\right\|}{1-\lambda} \lambda^{R_x+1} \left\|\Delta^{(k)}\right\| \\
&\leq \sum_{k=1}^{|I^{(0)}|} \frac{\varepsilon_x}{|I^{(0)}|} \\
&< \varepsilon_x
\end{aligned}$$

Then, within $|I^{(0)}|$ outer iterations, $I^{(|I^{(0)}|)} = \emptyset$ so the algorithm terminates. The termination condition, $I^{(|I^{(0)}|)} = \emptyset$ ensures that the constraint bounds are met. □

### *D. The Linial and Saks Algorithm for Weak Diameter Graph Decomposition [19]*

In [19], Linial and Saks presented a randomized distributed algorithm for weak-diameter graph decomposition. We use their algorithm to partition our original problem into sub-problems that are solved locally. For a graph $G = (V, \mathscr{E})$, with $n = |V|$ nodes, and parameters $p \in [0,1]$, and $R \geq 1$, their algorithm generates a subset of the nodes $S \subseteq V$, and leaders $l(u)$ for all $u \in S$ such that

1) For all $u \in S$, $\quad d_G(u, l(u)) \leq R$
2) If $l(u) \neq l(v)$ then $(u,v) \notin E$
3) For all $u \in V^{(d)}$, $\quad \mathbb{P}[u \in S] \geq p(1-p^R)^{n-1}$

In other words, the algorithm of Linial and Saks clusters nodes and elects cluster-heads such that no node is of distance greater than $R$ from its cluster-head, and nodes belonging to different clusters are of distance at least 2 away from each other. For the purposes of generating the constraint cut set and decomposing the original problem into sub-problems, each $G_i$ is one of these clusters, $C = V^{(d)} \setminus \bigcup_i G_i$, and the sub-problem associated with each cluster is solved by the cluster-head who then relays the solution to the appropriate agents in his cluster. The algorithm is summarized as follows:

**Algorithm 2:** Linial and Saks

**input:** $G = (V, \mathscr{E})$
**foreach** *Node* $y \in V$ **do**
  Select $r_y$ according to
    $\mathbb{P}[r_x = j] = p^j(1-p), \quad j = 0, ..., R-1$
    $\mathbb{P}[r_x = B] = p^B$
  ;
  Broadcast $(ID_y, r_y)$ to all vertices within distance $r_y$;
  Select vertex $C(y)$ of highest *ID* from the broadcasts it received (including itself);
  Join the cluster if $d(y, C(y)) < r_{C(y)}$;
**end**

### *E. Probabilistic Bounds on the Constraint Cut-Set*

In phase I of the distributed algorithm, we proposed generating the constraint cut-set using the algorithm of Linial and Saks for weak-diameter graph decomposition [19]. The algorithm takes $p$ and $R$ as parameters and guarantees that for all $u \in V^{(d)}$, the probability that $u$ is in $\tilde{C}$ is greater than or equal to $p(1-p^R)^{n-1}$, which implies that

$$\mathbb{P}[u \in C] \leq 1 - p(1-p^R)^{m-1}.$$

Suppose we estimate $|I^{(0)}| \leq I$. We use the multiplicative Chernoff bound to upper-bound the probability that $|I^{(0)}| > I$, which will upper-bound the probability that

$$\frac{\left\|\Sigma A^T\right\|}{1-\lambda} \lambda^{r+1} \left\|\Delta^{(k)}\right\| > \frac{\varepsilon_x}{|I^{(0)}|}$$

for any given iteration.

Letting $q = 1 - p(1-p^R)^{m-1}$, the Chernoff bounds says that

$$\mathbb{P}[|I^{(0)}| > (1+\delta)\mu] < \left( \frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^\mu$$

where $\mu = m \cdot q = m(1 - p(1-p^R)^{m-1})$. This expression can be used to tune the parameter, $p$, used for partitioning the problem, as well as assess the trade-off between the estimate for $|I^{(0)}|$ versus the probability the algorithm succeeds. The estimate for $|I^{(0)}|$, whether it is high or low, will ultimately determine the communication volume in the second phase.

### F. Convergence Comparison to the Projected Subgradient Method

To demonstrate that our algorithm exploits locality in a way that black-box algorithms fail to do, we now evaluate the convergence of the distributed projected subgradient method on our problem.

The distributed subgradient method assumes an optimization problem of the form

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} f_i(x) \\ x \in \mathbb{R}^N & \\ \text{subject to} & x \in \chi_i \end{array} \tag{21}$$

where each $f_i(x)$ and $\chi_i$ is only known by agent $i$, and messages are passed over a fixed communication topology. As this does not exactly match our problem statement, we make several modifications which are summarized and justified as follows.

In problem 1 we assumed that all agents know $f(x)$, and initially, only agent $i$ knows $A_{*,i}$. After the initial communication round, if $v_i^{(p)} \in S_j$ then agent $i$ knows $A_{j,*}$. In the algorithm of Section IV, each dual variable is assigned to a primal variable, who is then "responsible" for that constraint. We also note that the distance metric proposed in Section III evolves according to the geodesic distance in $G_{\text{eff}}$. Consequently, in order to most equitably compare the convergence of the locality aware algorithm to the distributed projected subgradient method, we simulate the projected subgradient method on the dual variables with a communication graph defined by $G_{\text{eff}}$.

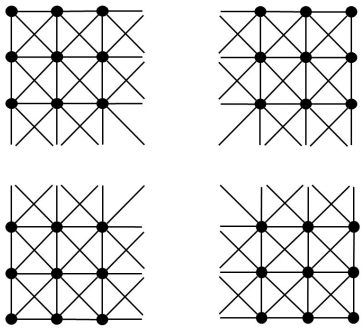

Fig. 5: Topology of $G_{\text{eff}}$

We rewrite problem 1 in the form

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{(M-1)\times(N-1)} \frac{f(x)}{(M-1)\times(N-1)} \\ x \in \mathbb{R}^N & \\ \text{subject to} & A_{i,*}x = b_i \end{array} \tag{22}$$

to match the form of Equation (21). A fraction of $f(x)$ is included in each agents' private objective function to distribute processing of $f(x)$, and speeds up convergence. We also initialize each node with $x^{(0)}$ i.e., the subgradient method is warm-started with the solution after the first phase of the algorithm to compare convergence against that of the second phase of our algorithm. The lazy Metropolis weighting given by,

$$x_i^{k+1} = x_i^k = \sum_{j=\mathcal{N}_i^k} \frac{1}{2\max\{d_i^k, d_j^k\}}(x_j^k - x_i^k)$$

is used for the consensus step for of its attractive convergence properties. In matrix form, we write $x^{k+1} = Lx^k$. We choose step-sizes $\alpha^{(k)} = \frac{\alpha_0}{\sqrt{k}}$ based on the standard divergent series rule, and vary $\alpha_0 \in \{0.03125, 0.0625, 0.125, 0.250, 0.5, 1\}$

In the projected subgradient algorithm, every agents maintains and updates a copy of the global variable during each iteration. Let $x_{(i)}^k$ denote the $i$th agent's copy of the global optimization variable at iteration k. The projected subgradient updates are given by

$$x_{(i)}^{k+1} = \Pi_{\chi_i}\left(\sum_j L_{ij}x_{(j)}^k - \frac{\alpha_0}{\sqrt{k}}g_{(i)}^k\right)$$

where $\chi_i := \{x | A_{i,*}x = b_i\}$, and $\Pi_S(x)$ is the orthogonal projection of the point $x$ on the set $S$.

We fixed the dimension of the global problem to be $10 \times 10$ , maximum sub-problem size to be $4 \times 4$, $\alpha = 1$, and $\beta = 3$. These parameters partition the global problem into 4 sub-problems, and result in a locality parameter of

Figure 6 plots the convergence of the distributed subgradient method from a warm-start obtained by solving the sub-problems. Figure 7 plots convergence of phase two of the locality-aware algorithm from the same starting value. The locality-aware algorithm exhibits orders of magnitude better performance than the subgradient method in terms of convergence rate, which translates directly into massive savings in communication cost.

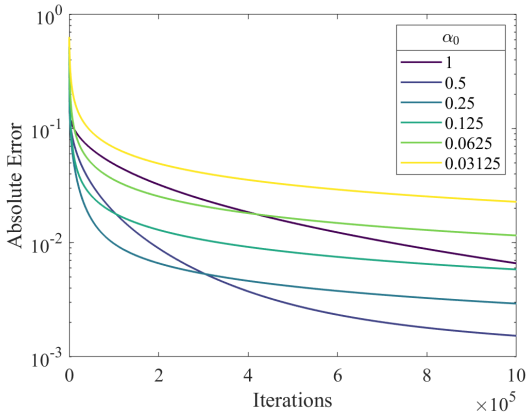


Fig. 6: Convergence rate of the projected subgradient algorithm for stepsizes $\alpha_0 \in \{0.03125, 0.0625, 0.125, 0.250, 0.5, 1\}$

### G. Scaling with Problem Size

We now show that for a specific instance of our example problem, the locality parameter appears to asymptotically approach a limit that is less that one. This is significant because for problems of this form, it means that locality remains bounded independently of problem size. In contrast, we show that the convergence time of a distributed subgradient method using lazy Metropolis weights scales with the square of the number of nodes in the network. This implies that the total messages passed needed for the projected distributed subgradient method scales cubically with the number of nodes in the network.

In particular, we fix $\alpha = 1$ and $\beta = 3$, let $\rho_N$ be the locality parameter for the problem on a $N \times N$ grid. Figure 8 plots the $\rho_N$ against $N^2$, the number of nodes in the network. Noteably,

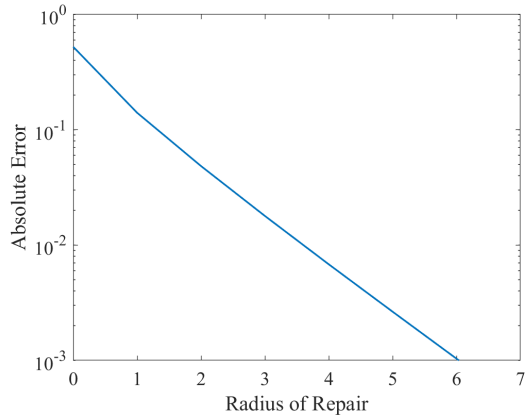


Fig. 7: Convergence of phase two of the locality-aware algorithm

we observe that $\rho_N$ appears to asymptotically approach 0.43

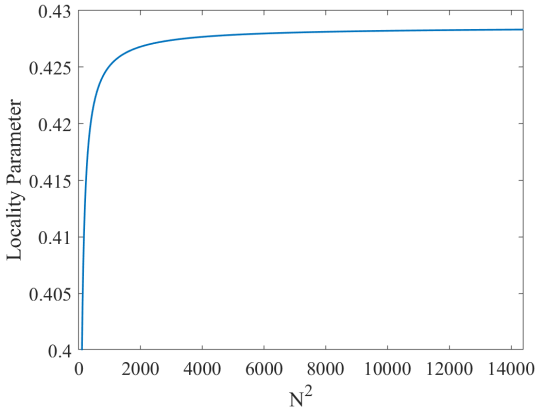


Fig. 8: Scaling of locality parameter $\rho_N$

Let $L_N$ be the matrix encoding the lazy Metropolis update on the $N \times N$ grid. Convergence of the distributed subgradient method is dictated $\frac{1}{1-\gamma(L_N)}$ where $\gamma(L_N)$ is the second largest singular value of $L_N$. Specifically, we define the $\varepsilon$-convergence time as the minimum $T$ such that

$$f\left(\frac{\sum_{l=0}^{T-1} y^l}{T}\right) - f(x^*) \leq \varepsilon$$

The $\varepsilon$ convergence time can be upperbounded by

$$\mathscr{O}\left(\frac{\max((y^0 - x^*)^4, C^4 \frac{1}{(1-\gamma(L_N))^2}}{\varepsilon^2}\right)$$

where $C$ is a universal bounding constant for $\left\|\nabla \frac{f(x)}{m} + \frac{1}{2}\|A_{i,*}x - b_i\|^2\right\|$ for all $i$ [1]. Figure 9 shows the scaling of $\frac{1}{1-\gamma(L_N)}$ with $N^2$. Empirically, we found that $\frac{1}{1-\gamma(L_N)}$ scales approximately linearly with $N^2$, and the convergence rate approximately scales with $N^4$. Note that at each time step, *every* node passes a message to each one of its neighbors, resulting in a total of $\mathscr{O}(N^6)$ messages passed.

### H. Discussion

As noted in Section IV, it is not strictly necessary to use a leader based algorithm for the first phase of the locality-aware algorithm. After partitioning, we could instead use

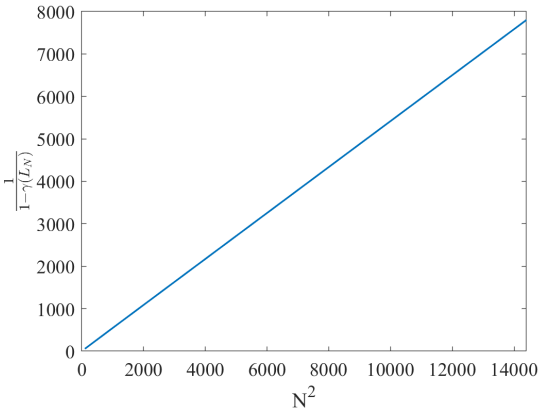


Fig. 9: Scaling of $\frac{1}{1-\gamma(L_N)}$

a subgradient method. If we partition the original $N \times N$ problem into $M \times M$ evenly sized sub-problems, each sub-problem has size $\frac{N}{M} \times \frac{N}{M}$. Based on the scaling results of this section, the total number of messages passed in the first phase would scale with $M^2\left(\frac{N^2}{M^2}\right)^3 = \frac{N^6}{M^4}$. We can expect communication volume in the second phase to scale as a function of the number of constraints cut in the first phase; approximately $2NM$ constraints are cut, and only nodes associated with the cut dual variables communicate within their coordination radius. Consequently, the communication volumne in the second phase scales approximately with $(NM)^2$. A quick back of the envelope calculation shows that if we choose $M \propto N^{\frac{2}{3}}$, total communication volume for both phases scales with $N^{\frac{10}{3}}$, which is a dramatic improvement over a naive implementation of the subgradient method. We note that this improvement factor is problem dependent and a direct consequence of the structure of the constraints and objective function