**Accelerating Distributed Task Assignment With Mixing Matrix Co-Estimation**

Name: Oscar Eriksson
SUNet ID: oscer
oscer@stanford.edu

June 2023

# Introduction

The field of distributed optimization is very mature, and a great deal of time is spent developing distributed optimization algorithms that can make improvements of convergence rates over other methods, given that one finds structure in the problem. There is however not the same abundance of experimental trials in robotics, with [Hak+22], [Sho+20] and [JLM03] being some examples. This paper aims to apply distributed methods on a problem in robotics directly, and letting it play out in real time in a simulated environment. We present a problem setting in which bots are to pick up and deliver packages subject to charge constraints. They operate using very simple dynamics and trajectory planning with no collision avoidance. A shared objective function to be minimized is presented where each bots task assignment make up the optimization variables. A relaxation of the shared objective is proposed, for which various decentralized methods are derived including co-estimation of the fastest mixing matrix on a graph.

### Related Work

One example of distributed task assignment is found in [Sho+23] in which a version consensus ADMM which does not require a central synchronized host is used to solve a task assignment problem with a linear objective such as (2). It reviews the use of the planning in a simulated system as well, and runs the algorithm every few iterations. The subject of the fastest mixing matrix on a graph is discussed in [XB04], and faster averaging methods are presented in [JJ08] yet not implemented. There is discussion regarding the convergence rate of decentralized algorithms subject to slowly varying graphs, as well as time varying functions in [Rog+19] where they state the assumptions under which these algorithms converge, though it is not possible to transpose the analysis to this setting.

### Contribution

In this paper fixed point iterations are derived that may run in real time, allowing the relaxed objective and the graph structure to change during the course of the algorithm, while maintaining stability and performance. In this sense the planning is applied like the controls of a dynamical system, rather than a regularly scheduled subroutine. To address the fact that communication is the bottleneck of this system we propose the co-estimation of the fastest mixing matrix along with the objective value to make better use of the transmitted data and show it's improvements on the baseline approach.

The code repository containing the numerical experiments is available in
`https://github.com/osceri/Accelerating-Distributed-Task-Assignment-With-Mixing-Matrix-Co-Estimation`.

# Problem Setup

The particular problem instance that we are considering is one with $B$ bots at positions $x \in \mathbb{R}^{B\times 2}$ with charge $U \in [0,1]^B$, that are to deliver $M$ packages from pick-up points $P \in \mathbb{R}^{M\times 2}$ to drop-off points $Q \in \mathbb{R}^{M\times 2}$. The time that a mission $m$ has been available is denoted $T_m$, and resets (to zero) when a mission is completed. Bots move in a

straight line towards their objectives and a displacement $\Delta$ results in a charge draw of $\Delta\eta$. If a bot is running low on charge it should go to a charging location $C \in \mathbb{R}^{C\times 2}$ and charge at some constant rate. Each bot $b$ moves according to a local task assignment matrix $\tau_b \in \{0,1\}^{B\times(M+C)}$ with $\tau_{bt}$ being 1 if the bot has been tasked to got to the mission or charger $t$. $\tau$ is furthermore constrained such that at most one task is assigned to at most one bot.

**Knowledge of mission status.** If a package is picked up then the mission is no longer available for other bots. If a charger is occupied it will not be available until the bot is fully charged. The status of chargers and missions are described by $K \in \{0,1\}^{M+C}$. $(P, Q, C, K, T)$ are instantaneously transmitted to each active agent regardless of their position, and any time a delivery is completed another pick-up emerges at some random location so that there always are $M$ missions available.

**Communication between bots.** Communication between bots is constrained to channels represented by a finite connected graph with no self-loops $G$ with vertices $i \in V$ corresponding to each bot, and edges $\{i, j\} \in E$. It may be represented by an adjacency matrix $A \in \{0,1\}^{B\times B}$ such that $A_{ij} = 1$ if $\{i, j\} \in E$ and $A_{ij} = 0$ if $\{i, j\} \notin E$. Bots are connected with the $I$ closest bots at any point in time. Communication occurs every $\Delta T$ seconds.

**Shared objective function.** The shared objective among the bots is to deliver as many packages as possible without neglecting difficult missions. The shared objective is thus

$$\underset{\tau_1,\dots,\tau_b\in\{0,1\}^{B\times(M+C)}}{\text{minimize}} \int_0^\infty \sum_{m=1}^{M} T_m dt \tag{1}$$

# Centralized Relaxation

We allow each bot to produce an objective vector $L^b \in \mathbb{R}^{M+C}$ which encodes the bots wish to go to a certain mission or charger at the instance that the vector is produced. The composition of all $L^b$ is denoted $L \in \mathbb{R}^{B\times(M+C)}$. We define this as $L = \left[\begin{array}{cc} \mathbf{M} & \mathbf{C} \end{array}\right] \in \mathbb{R}^{B\times M+B\times C}$ as

$$\mathbf{M}_{bm} = \begin{cases} -1/(1+D_{bm}) - \mu T_m & \text{if } U_b > \eta D_{bm} + 0.2 \\ 1 & \text{else} \end{cases}, \quad \mathbf{C}_{bc} = \begin{cases} -1/(1+D_{bc}) & \text{if } 0.8 \geq U_b \geq \eta D_{bc},\ \mathbf{M}_b = \mathbf{1} \\ 1 & \text{else} \end{cases}$$

where $D_{bm}$ is the shortest possible distance bot $b$ has to traverse in order to pick up, drop off and reach a charging station, and $D_{bc}$ is the distance from bot $b$ to charger $c$. $\mu T_m$ is the time that a mission has been available scaled by a parameter $\mu$, which penalizes bots ignoring difficult missions. We also define $\mathbf{L} \in \mathbb{R}^{B\times B\times(M+C)}$ such that $\mathbf{L}_{bbm} = L^b_m$ (all other elements zero), where $\mathbf{L}_b$ corresponds to the exact knowledge of bot $b$.

**Interpretation.** Elements in $\mathbf{M}$ are negative if the bot has enough charge to complete that mission and get to a charger afterwards, and the reward is greater for completing mission that are nearby yet have been active for long. Elements in $\mathbf{C}$ are negative when no missions are available and the charge low enough to warrant charging yet enough to make it to the charger.

**Centralized problem.** If we ignore the constraint on communication between bots, and allow for complete information we may elect to solve the following centralized relaxation problem

$$\underset{\tilde{\tau}_b\in\mathbb{R}^{B\times(M+C)}}{\text{minimize}} \langle \tilde{\tau}_b, L\rangle + \delta_T(\tilde{\tau}_b), \quad T = \{\ \tau \in \mathbb{R}^{B\times(M+C)} \mid 0 \preceq \tau,\ \tau\mathbf{1} \preceq 1,\ \tau^\intercal\mathbf{1} \preceq 1\ \} \tag{2}$$

using $\langle A, B\rangle = \sum_{i,j} A_{ij}B_{ij}$, and defining $T \subseteq \mathbb{R}^{B\times(C+M)}$, the subspace of inexact task assignment matrices where each bot has at most one task and each task has at most one bot. We use the indicator function $\delta_A(x)$ which is equal to plus infinity if $x \notin A$ and zero otherwise. It is easy to see that $\mathbf{conv}\ \{0,1\}^{B\times(M+C)} = T$, and given that entries of $L$ are not equal we will find the optimal solution to (2) to be a vertex of $T$ (an exact binary assignment).

# Decentralized Relaxation

We now introduce stacking notation [RY22], such that

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_B \end{bmatrix}, \quad \text{Prox}_{\partial f}(\boldsymbol{\tau}) = \begin{bmatrix} \text{Prox}_{\partial f_1}(\tau_1) \\ \text{Prox}_{\partial f_2}(\tau_2) \\ \vdots \\ \text{Prox}_{\partial f_B}(\tau_B) \end{bmatrix}$$

which allows us to express the decentralized task assignment problem as

$$\underset{\tilde{\boldsymbol{\tau}} \in \mathbb{R}^{B \times B \times (M+C)}}{\text{minimize}} \sum_{b=1}^{B} \langle \tilde{\boldsymbol{\tau}}_b, \mathbf{L}_b \rangle + \delta_T(\tilde{\boldsymbol{\tau}}) + \delta_O(\tilde{\boldsymbol{\tau}}), \quad O = \{ \ \tau \in \mathbb{R}^{B \times (M+C)} \mid \tau = \tau_1 = \tau_2 = ... = \tau_B \ \} \tag{3}$$

where $O$ is the consensus set defined such that all matrices contained are equal in values. In this case each bot only has access to it's own objective vector.

**Consensus through mixing matrices.** Consensus constraints may be implemented using mixing matrices $W \in \mathbb{R}^{B \times B}$. Any mixing matrix [XB04] should satisfy

$$\mathcal{N}(I - W) = \text{span}(\mathbf{1}), \ \sigma_{\max}(W) = 1, \ (\mathbf{1}\mathbf{1}^\intercal - A) \odot W = 0$$

where the first criterion makes sure that averaging algorithms reach consensus ($\mathbf{x}_i^\star = \mathbf{x}_j^\star \ \forall i, j$ and $\mathbf{x}^{k+1} = W\mathbf{x}^k$ if and only if $(I - W)\mathbf{x}^\star = 0$). The second criterion makes sure that averaging operations are non expansive and the third criterion makes sure that the mixing matrix is a decentralized mixing matrix corresponding to the adjacency matrix $A$ ($\odot$ denotes element wise multiplication). We define the space of all these decentralized mixing matrices $\mathcal{W}(A) \subseteq \mathbb{R}^{B \times B}$. We claim that the rate of convergence of data averaging is roughly $\approx \sigma_{\max}(W - U)$, the greatest singular value (being used as an approximation of the spectral radius) of the difference of the mixing matrix and the uniform matrix $U = \mathbf{1}\mathbf{1}^\intercal / B$. If not specified a mixing matrix which only uses known local information may be assumed to be

$$W_{ij} = \begin{cases} A_{ij} / \sum_j A_{ij} \text{ if } \sum_j A_{ij} > 0 \\ 0 \text{ else} \end{cases} \tag{4}$$

**Decentralized consensus problem.** We may approximate $\delta_O(\boldsymbol{\tau})$ with $\frac{1}{2\alpha}\|\boldsymbol{\tau}\|_{I-W}^2$ ($\alpha > 0$), since these expressions both evaluate to zero under consensus. We thus find the problem

$$\underset{\tilde{\boldsymbol{\tau}} \in \mathbb{R}^{B \times B \times (M+C)}}{\text{minimize}} \sum_{b=1}^{B} \langle \tilde{\boldsymbol{\tau}}_b, \mathbf{L}_b \rangle + \delta_T(\tilde{\boldsymbol{\tau}}_b) + \frac{1}{2\alpha}\|\tilde{\boldsymbol{\tau}}_b\|_{I-W}^2 \tag{5}$$

A fixed point iteration can be found by setting the subdifferential of the above objective function to zero

$$0 \in \alpha\mathbf{L} + \alpha\mathbb{N}_T(\tilde{\boldsymbol{\tau}}) + (I - W)\tilde{\boldsymbol{\tau}} \iff \tilde{\boldsymbol{\tau}} \in (I + \alpha\mathbb{N}_T)^{-1}(W\boldsymbol{\tau} - \alpha\mathbf{L}) \iff \tilde{\boldsymbol{\tau}}^{k+1} = \Pi_T(W\tilde{\boldsymbol{\tau}}^k - \alpha\mathbf{L}) \tag{6}$$

which gives us our first fixed point iteration. The optimal value of this fixed point iteration is not a vertex of $T$, and as such we find our task assignments in $\boldsymbol{\tau}$ below

$$\boldsymbol{\tau}_{rbp} = \begin{cases} 1 & \text{if } p = \text{argmax}\, \tilde{\boldsymbol{\tau}}_{rb} \\ 0 & \text{else} \end{cases}.$$

**Faster mixing matrices.** We recognize that we may theoretically be able to increase the speed at which the averaging converges by finding a mixing matrix $W$ which minimizes the spectral radius $\rho(W - U)$ [OT09]. We pose the following problem

$$\underset{\tilde{\mathbf{W}} \in \mathbb{R}^{B \times B \times B}}{\text{minimize}} \sum_{b=1}^{B} \rho(\tilde{\mathbf{W}}_b - U) + \delta_{\mathcal{W}(A^b)}(\tilde{\mathbf{W}}_b) + \frac{1}{2\alpha}\|\tilde{\mathbf{W}}_b\|_{I - \tilde{\mathbf{W}}_b}^2 \tag{7}$$

This is clearly not convex, and as such we pose the following sequential convex optimization problem

$$\tilde{\mathbf{W}}_b^{k+1} = \underset{\mathbf{W}\in\mathbb{R}^{B\times B\times B}}{\operatorname{argmin}} \sum_{b=1}^{B} \rho(\mathbf{W}_b - U) + \delta_{\mathcal{W}(A^b)}(\mathbf{W}_b) + \frac{1}{2\alpha}\|\mathbf{W}_b\|^2_{I-\tilde{\mathbf{W}}_b^k} \tag{8}$$

The spectral radius $\rho(\cdot)$ will have approximately the same zeros as $\lambda^2_{\max}(\cdot)$ [XB04]. We find this functions derivative in

$$\begin{aligned}\lambda^2_{\max}(X) &= \max_{v:v^Tv=1} (v^TXv)^2 \\ \nabla\lambda^2_{\max}(X) &= \lambda_{\max}(X)qq^T,\ q = \underset{v:v^Tv=1}{\operatorname{argmax}}\ (v^TXv)^2\end{aligned}$$

which may have poor numerical performance around $\lambda^2_{\max}(X) = 0$. We use Runge-Kutta methods [But07] to define the function $\nabla\lambda^2_{\mathrm{RK4}}(\alpha, X)$ below

$$\begin{aligned}k_1 = \lambda^2_{\max}(X),\ k_2 = \lambda^2_{\max}(X + \frac{\alpha}{2}k_1),\ k_3 = \lambda^2_{\max}(X + \frac{\alpha}{2}k_2),\ k_4 = \lambda^2_{\max}(X + \alpha k_3),\\ \nabla\lambda^2_{\mathrm{RK4}}(\alpha, X) = \frac{\alpha}{6}(k_1 + 2k_2 + 2k_3 + k_4)\end{aligned} \tag{9}$$

which we will use instead of $\nabla\rho(X)$. We find the optimality condition of (8) in

$$0 \in \alpha\nabla\rho(\mathbf{W}_b - U) + \alpha\mathbb{N}_{\mathcal{W}(A^b)}(\mathbf{W}_b) + (I - \tilde{\mathbf{W}}_b^k)\mathbf{W}_b$$

from which we may find a fixed point iteration

$$\begin{aligned}0 \in \alpha\partial\rho(\mathbf{W}_b - U) + \alpha\mathbb{N}_{\mathcal{W}(A^b)}(\mathbf{W}_b) + (I - \tilde{\mathbf{W}}_b^k)\mathbf{W}_b& \\ \iff \mathbf{W}_b \in \mathbb{J}_{\mathcal{W}(A^b)}(\tilde{\mathbf{W}}_b^k\mathbf{W}_b - \nabla\lambda^2_{\mathrm{RK4}}(\alpha, \mathbf{W}_b - U))& \\ \iff \tilde{\mathbf{W}}_b^{k+1} = \Pi_{\mathcal{W}(A^b)}(\tilde{\mathbf{W}}_b^k\tilde{\mathbf{W}}_b^k - \nabla\lambda^2_{\mathrm{RK4}}(\alpha, \tilde{\mathbf{W}}_b^k - U))&\end{aligned} \tag{10}$$

Composing 10 and 6 we may get the following sequential convex optimization algorithm

$$\begin{cases}\tilde{\boldsymbol{\tau}}_b^{k+1} = \Pi_T(\tilde{\mathbf{W}}_b^k\tilde{\boldsymbol{\tau}}_b^k - \alpha\mathbf{L}_b) \\ \tilde{\mathbf{W}}_b^{k+1} = \Pi_{\mathcal{W}(A^b)}(\tilde{\mathbf{W}}_b^k\tilde{\mathbf{W}}_b^k - \nabla\lambda^2_{\mathrm{RK4}}(\alpha, \tilde{\mathbf{W}}_b^k - U))\end{cases} \tag{11}$$

**Sharing objective vectors.** Mixing matrices are especially appropriate for data averaging, and as such we propose sharing the gradient $\mathbf{L}$. We pose the following optimization problem

$$\underset{\tilde{\mathbf{L}}\in\mathbb{R}^{B\times B\times(M+C)}}{\text{minimize}}\ \delta_{\mathcal{L}}(\tilde{\mathbf{L}}) + \frac{1}{2\alpha}\|\tilde{\mathbf{L}}\|_{I-W},\ \mathcal{L} = \{\mathbf{L}\in\mathbb{R}^{B\times B\times(C+M)} \mid \mathbf{L}_{bb} = L^b\} \tag{12}$$

$\mathcal{L}$ makes it so that bot $b$ enforces it's knowledge of $L^b$ (the objective vector it just produced). We state the optimality condition and find the following fixed point iteration

$$0 \in \alpha\mathbb{N}_{\mathcal{L}}(\tilde{\mathbf{L}}) + (I - W)\tilde{\mathbf{L}} \iff W\tilde{\mathbf{L}} \in (I + \alpha\mathbb{N}_{\mathcal{L}})\tilde{\mathbf{L}} \iff \tilde{\mathbf{L}} \in \mathbb{J}_{\mathcal{L}}(W\tilde{\mathbf{L}})$$

where $\mathbb{N}$ denotes the normal cone operator and $\mathbb{J}$ denotes the resolvent [RY22]. Using this $\tilde{\mathbf{L}}$ we may calculate the optimal $\tilde{\boldsymbol{\tau}}$ every time step. We write the following fixed point iteration

$$\begin{cases}\tilde{\boldsymbol{\tau}}_b^{k+1} = \underset{\tau\in\mathbb{R}^{B\times B\times(C+M)}}{\operatorname{argmin}} \langle\tau, \tilde{\mathbf{L}}_b^k\rangle + \delta_T(\tau) \\ \tilde{\mathbf{L}}^{k+1} = \Pi_{\mathcal{L}}(W\tilde{\mathbf{L}}^k)\end{cases} \tag{13}$$

**Sharing adjacency vectors.** We estimate a $\mathbf{W}$ and a $\mathbf{A}$ analogously to $\boldsymbol{\tau}$ and $\mathbf{L}$ above in the following optimization problem

$$\underset{\tilde{\mathbf{A}},\tilde{\mathbf{W}}\in\mathbb{R}^{B\times B\times B}}{\operatorname{argmin}} \sum_{b=1}^{B} \rho(\tilde{\mathbf{W}}_b - U) + \frac{1}{2\alpha}\|\tilde{\mathbf{A}}_b\|_{I-\tilde{\mathbf{W}}_b} + \delta_{\mathcal{A}}(\tilde{\mathbf{A}}) + \delta_{\mathcal{W}(\mathbf{A}_b)}(\tilde{\mathbf{W}}_b), \mathcal{A} = \{\mathbf{A}\in\mathbb{R}^{B\times B\times B} \mid \mathbf{A}_{bb} = A^b\} \tag{14}$$

which is not convex. $\mathcal{A}$ makes it so that bot $b$ enforces it's knowledge of $A^b$ (what other bots it has communication with). We instead elect to solve the sequential convex optimization problem

$$(\tilde{\mathbf{A}}^{k+1}, \tilde{\mathbf{W}}^{k+1}) = \underset{\mathbf{A},\mathbf{W}\in\mathbb{R}^{B\times B\times B}}{\text{minimize}} \sum_{b=1}^{B} \rho(\tilde{\mathbf{W}}_b - U) + \frac{1}{2\alpha}\|\mathbf{A}_b\|_{I-\tilde{\mathbf{W}}_b^k} + \delta_{\mathcal{A}}(\mathbf{A}) + \delta_{\mathcal{W}(\tilde{\mathbf{A}}_b^k)}(\mathbf{W}_b) \tag{15}$$

with optimality conditions

$$0 \in \partial\rho(\tilde{\mathbf{W}}_b - U) + \mathbb{N}_{\mathcal{W}(\tilde{\mathbf{A}}_b^k)}(\mathbf{W}_b),\ 0 \in (I - \tilde{\mathbf{W}}_b^k)\mathbf{A} + \alpha\mathbb{N}_{\mathcal{A}}(\mathbf{A}_b).$$

Since these optimality conditions have separated $\mathbf{W}$ from $\mathbf{A}$ we optimize with respect to each condition independently. Following the derivation of (13) we may arrive at

$$\begin{cases} \tilde{\boldsymbol{\tau}}_b^{k+1} = \underset{\tau\in\mathbb{R}^{B\times(C+M)}}{\text{argmin}} \langle \tau, \tilde{\mathbf{L}}_b^k\rangle + \delta_T(\tau) \\ \tilde{\mathbf{L}}_b^{k+1} = \Pi_{\mathcal{L}}(\tilde{\mathbf{W}}_b^k \tilde{\mathbf{L}}_b^k) \\ \tilde{\mathbf{W}}_b^{k+1} = \underset{W\in\mathbb{R}^{B\times B\times B}}{\text{argmin}} \rho(W - U) + \delta_{\mathcal{W}(\tilde{\mathbf{A}}_b^k)}(W) \\ \tilde{\mathbf{A}}_b^{k+1} = \Pi_{\mathcal{A}}(\tilde{\mathbf{W}}_b^k \tilde{\mathbf{A}}_b^k) \end{cases}. \tag{16}$$

# Results

The objective heuristic performs well as one can tell that bots seldom run out of charge in figure 1.

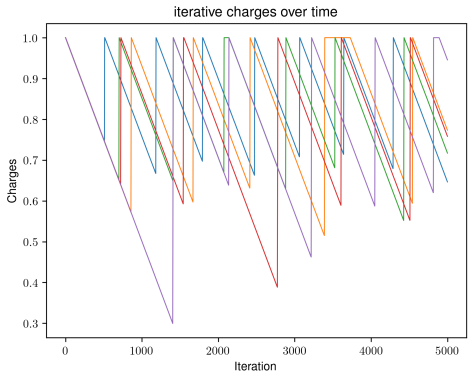



Figure 1: Charge of the bots every iteration for $B = 5$, $M = 5$, $C = 3$, $I = 3$, $\Delta T = 0.05$.

**Performance of various fixed point iterators.** We try out various planning algorithms. The first scheme is a naive approach where the task corresponding to the lowest cost is selected every iteration. The second scheme where every bot is assigned the optimal ("exact") solution of the shared heuristic equation (2) every iteration. We also implement algorithms for equations (6), (11), (13) and (16) which we denote "iterate", "iterate + mix", "share" and "share + mix" respectively. Numerical results are shown in figure 2, and presented in table 1 where $\int\sum T$ denotes the objective value (1) and # is the amount of delivered packages.

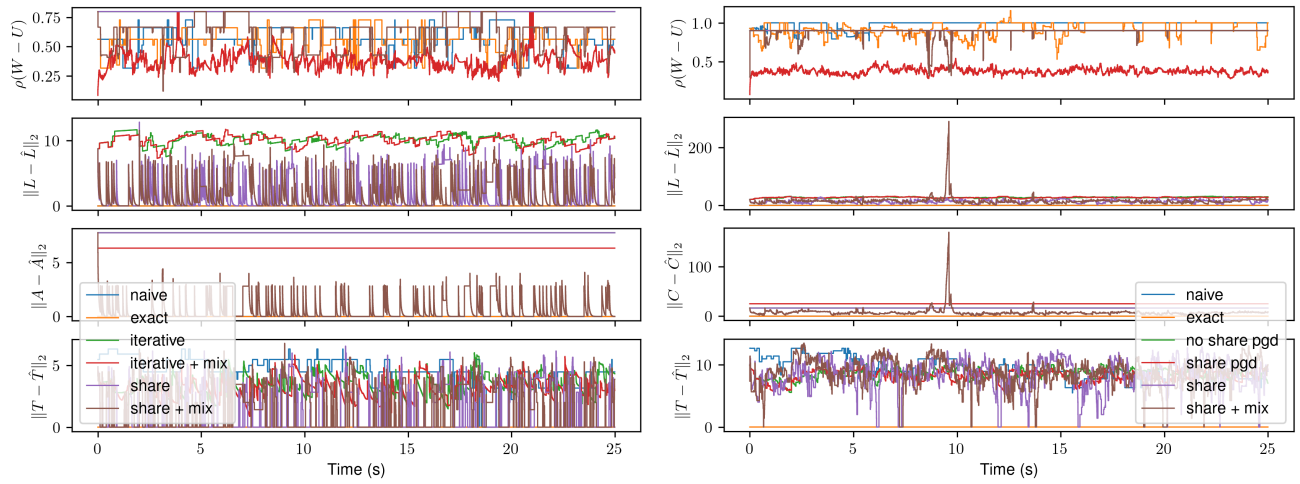
| | $\int\sum T$ | # | $\|L - \hat{L}\|_2$ | $\|T - \hat{T}\|_2$ | $\rho(W - U)$ |
|---|---|---|---|---|---|
| naive | 278183 | 30 | 0.00 | 4.59 | 0.57 |
| exact | 229395 | 45 | 0.00 | 0.00 | 0.56 |
| iterate | 232954 | 37 | 10.11 | 3.43 | 0.80 |
| iterate + mix | 201695 | 44 | 10.06 | 3.18 | 0.38 |
| share | 251752 | 41 | 1.73 | 1.12 | 0.80 |
| share + mix | 270589 | 36 | 1.74 | 1.36 | 0.61 |

(a) Simulation for $B = 5$, $M = 5$, $C = 3$, $I = 3$.

| | $\int\sum T$ | # | $\|L - \hat{L}\|_2$ | $\|T - \hat{T}\|_2$ | $\rho(W - U)$ |
|---|---|---|---|---|---|
| naive | 1523293 | 41 | 0.00 | 9.32 | 0.98 |
| exact | 1082941 | 82 | 0.00 | 0.00 | 0.92 |
| iterate | 1131143 | 70 | 27.53 | 8.58 | 0.90 |
| iterate + mix | 865932 | 86 | 27.30 | 8.21 | 0.38 |
| share | 1206269 | 91 | 13.27 | 8.28 | 0.90 |
| share + mix | 924266 | 96 | 14.81 | 8.56 | 0.88 |

(b) Simulation for $B = 10$, $M = 10$, $C = 3$, $I = 3$.

Table 1: Simulation over 25 seconds for $\Delta T = 0.005$. $\hat{T}$, $\hat{L}$ are values corresponding to the "exact" solutions.




(a) Simulation for $B = 5$, $M = 5$, $C = 3$, $I = 3$.

(b) Simulation for $B = 10$, $M = 10$, $C = 3$, $I = 3$.

Figure 2: Simulation over 25 seconds for $\Delta T = 0.01$.

# Conclusions

The objective function (2) which is supposed to correspond to the "exact" solution performs very well since it attains good objective values (1), but was often outperformed by the iterative and sharing methods in our trials shown in figure 2. The performance of the iterative methods was dubious at first since the algorithm doesn't converge to an exact solution, instead arriving at a uniform task assignment matrix with a slight bias towards the preferred solution which one may round to. The iterative methods do however consistently outperform the shared methods which do arrive at exact solutions (often, given a high degree of connectivity $I$). Since the shared methods require more data to be transmitted, we find that the iterative methods always are better for this problem.

More surprising is that these methods would outperform the "exact" solution. The exact solution has demonstrated unwanted behaviour wherein two bots exchange assignments with each other back and forth until they randomly disentangle and find their own ways. This is resolved by unintended delays in the fixed point iterators which induce a hysteresis effect which prevents this problem.

Co-estimation of the fastest mixing matrix on the graph greatly improves the performance of the distributed methods. The "iterative + mix" method presents a really low spectral radius, but this is since every bot locally assumes the rest of the graph to be perfectly connected. It should locally not have the fastest mixing matrix, but it doesn't seem to be a problem for performance as it enjoys a great speed-up. The shared methods also enjoy a significant speed-up. The methods appear to be very robust.

**Future work.** The use of shifting registers in [JJ08] to improve the rate of convergence of linear iterators may be used to further increase the gain of the mixing matrix co-estimation. This would require one to reformulate the sharing to not use a proximal operator, which should be doable given how simple it is in form. One may of course also attempt accelerated methods.

No consensus-ADMM approach was devised for this setting, and the setting was too different to offer a fair comparison to [Sho+23]. Algorithms were attempted but no great effort was expended in making it perform comparably to the presented algorithms. It would be interesting to see how fast such a method is, given that it may not be accelerated with the use of a fast mixing matrix.

# References

[JLM03] Ali Jadbabaie, Jie Lin, and A Stephen Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". In: *IEEE Transactions on automatic control* 48.6 (2003), pp. 988–1001.

[XB04] Lin Xiao and Stephen Boyd. "Fast linear iterations for distributed averaging". In: *Systems & Control Letters* 53.1 (2004), pp. 65–78.

[But07] John Butcher. "Runge-kutta methods". In: *Scholarpedia* 2.9 (2007), p. 3147.

[JJ08] Björn Johansson and Mikael Johansson. "Faster linear iterations for distributed averaging". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 2861–2866.

[OT09] Alex Olshevsky and John N Tsitsiklis. "Convergence speed in distributed consensus and averaging". In: *SIAM journal on control and optimization* 48.1 (2009), pp. 33–55.

[Rog+19] Alexander Rogozin et al. "Optimal distributed convex optimization on slowly time-varying graphs". In: *IEEE Transactions on Control of Network Systems* 7.2 (2019), pp. 829–841.

[Sho+20] Ola Shorinwa et al. "Distributed multi-target tracking for autonomous vehicle fleets". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 3495–3501.

[Hak+22] Ravi N Haksar et al. "Consensus-based ADMM for task assignment in multi-robot teams". In: *Robotics Research: The 19th International Symposium ISRR*. Springer. 2022, pp. 35–51.

[RY22] Ernest K Ryu and Wotao Yin. *Large-Scale Convex Optimization: Algorithms & Analyses via Monotone Operators*. Cambridge University Press, 2022.

[Sho+23] Ola Shorinwa et al. "Distributed Multirobot Task Assignment via Consensus ADMM". In: *IEEE Transactions on Robotics* (2023).