On the Trade-Off Between Communication and Execution Overhead for Control of Multi-Agent Systems *

Anqi Li[†] and Magnus Egerstedt[†]

***Abstract*— For multi-agent systems, it is common to encode the task as an optimization problem with two distinctly different solution methodologies – one is to directly apply control inputs as optimization updates, the other is to solve the optimization problem through communications before applying actual control inputs. This reveals an important trade-off between communication and execution overhead for control of multi-agent systems. To formally study this trade-off, we restrict our consideration to a class of commonly studied multi-agent problems where the objective function is the sum of a set of edge potential functions. The gradient descent algorithm and Newton's method are viewed as the proxy for the pure execution and the pure communication strategy, respectively. We propose an algorithm based on truncated Newton's method that provides tunable levels of trade-off between communication and execution efforts. Theoretical results on the convergence rate of the purposed algorithm are studied for the consensus problem under different trade-off strategies. The performance of the proposed algorithm is validated through simulation.**

I. Introduction

In the multi-agent systems literature, it is common to encode the team-level, global task as an optimization problem with respect to a performance objective function [7], [10], [13]. Consequently, the control and coordination strategies are essentially distributed optimization algorithms with respect to the objective function [4], [6]. There are two fundamentally different ways, however, to perform optimization updates as multi-agent systems. One way is to directly apply the control inputs based on on-board sensor measurements. We name this type of updates as execution updates. The other is to update local decision variables through inter-agent communications, which we call communication updates.

As a result of that, there have been two methodologies for solving the underlying optimization problem. One methodology purely relies on execution updates, which has been investigated extensively in a lot of applications, such as rendezvous [12], [13], formation controls [10], coverage controls [7], etc. The other methodology is to use communication updates to find an optimal or a near optimal solution to the optimization problem, then apply control inputs to drive the system to the solution directly. This methodology is used in applications such as multi-robot motion planning [5], [8].

We observe that the two methodologies are essentially trade-off strategies that result from different assumptions on communication and execution cost. In particular, the pure execution strategy implicitly assumes that communication is much more expensive than execution, and hence the strategy surrenders the use of communications for a lower cost. For example, for underwater robots [9], communications may involve moving to the surface, which comes with a high risk of being detected by adversarial agents, while operating in the deep water is relatively safe and hence more desirable. Conversely, the pure communication strategy can be resulted from assuming execution cost is much higher than communication cost. One example is light-weighted quadcopters [14] in a laboratory environment. Communications are inexpensive since the communication bandwidth is high and the privacy requirement is low. However, the cost for moving is relatively high since the battery-life of these quadcopters are usually short.

In this paper, we make an attempt to address this aforementioned trade-off between communication and execution for multi-agent systems. We seek to understand the condition under which one methodology would be preferred over the other, and whether using a strategy that is in between the two could provide potential advantages. The main challenge to this work is that the two methodologies have been developed independently at large. Therefore, they have different underlying assumptions and performance metrics. This makes it difficult to directly compare one algorithm against another.

To focus on the trade-off between communication and execution, we restrict our consideration to a class of commonly studied multi-agent system problems where the objective function is the sum of a set of symmetric, edgewise potential functions [6], [10]. For this class of problems, the gradient descent algorithm can often be distributedly executed through a weighted consensus protocol [6]. Therefore, the gradient descent algorithm is a valid representative of the pure execution strategy. On the other hand, Newton's method is generally known to provide faster convergence compared to the gradient descent algorithm [4]. However, it involves an inversion of the Hessian matrix of the objective function, which often requires global information. Therefore, communications are needed for performing Newton updates. This makes Newton's method an appropriate proxy for the pure communication strategy, especially for quadratic problems. To solve for the Newton updates, we design a distributed communication update procedure based on the gradient descent algorithm. Inspired by truncated Newton's method [11], we discontinue the communication procedure after a fixed number of updates to execute the resulting inexact Newton update, and iterate this process. This algorithm provides a mechanism for allocating efforts between

*This work was sponsored by Grant No. W911NF-17-2-0181 from the U.S. Army Research Laboratory DCIST CRA.

[†]Anqi Li and Magnus Egerstedt are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: {anqi.li, magnus}@gatech.edu

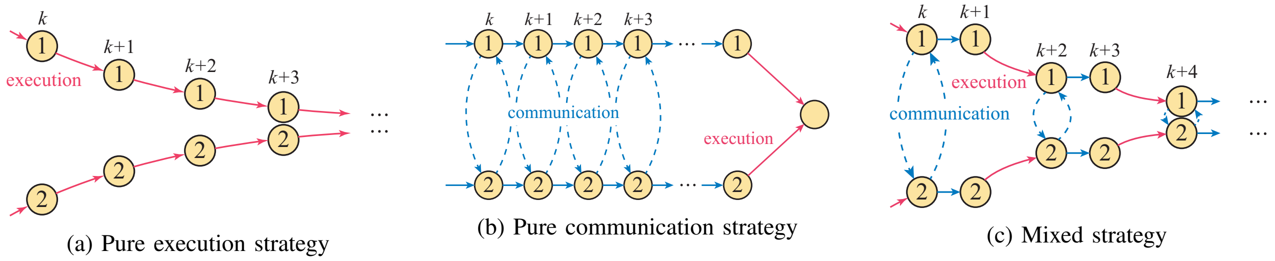


Fig. 1: Strategies for solving a 2-agent consensus problem: (a) the pure execution strategy where optimization updates are directly taken as control inputs, (b) the pure communication strategy where the optimization problem is solved through communication before a final execution, and (c) the mixed strategy where the communication updates (blue) and execution updates (red) are interleaved. To study the trade-off between communication and execution, we propose a general strategy that (a), (b) and (c) are special cases of it.

communication and execution through the truncation value. It also provides a consistent metric for comparing different trade-off strategies.

It is worth mentioning that this paper differs from the broad literature of distributed optimization [2], [3], [16], [17] for the following reason. In the distributed-optimization, the goal is to design an algorithmic procedure to find an optimal or near-optimal solution. As a result of that, there are usually no physical concepts of communication and execution. The performance of algorithms are often evaluated by their convergence rates. This paper, in contrast, focuses on understanding how to analyze the actual incurred costs of optimization algorithms that interleave communications and executions. One of the exceptions is the adaptive cost framework [1], which shares a similar insight to this paper. The difference is that it considers the trade-off between communication cost and computation cost rather than execution.

The main contribution of this paper is three-fold. Firstly, we discover the fundamental trade-off between communication and execution overhead which is implicitly considered in most multi-agent algorithms, and introduce a performance metric for evaluating the trade-off strategies. Secondly, we propose an algorithm that supports tunable levels of trade-off between communications and executions to represent different trade-off strategies for a class of commonly studied multi-agent problems. Thirdly, we analyze the convergence properties of the proposed algorithm for the consensus problem, and derive the optimal strategies for different assumptions of communication and execution cost.

II. Background and Problem Statement

We consider a multi-agent system consisting of N agents. The agents are indexed over $i \in [N] = \{1, \dots, N\}$. Each agent i has an associated state vector $x_i \in \mathbb{R}^d$. The agents are modeled as discrete-time systems with dynamics,

$$x_i^+ = x_i + u_i, \quad \forall i \in [N], \tag{1}$$

where x_i^+ is the state of agent i at the next time step. Let $x = [x_1^\top, \dots, x_N^\top]^\top \in \mathbb{R}^{Nd}$ and $u = [u_1^\top, \dots, u_N^\top]^\top \in \mathbb{R}^{Nd}$ denote the ensemble state vector and control input for the multi-agent system, respectively.

In order to focus on the trade-off between communication and execution, we assume that the system is associated with a fixed, connected, and undirected underlying communication graph. The communication graph structure for the system is defined as $\mathcal{G} = (V, E)$, where $V = [N]$ is the vertex set consisting of N agents and $E \subseteq V \times V$ is the set of edges denotes the communication links between agents. We denote the set of neighbors of each agent i as,

$$\mathcal{N}_i = \{j \in V : (i, j) \in E\}. \tag{2}$$

The agents are assumed to be equipped with sensors to measure the state displacement with their neighbors, i.e. $(x_i - x_j)$, for all $j \in \mathcal{N}_i$. However, each agent does not have the sensing modality to measure the state x_i directly. For example, in multi-robot systems, the robots equipped with laser scanners can measure the relative positions of their neighbors, while the absolute positions can not be measured.

We denote the graph Laplacian for the communication graph as $L \in \mathbb{R}^{N \times N}$, given by

$$L = [l_{ij}], \quad l_{ij} = \begin{cases} \deg(i) & \text{if } i = j, \\ -1 & \text{if } (i, j) \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $\deg(i) = |\mathcal{N}_i|$ is the number of agents that are adjacent to agent i. Let $\mathcal{G}^o = (V, E^o)$ be a directed graph introduced through $\mathcal{G}$ by arbitrarily assigning orientation to the edges. The incidence matrix $D \in \mathbb{R}^{N \times |E|}$ for $\mathcal{G}^o$ is given by,

$$D = [d_{ij}], \quad d_{ij} = \begin{cases} -1 & \text{if } v_i \text{ is the tail of } e_j \in E^o, \\ 1 & \text{if } v_i \text{ is the head of } e_j \in E^o, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

It can be shown that $L = DD^\top$.

We are interested in solving optimization problems over the multi-agent system. In particular, we consider the objective function to be the sum of the set of symmetric, twice-differentiable, pairwise potential functions $\mathscr{E}(\|x_i - x_j\|)$ between every pair of agents i and j that are adjacent in the communication graph, i.e.

$$\min_x J(x) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \mathscr{E}(\Delta_{ij}), \tag{5}$$

where $\Delta_{ij} = \|x_i - x_j\|$.

This is a general formulation for a lot of multi-agent problems, such as rendezvous and formation controls, etc. For the sake of simplicity, we assume that the agents are in a 1-d space, i.e. $d = 1$, when deriving the gradient and Hessian for the cost function $J(x)$. We refer the readers to [15] for the general expressions when $d > 1$.

One common feature for this class of problems is that, the gradient descent update rule for (5) corresponds to a weighted consensus protocol,

$$\begin{aligned}\nabla J(x) &= D \,\mathrm{diag}\left\{ \left[\frac{1}{\|x_i - x_j\|} \frac{\partial \mathscr{E}}{\partial \Delta_{ij}} \right]_{(i,j)\in E} \right\} D^\top x \\ &\doteq D W(x) D^\top x = L_w x.\end{aligned} \tag{6}$$

Then by chain rule, we have,

$$\begin{aligned}\nabla^2 J(x) &= D \left(W(x) + \mathrm{diag}\left\{ \left[\frac{\partial w_{(i,j)}}{\partial x_i} (x_i - x_j) \right]_{(i,j)\in E} \right\} \right) D^\top \\ &\doteq D V(x) D^\top = L_v,\end{aligned} \tag{7}$$

where $w_{(i,j)}$ denotes the diagonal entry of W corresponding to the edge (i, j). By their definitions, $W(x), V(x) \in \mathbb{R}^{|E|\times|E|}$ are diagonal matrices.

Note that although we restrict our attention to the problems in the form of (5), the algorithm introduced in this paper is in fact applicable to *any* objective function that satisfies the following condition,

$$\begin{aligned}\nabla J(x) &= (D \otimes I_d) W(x) (D^\top \otimes I_d) \doteq L_w x, \\ \nabla^2 J(x) &= (D \otimes I_d) V(x) (D^\top \otimes I_d) \doteq L_v,\end{aligned} \tag{8}$$

where $\otimes$ denotes the Kronecker product, and $W(x) = \text{blk-diag}(\{W_{(i,j)}(x_i,x_j)\}_{(i,j)\in E}) \in \mathbb{R}^{|E|d\times|E|d}$ and $V(x) = \text{blk-diag}(\{V_{(i,j)}(x_i,x_j)\}_{(i,j)\in E}) \in \mathbb{R}^{|E|d\times|E|d}$ are block diagonal matrices, where each diagonal block is a $d \times d$ matrix. One can see that when $d = 1$, the condition is satisfied for the objective function (5). Further, it is shown in [15] that the gradient vector and the Hessian matrix satisfy the condition (8) when $d > 1$. The conditions can be interpreted as a second-order structure perseverance condition of the objective function, i.e. its gradient vector and Hessian matrix respect the sparsity structure of the communication graph.

III. Distributed Truncated Newton's Method

In this section, we present an algorithm that deliberately considers the trade-off between communication and execution. The gradient descent algorithm is considered as a representative of the pure execution strategy since it can be executed distributedly under our assumptions, while Newton's method is viewed as a proxy for the pure communication strategy due to its faster convergence compared to the gradient-descent method [4]. However, Newton's method can not be implemented distributedly in general since solving for the Newton's step involves solving a group of linear equations. The motivation of the proposed algorithm comes from the truncated Newton's method [11] which has an inner loop that approximately solves for the Newton's step in an iterative manner. The truncated Newton's method pauses the inner loop after a fixed number of iteration and then uses the resulting inexact Newton's step for the optimization update. Then, the process is repeated until convergence. In the proposed algorithm, we design an inner loop that computes the Newton update through local communications. By the nature of the truncated Newton's method, we can adjust the amount of effort allocated to the communication updates by changing the truncation value – with more communication iterations in the inner loop, we get a better estimate of the Newton's step, and hence get faster convergence rate with respect to execution updates.

By plugging in the Hessian and the gradient (8), the update rule for Newton's method becomes,

$$\begin{aligned}x^+ &= x - y, \\ y &= (L_v^\dagger L_w) x,\end{aligned} \tag{9}$$

where $L_v^\dagger$ denotes the Moore–Penrose pseudoinverse of the weighted graph Laplacian matrix L_v. It should be noticed that the update rule (9) can not be directly computed through local interactions since $L_v^\dagger$ does not have the same sparsity structure as the communication graph. Therefore, solving for (9) requires global information that is not directly available to each of the agents.

We propose to solve for the Newton update y iteratively through the gradient descent algorithm. Notice that y is the minimizer of the local quadratic approximation of the objective function at x, i.e.

$$y = \arg\min_z \frac{1}{2} z^\top L_v z - x^\top L_w z. \tag{10}$$

Therefore, the Newton update y can be solved through gradient descent updates with respect to (10),

$$y^+ = y - \eta L_v y + \eta L_w x, \tag{11}$$

where $\eta > 0$ is the step size of the gradient descent algorithm.

Then, the algorithm can update the state variables by executing the update rule for the state variables,

$$x^+ = x - y. \tag{12}$$

There are a few important notes to the proposed update rules. First, although the update rules (11) and (12) are presented in a centralized manner using the ensemble state, the update rules can be implemented in a fully distributed way. For each agent i, the update rules for x_i and y_i are,

$$\begin{aligned}x_i^+ &= x_i - y_i \\ y_i^+ &= y_i - \eta \sum_{j\in\mathcal{N}_i} \left(V_{(i,j)} (y_i - y_j) - W_{(i,j)} (x_i - x_j) \right),\end{aligned} \tag{13}$$

where $W_{(i,j)}$ and $V_{(i,j)}$ denotes the block diagonal entry corresponding to the edge (i, j) of W and V, respectively. Therefore, only local interactions are required to compute the updates. The distributed implementation of the truncated Newton's method is presented in Algorithm 1.

The second note is that the update rules for solving (11) and (12) are fundamentally different in the sense that (12) involves updating the actual state variable, which means that one need to actually drive the agent using controls. While (11) only involves updating the local decision variable, hence y can be updated through communications without applying actual control inputs.

Algorithm 1 Truncated Newton's Method

```
Require: $K \geq 1, \eta > 0, \mathcal{N}_i$
  $y_i \leftarrow 0$
  while True do
    for $k = 1$ to $K$ do
      evaluate $W_{(i,j)}$, $V_{(i,j)}$
      $y_i^+ \leftarrow y_i - \eta \sum_{j\in\mathcal{N}_i} \left( V_{(i,j)}(y_i - y_j) - W_{(i,j)}(x_i - x_j) \right)$
    end for
    $x_i \leftarrow x_i - y_i$
  end while
```

IV. Convergence Rate of the Truncated Newton's Method

In this section, we analyze the convergence rate of the algorithm for the 1-dimensional consensus problem. We choose the 1-d consensus problem for the following two reasons. First, it is an important proxy for a large number of commonly studied multi-agent problems such as formation controls and distributed estimation [10]. Moreover, the consensus problem itself is of special importance since it can be served as an algorithmic building block for a lot of distributed optimization algorithms.

However, before analyzing the convergence property, we need to first introduce how to define the convergence rate of algorithms in the situation we are considering. As is mentioned in Section III, we observe that control and coordination for multi-agent systems involve two kinds of fundamentally different updates – communication updates and execution updates. Communication updates involve exchanging messages about local decision variable between adjacent agents in the communication graph, while execution updates involve exerting control inputs using only on-board sensor measurements. The cost of communication and execution is very application dependent. Therefore, it is necessary to take into account the distinctions between communication and execution updates, as well as the setting of the problem, when analyzing the performance of the algorithm.

To analyze the convergence rate of the algorithm, we assume that the cost for each communication update and execution update is c_c and c_e. The cost can encode safety, time consumption, energy consumption, and disturbance intensity, etc. We are interested in the convergence rate of the algorithm with respect to the total cost incurred, rather than the number of update steps. Formally, let n_c and n_e be the number of communication and execution steps. The total cost is given by,

$$c = c_c n_c + c_e n_e \tag{14}$$

We study the 1-d consensus problem as a proxy to a class of multi-agent problems discussed in Section II. The goal of the 1-d consensus problem is to drive the agents to a common coordinate on the real line. One way of approaching the problem to optimize the objective function,

$$\min_x J(x) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}\|x_i - x_j\|^2 = \frac{1}{2}x^\top Lx. \tag{15}$$

It can be shown that $\nabla J = Lx$, and $\nabla^2 J = L$. Hence, the objective function in (15) satisfies the constraints (8) we defined in the problem statement. Therefore, we can solve the consensus problem through the proposed distributed truncated Newton's method.

Let each agent maintain a local decision variable y_i as the estimate to the Newton's step. We can solve for the Newton update y through the gradient descent algorithm,

$$y^+ = y - \eta Ly + \eta Lx, \tag{16}$$

where $y = [y_1, y_2, \ldots, y_n]^\top$. Assume that the communication updates (16) are executed for K times before each execution step, we have,

$$\begin{aligned} y^{K+} &= (I - \eta L)^K y + \sum_{t=0}^{K-1}(I - \eta L)^{K-t-1}\eta Lx \\ &= (I - (I - \eta L)^K)x + (I - \eta L)^K y. \end{aligned} \tag{17}$$

Therefore, the dynamics of the discrete-time system can be rewritten as,

$$\begin{bmatrix} x^+ \\ y^{K+} \end{bmatrix} = \begin{bmatrix} (I-\eta L)^K & -(I-\eta L)^K \\ I-(I-\eta L)^K & (I-\eta L)^K \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{18}$$

The convergence rate for the truncated Newton's method is summarized in the following Theorem.

Theorem IV.1. *Given the consensus problem (15) with a fixed, connected and undirected communication graph. Assume that the step size η is chosen such that $0 < \eta < \frac{1}{\lambda_{\max}(L)}$. Let the second largest eigenvalue of matrix $(I - \eta L)$ be λ. Then λ satisfies $0 < \lambda < 1$, and the truncated Newton's method converges to the consensus subspace $\mathscr{C} = \{(x,y) : x \in \operatorname{span}\{\mathbb{1}_N\}, y \in \operatorname{span}\{\mathbb{1}_N\}\}$ with a rate of convergence (with respect to the total cost) dictated by $\gamma_{TN} = \lambda^{\frac{K}{2Kc_c+2c_e}}$.*

Proof. Let $\delta \in \mathbb{R}^{2N}$ denote the orthogonal projection of the vector $[x^\top \; y^\top]^\top$ onto the orthogonal complement of the consensus subspace,

$$\delta = \begin{bmatrix} x \\ y \end{bmatrix} - \frac{1}{N}\begin{bmatrix} (\mathbb{1}^T x)\mathbb{1} \\ (\mathbb{1}^T y)\mathbb{1} \end{bmatrix}. \tag{19}$$

We need to show that δ converges to zero with a rate of convergence that is dictated by $\lambda^{\frac{K}{2Kc_c+2c_e}}$.

For notational convenience, let $M = (I - \eta L)^K$. By assumption, L has one eigenvalue at 0 and $N-1$ eigenvalues in the range of $(0, \frac{1}{\eta})$. Hence, $0 < \lambda < 1$ and the second largest eigenvalue of M is λ^K. The dynamics of the projection δ is,

$$\delta^+ = \begin{bmatrix} M & -M \\ I-M & M \end{bmatrix}\delta \doteq A\delta. \tag{20}$$

That is, δ satisfies the same dynamics as $[x^\top \; y^\top]^\top$. By facts from linear algebra, the eigenvalues of A are given by $\{\mu \pm i\sqrt{\mu - \mu^2} : \mu \text{ is an eigenvalue of } M\}$. Hence, the matrix A has $2N-2$ stable eigenvalues and an eigenvalue λ_0 of value 1 that has geometric multiplicity of 1 but algebraic multiplicity of 2. However, the span of the generalized eigenvectors corresponding to λ_0 is $\mathscr{C}$. But $\delta \in \mathscr{C}^\perp$. Hence, δ decades exponentially. Further, the largest non-one singular value for the matrix A is $|\lambda^K \pm i\sqrt{\lambda^K - \lambda^{2K}}| = \lambda^{K/2}$. Hence,

$$\|\delta^+\| \leq \lambda^{K/2}\|\delta\|. \tag{21}$$

Since each update of δ involves K communication updates and 1 execution update, the convergence rate of the algorithm is dictated by,

$$\gamma_{TN} = \left(\lambda^{K/2}\right)^{\frac{1}{Kc_c+c_e}} = \lambda^{\frac{K}{2Kc_c+2c_e}}. \tag{22}$$

□

On the other hand, the gradient-descent update law for the objective function is,

$$x^+ = x - \eta Lx = (I - \eta L)x. \tag{23}$$

The gradient-descent update rule respects the communication structure as the update ηLx can be computed through local interactions. The convergence rate for the gradient descent algorithm is summarized in the following Theorem.

Theorem IV.2. *Given the consensus problem (15) with a fixed, connected and undirected communication graph. Assume that the step size η is chosen such that $0 < \eta < \frac{1}{\lambda_{\max}(L)}$. Let the second largest eigenvalue of matrix $(I - \eta L)$ be λ. Then λ satisfies $0 < \lambda < 1$, and the gradient descent algorithm converges to the consensus subspace $\mathscr{C} = \operatorname{span}\{\mathbb{1}_N\}$ with a rate of convergence that is dictated by $\gamma_{GD} = \lambda^{\frac{1}{c_e}}$.*

Proof. Since $0 < \eta < \frac{1}{\lambda_{\max}(L)}$, the eigenvalues of the matrix $(I - \eta L)$ has one eigenvalue at 1 and $N-1$ eigenvalues in the range of $(0,1)$. Therefore, $0 < \lambda < 1$. Let $\delta \in \mathbb{R}^N$ denote the orthogonal projection of the vector x onto the orthogonal complement of the consensus subspace. According to [10],

$$\|\delta^+\| \leq \lambda\|\delta\|. \tag{24}$$

Since each update of δ involves a cost of c_e, the convergence rate of the algorithm is dictated by,

$$\gamma_{GD} = \lambda^{\frac{1}{c_e}}. \tag{25}$$

Therefore, the algorithm converges to the consensus subspace with a rate of convergence dictated by $\gamma_{GD} = \lambda^{\frac{1}{c_e}}$. □

We are interested in how the performance of the proposed truncated Newton's method and the gradient descent algorithm are related to the parameters K, c_c and c_e. Since $0 < \lambda < 1$, the proposed algorithm outperforms the gradient descent algorithm when,

$$c_c \leq \frac{K-2}{2K}c_e. \tag{26}$$

This provides an analytic criterion on choosing between pure execution strategy and the strategies involve communications. The result is consistent with the common intuition that the pure execution strategy is preferred when communications are very expensive and the communications should be taken use of when the cost of execution is relatively high.

Another observation is that the convergence rate is monotonically increasing with respect to K. This means that regardless of the value of c_c and c_e, it is always better to run the communication update until convergence for the truncated Newton's method, rather than terminate early and take actions based on a premature estimate of the Newton update. This observation, combined with the previous fact, implies that the most efficient solution to the consensus problem is either (i) conduct pure communication-based update until convergence then drive the system directly to the optimum (when moving is expensive) or (ii) perform execution updates always (when communication is expensive). This observation aligns well with existing literature on multi-agent systems, where most of the work either focuses on pure execution updates or pure communication updates.

It is also noteworthy that when communication and execution are the same and both equal to 1, the convergence rate of the algorithm is equivalent to the convergence rate in the optimization sense. In this case, we have,

$$\gamma_{TN} = \lambda^{\frac{K}{2K+2}} \geq \lambda = \gamma_{GD} \tag{27}$$

This means that in the optimization point of view, the distributed truncated Newton's method is always inferior to the gradient descent algorithm. However, we have seen from the previous discussion that the truncated Newton's method can indeed over-perform the gradient descent algorithm when the actual cost is considered. Therefore, approaching the problem from a pure optimization point of view may fail to fully address the practical situation, and hence may lead to sub-optimal strategies.

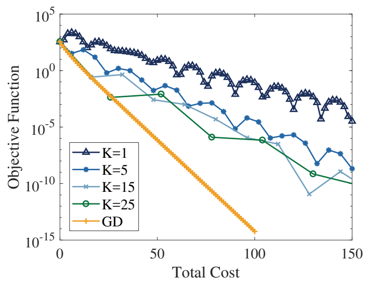
V. Simulation Results

In this section, we study the performance of the proposed algorithm through numerical simulation. We consider two multi-agent problems, a 1-d consensus problem, and a 2-d formation control problem. In both problems, we consider a team of 6 agents with a fixed communication graph structure shown in Fig. 3. We use the 1-d consensus problem to validate the theoretical results developed in Section IV, and use the formation control problem to show the applicability of the proposed algorithm beyond the consensus problem.

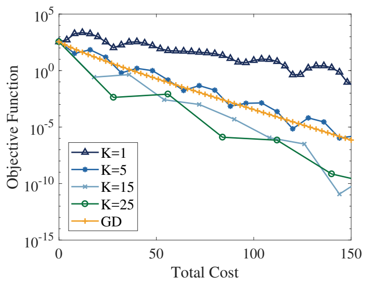
A. Consensus

We first study a 1-d consensus problem to validate the theoretical results we developed in Section IV. We consider a team of 6 agents with the communication graph shown in Fig. 3a. The agents are initialized with the ensemble state $x_0 = [-10, -6, -1, 5, 6, 10]^\top$. The goal for the agents is to rendezvous at a single point on the 1-d real line. As is discussed in Section IV, the instantaneous state of the system is evaluated by the objective function $J(x) = \frac{1}{2}x^\top Lx$.

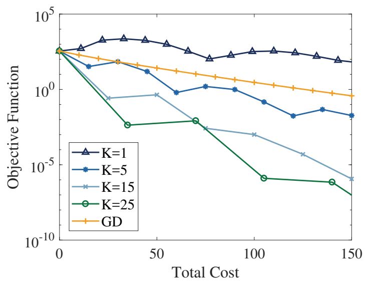
We are interested in the convergence rate of the objective function under the algorithms we discussed, namely, the truncated Newton's method and the gradient descent



(a) $c_c = 1, c_e = 1$

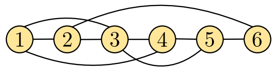


(b) $c_c = 1, c_e = 3$

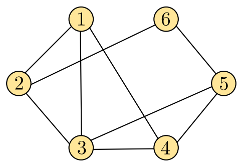


(c) $c_c = 1, c_e = 10$

Fig. 2: The value of the objective function with respect to the total cost incurred for the 1-d consensus problem. The truncated Newton's method gains more advantage over the gradient descent algorithm as c_e/c_c and K increases. The performance of the truncated Newton's method is improved as K increases. The gradient descent algorithm out-performs the truncated Newton's method when $c_c = c_e$. Note for the logarithmic scale. Notice that different curves have different spacing since the cost of operations is different across figures, and the objective function is only evaluated at execution steps.



(a) Consensus



(b) Formation Control

Fig. 3: The communication graph structure for (a) the consensus problem and (b) the formation control problem in the numerical simulation.

algorithm. We study how the performance of the truncated Newton's method varies with respect to the truncation value K under different combinations of the communication cost c_c and execution cost c_e. In particular, we consider the case when $K \in \{1, 5, 15, 25\}$, and $(c_c, c_e) \in \{(1,1), (1,3), (1,10)\}$. The performance of the truncated Newton's method and the gradient descent algorithm is also compared. The step sizes of all the algorithms are fixed as $\eta = 0.1$. The performance of the algorithms with respect to the total incurred cost is presented in Fig. 2. Notice that different curves have different spacing since the objective function is only evaluated during the execution step, and as a result, the truncated Newton's method has fewer data points. Also note that the objective function values are in logarithmic scale.

In Fig. 2, we observe that the gradient descent algorithm converges linearly for the consensus problem. The truncated Newton's Method also converges in a linear manner for the consensus problem. The convergence rate of the truncated Newton's is monotonically improving with respect to K. However, the effect of K decreases as K becomes larger. For example, the curves for $K = 15$ and $K = 25$ are relatively close in all three figures.

In the case of $c_c = c_e = 1$ (Fig. 2a), the gradient descent algorithm out-performs the truncated Newton's method for all values of K, which is consistent with the analysis in Section IV. We see that the truncated Newton's method with $K = 5$ and the gradient descent method converges in a similar rate when $c_c = 1$ and $c_e = 3$. This is also consistent with (26) since $\frac{2K}{K-2} = 3.3$. We also see that the truncated Newton's method gains more advantage over the gradient descent algorithm as the ratio c_e/c_c increases.

B. Formation Control

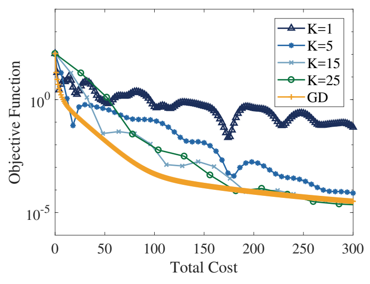
To evaluate the applicability of the proposed algorithm to problems beyond the consensus problem, we consider a 2-d formation control problem. This problem is different from the consensus problem since the objective function is no longer quadratic, and hence, Newton's method loses its exactness when applied to this problem. In the formation control problem, the agents are required to form a regular hexagonal formation with length size 1. The agents are initialized with a regular hexagonal formation, but with a length size of 2. One way to encode the formation control problem is to define the objective function as in [10],

$$J(x) = \frac{1}{4} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - d_{ij}^2)^2, \tag{28}$$

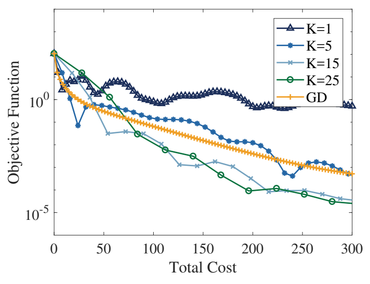
where d_{ij} is the distance between agent i and agent j in the desired formation. Then, the objective function satisfies the condition (8) with

$$\begin{aligned} L_w &= (D \otimes I_2)\,\text{blk-diag}\left(\left\{\left(\|x_i - x_j\|^2 - d_{ij}^2\right) I_2\right\}_{(i,j)\in E}\right)(D^\top \otimes I_2), \\ L_v &= (D \otimes I_2)\,\text{blk-diag}\Big(\Big\{2\,(x_i - x_j)\,(x_i - x_j)^\top + \\ &\qquad \left(\|x_i - x_j\|^2 - d_{ij}^2\right) I_2\Big\}_{(i,j)\in E}\Big)(D^\top \otimes I_2). \end{aligned} \tag{29}$$

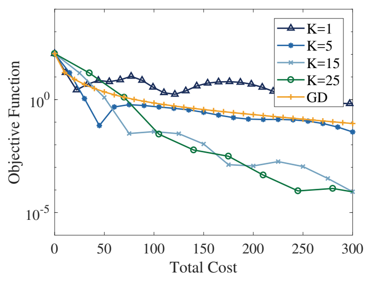
In the simulation, we fix the step size to be $\eta = 0.01$. As is in the consensus problem, we also consider $K \in \{1, 5, 15, 25\}$, and $(c_c, c_e) \in \{(1,1), (1,3), (1,10)\}$. The performance of the algorithms with respect to the total incurred cost is presented in Fig. 4. We see that the relative performance of the algorithms is similar – (i) the truncated Newton's method gains more advantage over the gradient descent algorithm as c_e/c_c and K increases, (ii) the performance of the truncated



(a) $c_c = 1, c_e = 1$

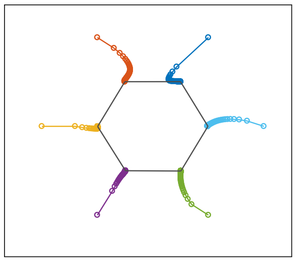


(b) $c_c = 1, c_e = 3$



(c) $c_c = 1, c_e = 10$

Fig. 4: The value of the objective function with respect to the total cost incurred for the 2-d formation control problem. Similar observation is obtained compard to Fig. 2, although the algorithms no longer converge linearly.


(a) Gradient Descent

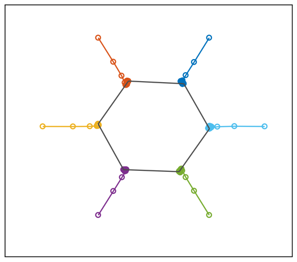

(b) Truncated Newton $K = 25$

Fig. 5: The trajectories of the agents in the formation control problem under (a) the gradient descent algorithm and (b) the truncated Newton's method with $K = 25$. The agents take more direct paths in the case of truncated Newton's method when $K = 25$. The gray hexagon denotes the final formation.

Newton's method is improved as K increases, (iii) the gradient descent algorithm out-performs the truncated Newton's method when $c_c = c_e$. In contrast to the consensus problem, however, the gradient descent algorithm no longer converges linearly. The reason is that the formation control problem (28) is no longer strongly convex. Therefore, a fixed step size can no longer achieve linear convergence.

To get a better intuition on how the execution cost can be traded off by the communication cost, we present the actual trajectory taken by the agents under the gradient descent algorithm and the truncated Newton's method with $K = 25$. The trajectories are shown in Fig. 5. We see that the agents take more direct paths under the truncated Newton's method. Therefore, by paying the cost of communication, the agents can indeed carry out execution steps more efficiently.

VI. Conclusion

This paper focuses on understanding the trade-off between communication and execution overhead for control of multi-agent systems. To provide a consistent metric for different trade-off strategies, we propose an algorithm based on the truncated Newton's method. The proposed algorithm can adjust the amount of effort allocated to communication and execution. Convergence analysis of the algorithm for the consensus problem shows that the optimal strategy is either pure communication-based or pure execution-based, which aligns well with existing multi-agent approaches. The performance of the proposed trade-off strategy is validated through numerical simulation.

References

[1] A. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei. Balancing communication and computation in distributed optimization. *IEEE Transactions on Automatic Control*, 2018.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt. A distributed version of the hungarian method for multirobot assignment. *IEEE Transactions on Robotics*, 33(4):932–947, 2017.

[6] J. Cortés and M. Egerstedt. Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6):495–503, 2017.

[7] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage Control for Mobile Sensing Networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[8] Y. Diaz-Mercado and M. Egerstedt. Multirobot mixing via braid groups. *IEEE Transactions on Robotics*, 33(6):1375–1385, 2017.

[9] G. A. Hollinger, S. Choudhary, P. Qarabaqi, C. Murphy, U. Mitra, G. S. Sukhatme, M. Stojanovic, H. Singh, and F. Hover. Underwater data collection using robotic sensor networks. *IEEE Journal on Selected Areas in Communications*, 30(5):899–911, 2012.

[10] M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

[11] S. G. Nash. A survey of truncated-newton methods. *Journal of computational and applied mathematics*, 124(1-2):45–59, 2000.

[12] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[13] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004.

[14] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3299–3304. IEEE, 2017.

[15] Z. Sun and T. Sugie. Identification of hessian matrix in distributed gradient-based multi-agent coordination control systems. *arXiv preprint arXiv:1805.02832*, 2018.

[16] E. Wei and A. Ozdaglar. Distributed alternating direction method of multipliers. 2012.

[17] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed newton method for network utility maximization–i: Algorithm. *IEEE Transactions on Automatic Control*, 58(9):2162–2175, 2013.