A Distributed Simplex Algorithm for Degenerate Linear Programs and Multi-Agent Assignments

Mathias Bürger [a] Giuseppe Notarstefano [b] Francesco Bullo [c] Frank Allgöwer [a]

[a] *Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany.*

[b] *Department of Engineering, University of Lecce, Via per Monteroni, 73100 Lecce, Italy.*

[c] *Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, CA 93106, USA.*

Abstract

In this paper we propose a novel distributed algorithm to solve degenerate linear programs on asynchronous peer-to-peer networks with distributed information structures. We propose a distributed version of the well-known simplex algorithm for general degenerate linear programs. A network of agents, running our algorithm, will agree on a common optimal solution, even if the optimal solution is not unique, or will determine infeasibility or unboundedness of the problem. We establish how the multi-agent assignment problem can be efficiently solved by means of our distributed simplex algorithm. We provide simulations supporting the conjecture that the completion time scales linearly with the diameter of the communication graph.

Key words: Distributed Simplex, Distributed Linear Programming, Asynchronous Algorithm, Multi-Agent Task Assignment.

⋆ M. Bürger and F. Allgöwer were supported by the German Research Foundation within the Cluster of Excellence in Simulation Technology (EXC 310/1) and the Priority Program SPP 1305. The research of G. Notarstefano has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 224428 (CHAT) and n. 231378 (CO3AUV) and from the Italian Minister under the national project "Sviluppo di nuovi metodi e algoritmi per l'identificazione, la stima bayesiana e il controllo adattativo e distribuito." The work by F. Bullo is supported by NSF Award CPS-1035917 and ARO Award W911NF-11-1-0092. The material in this paper was partially presented at the 2011 American Control Conference, June 29 - July 1, 2011, San Francisco.

Email addresses: mathias.buerger@ist.uni-stuttgart.de (Mathias Bürger), giuseppe.notarstefano@unile.it (Giuseppe Notarstefano), bullo@engineering.ucsb.edu (Francesco Bullo), frank.allgower@ist.uni-stuttgart.de (Frank Allgöwer).

1 Introduction

The increasing interest in performing complex tasks via multi-agent systems has raised the interest in solving distributed optimization problems. The fundamental paradigms in distributed computation are that: (i) information, relevant for the solution of the problem, is distributed all over a network of processors with limited memory and computation capability, and (ii) the overall computation relies only on local computation and information exchange amongst neighboring processors.

Although parallelized optimization algorithms have been studied for a long time, see e.g., [4], the multi-agent perspective has recently become subject of renewed interest. Several algorithms ranging from subgradient to Quasi-Newton methods were proposed to solve strictly convex programs [17], [22], [5]. The named approaches rely on strict convexity and cannot efficiently handle linear programs. Algorithms explicitly dedicated to distributed linear programming are, to the best of our knowledge, still rare. Simplex algorithms are of outstanding importance in linear programming and have been proven extremely successful. This success motivates searching for a distributed realization of it. While the idea of parallelizing the computation of the simplex algorithm has found some attention in the computer science literature, see e.g., [13], [20], a first contribution, explicitly dedicated to multi-agent networks, is given in [18], where *distributed abstract programs* are introduced as a general class of optimization problems including linear programs. A *constraints consensus* algorithm is proposed to solve them. We follow here the trail opened in [18].

A class of problems closely related to linear programs are multi-agent assignment problems. Assignment problems are combinatorial problems with exact linear programming representations, see [7] for an introduction. In contrast to general distributed linear programming, distributed multi-agent assignment problems have attained significant attention in the recent literature, and various problem formulations were studied [16], [8], [19]. To solve the standard assignment problem in a fully distributed way, suitable versions of the Auction Algorithm [3] combined with a max consensus law have been recently proposed in [6] or [21]. The progress made on distributed assignment problems has not been translated into a general theory for distributed linear programming.

The contributions of this paper are as follows. In a first step, the multi-agent assignment problem is used to derive a more general formulation of distributed linear programs. The assignment problem directly motivates the distribution of information throughout the network and emphasizes an important challenge for distributed algorithms: the non-uniqueness of optimal solutions or, more formally, the degeneracy of optimization programs. As main contribution, we introduce a distributed version of the well known simplex algorithm to solve these degenerate linear programs in asynchronous networks with time-varying directed communication topology. Our algorithm has the following structure. Each agent has a candidate basis in its memory that it exchanges with its neighbors. We propose a new *pivot* operation, which is a refined version of the operation proposed in [14], for the local basis update performed by each agent. The *pivot* operation uses a *lexicographic sorting* of the problem columns and a *lexicographic ratio test*. It is performed on a subset of columns given by its basis, its neighbors' candidate bases and its original columns it has been assigned. We prove that a network of agents running our algorithm will agree in finite time on a common

optimal solution of the linear program even if it is highly degenerate. The proposed algorithm is inspired by the constraints consensus algorithm proposed in [18]. We show that, if constraints consensus is implemented for non-degenerate linear programs, then the proposed distributed simplex turns to be its dual implementation. Finally, we show how to apply the algorithm to the multi-agent assignment problem, a highly degenerate linear program. We provide numerical experiments for the distributed assignment problem supporting our conjecture that the distributed simplex algorithm scales linearly with the diameter of the graph in networks with fixed topologies. We want to point out that our objective is not to improve the convergence speed of a centralized algorithm, but rather to design a stable and robust distributed negotiation protocol which allows a group of agents in a peer-to-peer network to solve a complex optimization problem without any coordination unit.

Before presenting the paper's organization, we want to discuss shortly the relation of our Distributed Simplex to known methods for linear programming. It is worth mentioning that our approach inherently differs from the classical decomposition methods such as the Dantzig-Wolfe or Bender's decomposition [9]. Classical decomposition methods utilize a master/subproblem structure, whereas the novel approach uses a peer-to-peer communication between identical agents without any coordination unit. Our algorithm also clearly differentiates form other parallel algorithms for linear programming, in particular from [10] and [20]. The two algorithms [10] and [20] require a "two-step" communication. In a first step, a consensus-like algorithm is performed over the whole network to determine the column with minimal reduced cost. In a second communication step, this column is then transmitted to all agents for a basis update. In contrast, a basis update in our algorithm happens directly after any information exchange between neighboring agents, and does not require agreement over the complete network.

The remainder of the paper is organized as follows. Section 2 introduces the multi-agent assignment as a motivating problem class. The framework set by the assignment problem is used to formulate the notion of distributed linear programs in Section 3. Section 4 reviews some fundamental principles of degenerate linear programming and introduces a non-standard version of the simplex algorithm. The main results of the paper are presented in Section 5, where a distributed simplex algorithm for degenerate linear programs is proposed and analyzed. The usefulness of the algorithm is illustrated in Section 6, where its application to the class of multi-agent assignment problems is discussed and simulation results are provided for a time complexity analysis of the algorithm.

2 A Motivating Example for Distributed Linear Programming: the Multi-Agent Assignment

Distributing a number of tasks to a number of agents is one of the most fundamental resource allocation problems and has applications in numerous control systems. It serves as a motivating application in this paper and will lead to a general definition of distributed linear programs.

We shortly review the assignment problem here, following the standard reference [3]. The assignment problem can be illustrated with a bipartite assignment graph $\mathcal{G}_a = \{V_a, U_a; E_a\}$, where the node sets $V_a = \{1, \dots, N\}$ and $U_a = \{1, \dots, N\}$ represent the set of agents and tasks respectively. An edge $(i, \kappa) \in E_a$ exists if and only if agent i

can be assigned to the task κ. The cost $c_{i\kappa} \in \mathbb{Z}$ for agent i to perform the task κ is a weight on the edge (i, κ). Figure 1 illustrates the assignment graph. For each agent i and task κ a binary decision variable $x_{i\kappa} \in \{0, 1\}$ is introduced,

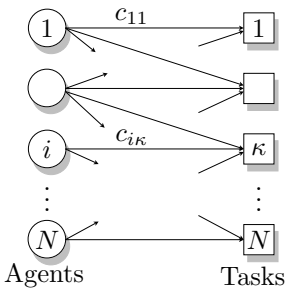

Fig. 1. Assignment Graph $\mathcal{G}_a$.

which is 1 if agent i is assigned to task κ and 0 otherwise. The assignment problem corresponds now to the *integer optimization problem* $\min_{x_{i\kappa}\in\{0,1\}} \sum_{(i,\kappa)\in E_a} c_{i\kappa}x_{i\kappa}$, with the constraints that a full assignment is achieved. The convex relaxation gives rise to the linear optimization problem [3]:

$$\begin{aligned} \min_{x\geq 0} \quad & \sum_{(i,\kappa)\in E_a} c_{i\kappa}x_{i\kappa} \\ \text{s.t.} \quad & \sum_{\{\kappa|(i,\kappa)\in E_a\}} x_{i\kappa} = 1, \quad \forall i \in \{1,\dots,N\}, \\ & \sum_{\{i|(i,\kappa)\in E_a\}} x_{i\kappa} = 1, \quad \forall \kappa \in \{1,\dots,N\}. \end{aligned} \tag{1}$$

It is well-known that (1) has always an optimal solution $x_{i\kappa} \in \{0, 1\}$, and that this solution corresponds exactly to the optimal assignment. Note that the linear program (1) has $n = |E_a|$ decision variables with $|E_a| \leq N^2$, and $d = 2N$ equality constraints.

Envisioning self-organizing systems, it is important to design algorithms to solve (1) only by negotiation between agents. We are interested in applications where only the agents can communicate and perform computations, and where no coordinator or mediator is involved. This setup is different from the one in [2], where the tasks serve as auctioneers and perform computations in the proposed Distributed Auction Algorithm. We assume that every agent initially knows the cost it takes for itself to perform a specific task. In particular, agent i knows all $c_{i\kappa}$ with $(i, \kappa) \in E_a$. This is a reasonable assumption in a multi-agent scenario. We want to stress that it is fundamental that the agents agree upon the same optimal assignment. In this respect, the assignment problem highlights an additional challenge for distributed optimization algorithms.

Remark 2.1 *In general,* (1) *has several optimal solutions x^* corresponding to different optimal assignments.* □

This property of the optimization problem is called *degeneracy*, and has strong implications on the design of distributed algorithms.

3 Distributed Linear Programming Set-Up

Throughout this paper we consider linear programs in the standard equality form

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax = b, \quad x \geq 0, \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{d\times n}$, $b \in \mathbb{R}^d$ and $c \in \mathbb{R}^n$, are the problem data and $x \in \mathbb{R}^n$ is the vector of decision variables. Note that the assignment problem (1) is a particular representation of (2). We assume in the following that $\text{rank}(A) = d$. We call a *column* of problem (2) a vector $h_i \in \mathbb{R}^{1+d}$ defined as

$$h_i := [\ c_i,\ A_i^T\ ]^T, \tag{3}$$

where $c_i \in \mathbb{R}$ is the i-th entry of c and $A_i \in \mathbb{R}^{d\times 1}$ is the i-th column of the matrix A. The set of all columns is denoted by $\mathbb{H} = \{h_i\}_{i\in\{1,\dots,n\}}$. For any subset $\mathbb{G} \subset \mathbb{H}$, the notation c_G and A_G refers to the cost vector and the constraint matrix, respectively constructed from the columns contained in $\mathbb{G}$. The same notation x_G is used to denote the corresponding decision variables. A linear program (2) is fully characterized by the pair $(\mathbb{H}, b)$.

Distributed Information Structure: Distributed linear programs are inherently associated with an initial information structure. We assume that every agent initially knows a subset of the problem columns, i.e., that there exists a unique partitioning $\mathcal{P} = \{\mathbb{P}^{[1]}, \dots, \mathbb{P}^{[N]}\}$ of the problem columns. By partitioning we mean that $\mathbb{H} = \cup_{i=1}^N \mathbb{P}^{[i]}$ and $\mathbb{P}^{[i]} \cap \mathbb{P}^{[j]} = \emptyset$. We assume that agent i initially knows the column set $\mathbb{P}^{[i]}$. Throughout the paper we use the notational convention that superscript $[i]$ denotes a quantity belonging to agent i. Note that in the assignment problem (1) there is a one-to-one correspondence between a column and an edge of the assignment graph $\mathcal{G}_a$ and that each agent initially knows a subset of these edges, i.e., those corresponding to its own assignments.

Communication Network Model: We model the communication between the N agents by a directed graph (digraph) $\mathcal{G}_c = (\{1, \dots, N\}, E_c)$, named *communication graph*. The node set $\{1, \dots, N\}$ represents the agents, and the edge set $E_c \subset \{1, \dots, N\}^2$ characterizes the communication between the agents. We consider in particular time-dependent digraphs of the form $\mathcal{G}_c(t) = (\{1, \dots, N\}, E(t))$, where $t \in \mathbb{R}_{\geq 0}$ represents the universal time. A graph $\mathcal{G}_c(t)$ models the communication in the sense that at time t there is an edge from node i to node j if and only if agent i transmits information to agent j at time t. The time-varying set of outgoing (incoming) *neighbors* of node i at time t, denoted by $\mathcal{N}_O(i,t)$ ($\mathcal{N}_I(i,t)$), are the set of nodes to (from) which there are edges from (to) i at time t. A static digraph is said to be *strongly connected* if, for every pair of nodes (i, j), there exists a path of directed edges that goes from i to j. For the time-varying communication graph, we rely on the notion of a uniformly jointly strongly connected graph.

Assumption 3.1 (Uniform Joint Strong Connectivity) *There exists a positive and bounded duration T_c, such that for every time instant $t \in \mathbb{R}_{\geq 0}$, the digraph $\mathcal{G}_c^{t+T_c}(t) := \cup_{\tau=t}^{T_c} \mathcal{G}_c(\tau)$ is strongly connected.* □

In a static directed graph, the minimum number of edges between node i and j is called the *distance* from i to j and is denoted by $\text{dist}(i,j)$. The maximum $\text{dist}(i,j)$ taken over all pairs (i,j) is the *diameter* of the graph $\mathcal{G}_c$ and is denoted by $\text{diam}(\mathcal{G}_c)$.

A first contribution of this paper is the formal definition of distributed linear programs.

Definition 3.2 (Distributed Linear Program) *A distributed linear program is a tuple $(\mathcal{G}_c, (\mathbb{H}, b), \mathcal{P})$ that consists of: (i) a time-varying communication graph $\mathcal{G}_c(t) = (\{1,\dots,N\}, E_c(t))$; (ii) a linear program $(\mathbb{H}, b)$; (iii) a unique partitioning $\mathcal{P} = \{\mathbb{P}^{[1]},\dots,\mathbb{P}^{[N]}\}$ of the problem columns. A solution of the distributed linear program is attained when all agents have agreed on the same optimal solution solving $(\mathbb{H}, b)$.* □

We develop in this paper a distributed algorithm which solves this class of problems. A distributed algorithm can be abstractly described as follows. It consists of: (i) the set W, called the set of *states* $w^{[i]}(t)$, (ii) the set Σ, called the *communication alphabet* including the null element, (iii) the map $\texttt{MSG} : W \times \{1,\dots,N\} \to \Sigma$, called *message function*, and (iv) the map $\texttt{STF} : W \times \Sigma^{|\mathcal{N}_I(i,t)|} \to W$, called the *state transition function*. The algorithm starts at $t = 0$ and each agent initializes its state to $w^{[i]}(0)$. Every agent i performs two actions repeatedly: (i) it sends to each of its outgoing neighbors $j \in \mathcal{N}_O(i,t)$ a message computed as $\texttt{MSG}(w^{[i]}(t), j)$; (ii) whenever it receives information from its in-neighbors $\mathcal{N}_I(i,t)$, it updates its state $w^{[i]}(t)$ according to the state transition function. Each agent performs these two actions at its own speed and no synchronization is required in the network.

4 Centralized Solution of Degenerate Linear Programs

We discuss in the following a non-standard algorithm, proposed in [14], which solves degenerate linear programs of the form (2) in a centralized framework without network constraints.

An important concept in linear programming is the one of a *basic solution*. A set of exactly d columns $\mathbb{B} \subseteq \mathbb{H}$ is called a *basis* if $\text{rank}(A_B) = d$. If a linear program $(\mathbb{H}, b)$ has an optimal feasible solution, it has an optimal basic feasible solution, i.e., there exists an optimal vector x^* composed of d *basic variables* $x_B \neq 0$, and $n - d$ *non-basic variables* $x_N = 0$. Note that for linear programs with $d \ll n$ the dimension of the non-zero components of the solution vector is significantly smaller than the dimension of the full decision vector. Two bases $\mathbb{B}_1$ and $\mathbb{B}_2$ are said to be *adjacent*, if there exist columns $e \in \mathbb{B}_2$ and $\ell \in \mathbb{B}_1$ such that $\mathbb{B}_2 = \{\mathbb{B}_1 \cup e\} \setminus \{\ell\}$. For each feasible basis there is a value called the *primal cost* $z_B = c_B^T x_B$. Given a basis $\mathbb{B}$ and a single non-basic column $h \notin \mathbb{B}$, the *reduced cost* of the column is

$$\bar{c}_h = c_h - A_h^T (A_B^{-1})^T c_B \;=:\; r_{\{B \cup h\}}^T c_{\{B \cup h\}}, \tag{4}$$

where $c_h \in \mathbb{R}$ and $A_h \in \mathbb{R}^d$ refer to the problem data related to column h. The reduced cost gives rise to the standard optimality condition [15]: Given a basis $\mathbb{B}$ and the corresponding primal feasible solution $x_B = A_B^{-1} b$, if $\bar{c}_h \geq 0$, for all columns $h \in \mathbb{H}$, then x_B is an optimal solution.

A major challenge in linear programming is the non-uniqueness of optimal solutions, called degeneracy. Commonly,

two types of degeneracy are distinguished. A linear program is said to be *primal degenerate* if there are more than one basis that lead to the same optimal primal solution. It is said to be *dual degenerate* if there are more than one primal optimal solutions. That is, there exist several bases, say $\mathbb{B}_i$ and $\mathbb{B}_j$, providing different basic solutions, $x_{B_i} \neq x_{B_j}$ but have the same optimal primal cost $z_{B_i} = z_{B_j} = z^*$. A problem is said to be fully non-degenerate, if it is neither primal nor dual degenerate. The following holds for non-degenerate linear programs.

Lemma 4.1 (Solution Uniqueness [11]) *Every fully non-degenerate linear program has at most one optimal solution.* □

Theorem 4.2 (Solution Reachability [11]) *If a linear program has an optimal solution and is fully non-degenerate, then there exists a sequence of adjacent bases from any basis $\mathbb{B}$ to the unique optimal basis $\mathbb{B}^*$.* □

However, as we have illustrated with the multi-agent assignment, degeneracy is a very common phenomenon and requires special attention.

A well established procedure for solving linear programs is the simplex algorithm, which is informally described as follows: *Let a primal feasible basis $\mathbb{B}$ be given. While there exists an entering column $e \notin \mathbb{B}$ such that $\bar{c}_e < 0$, find a leaving basic column $\ell(e) \in \mathbb{B}$ such that $(\mathbb{B} \cup \{e\}) \setminus \{\ell\}$ is again a feasible basis. Exchange the column ℓ with e to get a new basis.* The procedure of replacing a basic column with a non-basic one is called *pivot*.
In order to handle degeneracy, a variant of the simplex algorithm is presented in the following. We use the algorithm proposed in [14], building on results in [11]. The algorithm relies on the lexicographic ordering of vectors.

Definition 4.3 (Lex-positivity) *A vector $\gamma = (\gamma_1, \ldots, \gamma_r)$ is said to be* lexico-positive *(or lex-positive) if $\gamma \neq 0$ and the first non-zero component of γ is positive.* □

Lexico-positivity will be denoted by the symbol $\gamma \succ 0$. We write $\gamma \succeq 0$ if γ is either lex-positive or the zero vector. The notion $\gamma \prec 0$ denotes that $-\gamma \succ 0$. For two arbitrary vectors v and u, we say that $v \succ u$ ($v \succeq u$) if $v - u \succ 0$ ($v - u \succeq 0$). Given a set of vectors $\{v_1, \ldots, v_r\}$, the lexicographical minimum, denoted *lexmin*, is the element v_i, such that $v_j \succeq v_i$ for all $j \neq i$. We use lexsort$\{v_1, \ldots, v_r\}$ to refer to the lexicographically sorted set of the vectors $\{v_1, \ldots, v_r\}$. Note that for any list of pairwise distinct vectors *lexsort* provides a unique ordering of the vectors. The concept of lex-positivity gives rise to a refinement of the notion of a feasible basis.

Definition 4.4 (Lex-feasibility) *A feasible basis $\mathbb{B}$ is called* lexicographically feasible *(lex-feasible) if every row of the matrix $[A_B^{-1}b,\ A_B^{-1}]$ is lex-positive.* □

Starting with a lex-feasible basis, the simplex method can be refined such that after a pivot iteration the new basis is again primal lex-feasible, see [14]. Let $\mathbb{B}$ be a lex-feasible basis and e the entering column, let the leaving column be chosen by the *lexicographic ratio test*,

$$\ell_{\text{lex}}(e) = \arg \operatorname*{lexmin}_{\text{j}\in\mathbb{B}} \{[A_B^{-1}b,\ A_B^{-1}]_{\bullet j} / (A_B^{-1}A_e)_{\bullet j} |\ (A_B^{-1}A_e)_{\bullet j} > 0\}, \tag{5}$$

where the subscript $\bullet j$ denotes selection of the j-th row of the matrix, respectively vector, then the next basis is again lex-feasible. Such a selection rule prevents the simplex algorithm from cycling. To guarantee convergence to a unique solution, we change the optimization criterion to the lexicographically perturbed cost

$$\phi(x) = c^T x + \delta_0 x_1 + \delta_0^2 x_2 + \ldots + \delta_0^n x_n, \tag{6}$$

where x_i represents the i-th component of the vector x. Note that for a sufficiently small constant δ_0, the optimizer x^*=arg $\min_x \phi(x)$ corresponds to the unique lexicographically minimal solution of (2). Let now $\delta = [\delta_0, \delta_0^2, \ldots, \delta_0^n]^T$ and write $\phi(x) = (c^T + \delta^T)x$. A column becomes admissible with respect to the perturbed cost ϕ if the reduced cost is negative, i.e., $\bar{c}_e = r^T_{\{B\cup e\}} c_{\{B\cup e\}} + r^T_{\{B\cup e\}} \delta_{\{B\cup e\}} < 0$, which is equivalent to the analytic condition, [14],

$$[r^T_{\{B\cup e\}} c_{\{B\cup e\}}, r^T_{\{B\cup e\}}] \prec 0. \tag{7}$$

Replacing the original definition of the reduced cost with (7) provides a mean to minimize the lexicographically perturbed cost (6) without having small numbers involved in the computation.

Algorithms 1 and 2 give a pseudo code description of the modified `Pivot` and `Simplex` algorithms. The basic algorithm is here slightly extended to handle unbounded problems. If at some `Pivot` iteration (5) returns no leaving column, then the problem is unbounded. We then set the new basis to `null`, and stop the execution of the algorithm. This version of the `Simplex` is proposed in [14] and determines the unique lexicographically optimal solution.

Algorithm 1 `Pivot` $(\mathbb{B}, e)$

Require: A lex-feasible basis $\mathbb{B}$, a non-basic column $e \notin \mathbb{B}$
  if $[r^T_{\{B\cup e\}} c_{\{B\cup e\}},\ r^T_{\{B\cup e\}}] \prec 0$ **then**
    select the leaving column $\ell_{\text{lex}}(e)$ via *lex ratio test* (5)
    if $\ell_{\text{lex}}(e) \neq \emptyset$ **then**
      $\mathbb{B} \leftarrow (\mathbb{B} \cup \{e\}) \setminus \{\ell_{\text{lex}}(e)\}$ *% make the pivot*
    else
      $\mathbb{B} \leftarrow$ `null` *% problem is unbounded*
    end if
  end if
  Return $\mathbb{B}$

Algorithm 2 `Simplex` $(\mathbb{H}, \mathbb{B})$

Require: A set of columns $\mathbb{H}$, a lex-feasible basis $\mathbb{B} \subseteq \mathbb{H}$
  while $\exists e \in \mathbb{H}$ such that $[\mathrm{r}^T_{\{B\cup e\}} c_{\{B\cup e\}}, \mathrm{r}^T_{\{B\cup e\}}] \prec 0$ **do**
    $\mathbb{B} \leftarrow$ `Pivot` $(\mathbb{B}, e)$
  end while

The following remark points out a feature of the algorithm which will need a special care in the distributed setup.

Remark 4.5 *The perturbation δ used in* (6) *and* (7) *requires a fixed and known ordering of the decision variables and the corresponding columns. For a centralized algorithm working with a classical simplex tableau, such an ordering is naturally given. However, this requirement is not naturally satisfied in the distributed settings.* □

5 The Distributed Simplex Algorithm

The main contribution of this paper is the presentation of the Distributed Simplex algorithm, to solve distributed linear programs within multi-agent systems.

5.1 Distributed Simplex Definition

Using the abstract description for distributed algorithms presented in Section 3, the Distributed Simplex algorithm is informally as follows.

The *state of each agent* is a lex-feasible basis, $w^{[i]}(t) = \mathbb{B}^{[i]}$, i.e., each agent stores and updates a set of columns forming a basis. A set of basis columns suffices to fully determine the solution of a linear program, and thus every agent keeps and updates a copy of a possible solution. A basis is defined by exactly d columns, and therefore the state $w^{[i]}(t) \in \mathbb{R}^{d\times(d+1)}$. For linear programs with $d \ll n$ the basis information required to be stored is reasonably small. To initialize the algorithm, every agent creates an artificial basis using a method known as *big-M* method. Let, without loss of generality, each entry of the vector b be non-negative. Then, each agent constructs artificial decision variables $\hat{x}_1, \ldots, \hat{x}_d$ and generates a corresponding initial basis $\mathbb{B}_M$ as follows: It chooses $A_{B_M} = \mathbb{I}_d$ and $c_{B_M} = M \cdot 1$, where $\mathbb{I}_d$ is the $d \times d$ identity matrix, and 1 is a d-dimensional vector of ones. The cost coefficients are all given the artificial value M, which is larger than any cost coefficient in the original problem ($M \gg \max_{i=1,\ldots,n}(c_i)$), i.e., M can be chosen as the largest number realizable by the processor.

The *message function* `MSG` simply implements the transmission of the current basis $\mathbb{B}^{[i]}$ to the out neighbors. Note that the size of `MSG` is bounded by $d \times (d+1)$ numbers, and that therefore the exchanged data is small for problems with small dimension $d \ll n$.

The *state transition function* $\texttt{STF}(w^{[i]}(t), \cup_{j\in\mathcal{N}_I(i,t)}\texttt{MSG}(w^{[j]}, i))$ performed by each agent to update its basis $\mathbb{B}^{[i]}$ proceeds as follows. Having received columns from its in-neighbors, and having its set of permanent columns $\mathbb{P}^{[i]}$ in its memory, the agent performs two actions:

(i) it sorts all columns in its memory lexicographically, i.e., $\text{lexsort}\{\mathbb{P}^{[i]}, \mathbb{B}^{[i]}, \cup_{j\in\mathcal{N}_I(i,t)}\texttt{MSG}(\mathbb{B}^{[j]}, i)\}$;

(ii) it performs the `Simplex` algorithm (Algorithm 2), and updates its basis with the new, improved basis.

We provide a pseudo code description of the algorithm in the following table.

`Problem data:`	$(\mathcal{G}_c(t), (\mathbb{H}, b), \mathcal{P})$
`Message alphabet:`	$\Sigma = \mathbb{H}^d \cup \{\texttt{null}\}$
`Processor state:`	$\mathbb{B}^{[i]} \subset \mathbb{H}$ with $\text{card}(\mathbb{B}) = d$
`Initialization:`	$\mathbb{B}^{[i]} := \mathbb{B}_M$

`function` $\texttt{MSG}(\mathbb{B}^{[i]}, j)$
  return $\mathbb{B}^{[i]}$

`function` $\texttt{STF}(\mathbb{B}^{[i]}, y)$ *% executed by agent i, with $y_j := \texttt{MSG}(\mathbb{B}^{[j]}, i)$*

if $y_j \neq \texttt{null}$ for all $j \in \mathcal{N}_I(i,t)$ **then**

$\mathbb{H}^{tmp} \leftarrow \text{lexsort}\{\mathbb{P}^{[i]} \cup \mathbb{B}^{[i]} \cup \big(\cup_{j\in\mathcal{N}_I(i,t)}\, y_j\big)\}$

$\mathbb{B}^{[i]} \leftarrow \texttt{Simplex}\,(\mathbb{H}^{tmp}, \mathbb{B}^{[i]})$

else

$\mathbb{B}^{[i]} \leftarrow \texttt{null}$

end if

Two steps of the algorithm deserve particular attention. First, every agent has to evaluate its own permanent columns $\mathbb{P}^{[i]}$ repeatedly. If the permanent columns were not evaluated, the algorithm is not guaranteed to converge to an optimal solution. The convergence of the algorithm will be analyzed in the technical proof later on. Second, each agent needs to perform a lexicographic sorting of the columns in its memory. Note that agents receive columns in an unsorted manner and an ordering of the decision variables and columns required for (6) and (7) is not given in distributed systems. The lexicographic comparison provides a unique ordering of the columns and, correspondingly, the decision variables. In a distributed setup, the lexicographic ordering has the advantage that it can be locally determined by every agent. Thus, the lexicographic ordering of the column data ensures that all agents optimize with respect to the same lexicographic objective (6). Since at each time instant an agent is working with a small subset of the full problem data the computational effort required for this step is small.

5.2 Convergence Analysis

The following theorem summarizes the convergence properties of the Distributed Simplex algorithm.

Theorem 5.1 (Distributed Simplex Correctness) *Consider a distributed linear program $(\mathcal{G}_c(t), (\mathbb{H}, b), \mathcal{P})$ with a uniformly jointly strongly connected network $\mathcal{G}_c(t)$, $t \in \mathbb{R}_{\geq 0}$. Let the agents run the* Distributed Simplex *algorithm. Then there exists a finite time T_f such that*

(i) if the centralized problem $(\mathbb{H}, b)$ has a finite optimal solution, then the candidate bases $\mathbb{B}^{[i]}$ of all agents have converged to the same lex-optimal basis;

(ii) if the centralized problem $(\mathbb{H}, b)$ is unbounded, then all agents have detected unboundedness, in the sense that all bases are the `null` *symbol; and*

(iii) if the centralized problem $(\mathbb{H}, b)$ is infeasible, then all agents can detect infeasibility, in the sense that all bases $\mathbb{B}^{[i]}$ have converged, but still contain artificial columns.

PROOF. Statement (i) is shown first. Here, by assumption the centralized linear program $(\mathbb{H}, b)$ is neither infeasible nor unbounded. Throughout the proof, let $x^{[i]}(t) \in \mathbb{R}^n_{\geq 0}$ denote a primal feasible solution to $(\mathbb{H}, b)$ whose nonzero components are the ones associated to the columns in the basis $\mathbb{B}^{[i]}$ of agent i at time t. A major premise for concluding the convergence is the use of the lexicographically perturbed cost $\phi^{[i]}(x^{[i]}(t)) = c^T x^{[i]}(t) + \delta_0 x_1^{[i]}(t) + \delta_0^2 x_2^{[i]}(t) + \ldots + \delta_0^n x_n^{[i]}(t)$, with δ_0 sufficiently small and the ordering of the decision variables defined according to

the lexicographic ordering of the complete column data set, i.e., lexsort$\{\mathbb{H}\}$. We denote the unique lexicographically minimal primal solution with x^* and correspondingly $\phi^* := \phi(x^*)$. To conclude convergence we consider the function

$$\mathcal{V}(t) := \sum_{i=1}^{N} \phi^{[i]}(x^{[i]}(t)). \tag{8}$$

The minimum of $\mathcal{V}(t)$ is attained if and only if at some time T_f, $\phi^{[i]}(x^{[i]}(T_f)) = \phi^*$ for all $i \in \{1, \ldots, N\}$, which is equivalent to all agents having converged to the unique lexicographic minimizer, i.e., $x^{[i]}(T_f) = x^*$. All agents initialize their local problems with a lex-feasible basis of finite value, i.e., $\phi^{[i]}(x^{[i]}(0)) < \infty$, and the `Pivot` iteration never leads to an increase of $\phi^{[i]}(x^{[i]}(t))$. Thus, $\mathcal{V}(0) \geq \mathcal{V}(t) \geq N\phi^*$. We have to show that $\mathcal{V}(t)$ strictly decreases within a finite time interval, as long as the algorithm is not converged. Since the number of possible bases for a problem $(\mathbb{H}, b)$ is finite, $\phi^{[i]}(x^{[i]}(t))$ can take only a finite number of values, and a strict decrease of $\mathcal{V}(t)$ ensures convergence to the optimal solution in finite time.

As long as the algorithm has not yet converged to the optimal solution there must exist at least one agent, say i, such that $\phi^{[i]}(x^{[i]}(t)) > \phi^*$. As an immediate consequence of Theorem 4.2, there exists at least one column $h \in \mathbb{H}$, such that the cost of the basis `Pivot` $(\mathbb{B}^{[i]}, h)$ is strictly less than the cost of $\mathbb{B}^{[i]}$. We show now that within a finite time at least one of the following two actions happens: (a) an agent with a non-optimal basis improves its basis with a column from its permanent columns $\mathbb{P}^{[i]}$, or (b) an agent in the network receives a column which improves its basis from one of its in-neighbors.

Since the state transition function `STF` is performed regularly by each agent, update (a) happens, if possible at all, in finite time. Our claim is now that if no agent can improve its objective with one of its permanent columns, then (b) happens in finite time. First, let us observe that if all agents have identical objectives $\phi^{[i]}(x^{[i]}(t)) = \phi^{[j]}(x^{[j]}(t)) \neq \phi^*$, then (a) happens. Thus, if (a) does not happen, there is at least one pair of agents, say i and j, with different objective values $\phi^{[i]}(x^{[i]}(t)) < \phi^{[j]}(x^{[j]}(t))$. As long as there are two agents i and j with $\phi^{[i]}(x^{[i]}(t)) < \phi^{[j]}(x^{[j]}(t))$, there must exist at least two agents l and k with $\phi^{[l]}(x^{[l]}(t)) < \phi^{[k]}(x^{[k]}(t))$, such that l is an in-neighbor to k in the (by Assumption 3.1) strongly connected communication graph $\mathcal{G}_c^{T_c}(t) = \cup_{\tau=t}^{t+T_c} \mathcal{G}_c(\tau)$ and the edge (l, k) is an element of the directed path from i to j. This last implication can be shown by the following contradiction argument. Assume no such agent pair l and k exists, i.e. $\phi^{[l]}(x^{[i]}(l)) = \phi^{[k]}(x^{[k]}(t))$, then all agents preceding l in the directed path, including the agent i must have the same objective value so that $\phi^{[i]}(x^{[i]}(t)) = \phi^{[l]}(x^{[l]}(t))$. Additionally all agents succeeding k, including agent j must have the same objective value so that $\phi^{[j]}(x^{[j]}(t)) = \phi^{[k]}(x^{[k]}(t))$, contradicting the basic assumption $\phi^{[i]}(x^{[i]}(t)) < \phi^{[j]}(x^{[j]}(t))$. We conclude that, due to Assumption 3.1, at least after a time interval with maximal length T_c, agent k will receive from agent l a basis $\mathbb{B}^{[l]}$, which has a smaller objective value than $\mathbb{B}^{[k]}$. Thus, within the time interval T_c agent k updates its basis so that $\phi^{[k]}(x^{[k]}(t + T_c)) < \phi^{[k]}(x^{[k]}(t))$ and $\mathcal{V}(t + T_c) < \mathcal{V}(t)$. This concludes the first part of the proof.

Next, we prove part (ii) and (iii) of the theorem. If the original centralized linear program is unbounded, also the modified problem including the artificial columns $\hat{h}$ is unbounded. Then at least one agent will detect unboundedness

in finite time and transmit the `null` symbol to all its out-neighbors. Since the communication graph is uniformly jointly strongly connected, after a finite time T_f the whole network will determine unboundedness.
Finally, if the original problem $(\mathbb{H}, b)$ is infeasible, then there exists no non-negative solution to $Ax = b$. Since $\phi^{[i]}(x^{[i]}(t))$ is not increasing and is bounded, and the number of possible bases is finite, all agents will converge in a finite time T_f to some limit bases. These limit bases must then contain some of the artificial columns generated for the initialization and infeasibility can then be concluded in finite time. □

We provide here, without proof, the stopping criterion for the algorithm proposed in [18] for a synchronous implementation. We say that the algorithm is implemented synchronous, if all agents exchange information and update their state at the same times, with the time elapsing between two communications being T_c.

Theorem 5.2 (Stopping Criterion [18]) *Consider a network of agents communicating according to a static, strongly connected digraph $\mathcal{G}_c$ and implementing* Distributed Simplex *algorithm in a synchronous realization. Each agent can stop the execution of the algorithm if $\mathbb{B}^{[i]}$ has not changed in a time interval of length $(2\,\mathrm{diam}(\mathcal{G}_c) + 1)T_c$.*

5.3 Discussion: Duality to Constraints Consensus *Algorithm*

Distributed linear programs are strongly related to *distributed abstract programs* introduced in [18]. An abstract program is defined as a pair (H, θ), where H is a finite set of *constraints*, and $\theta : 2^H \to \Theta$ is a function taking values in a linearly ordered set $(\Theta, \leq)$. In distributed abstract programs, defined in [18], the constraints are distributed throughout a network of agents. Abstract programs are a generalization of linear programs presented in the dual form $\{\max\ b^T y, \text{ subj. to } A^T y \leq c\}$, see, e.g., [12]. Note that an inequality constraint of the dual formulation corresponds to a column h of the primal problem, and a basis (in the abstract sense) is a set of d inequality constraints. The proposed distributed solution algorithm in [18], the Constraints Consensus, is such that agents exchange constraints which form a local basis. When receiving constraints, an agent updates its local basis based on the constraints received and its original constraints. We now clarify the relation between the Constraints Consensus algorithm and the Distributed Simplex algorithm. We write $\phi^{[i]}(t)$ for the value of the basis of agent i at time t, when applying the Distributed Simplex algorithm, and $\theta^{[i]}(t)$ for the value of the basis (in the abstract sense) for an agent applying the Constraints Consensus algorithm. We speak of the trajectories $\phi^{[i]}$ and $\theta^{[i]}$ to denote the evolution of these quantities during the run-time of the algorithm.

Proposition 5.3 (Duality to Constraints Consensus) *Let $(\mathbb{H}, b)$ be a fully non-degenerate, feasible linear program with a bounded optimal solution. Then the trajectories $\phi^{[i]}$ produced by the* Distributed Simplex *algorithm applied to the primal problem (2) are equivalent to the trajectories $\theta^{[i]}$ produced by the* Constraints Consensus *algorithm, [18], applied to the dual linear program $\{\min\ -b^T y, \text{ subj. to } A^T y \leq c\}$.*

PROOF. Given a subset of columns or constraints, respectively, $\mathbb{G} \subset \mathbb{H}$, by strong duality, it holds that $\phi_G = \{\min\ c_G^T x_G,\ A_G x_G = b, x_G \geq 0\} = \{\min\ -b_G^T y_G,\ A_G^T y_G \leq c\} = \theta_G$. In both algorithms, the "local" solvers

$h_{i\kappa}$:	edge of $\mathcal{G}_a$ connecting agent i and task κ
$\mathbb{H}$:	edges of $\mathcal{G}_a$
b :	vector of ones in $\mathbb{R}^{2N-1}$
$\mathbb{P}^{[i]}$:	edges of $\mathcal{G}_a$ connecting agent i with the tasks
$\mathbb{B}$:	spanning tree of $\mathcal{G}_a$, with $2N-1$ arcs

Table I: The assignment problem data as a distributed linear program.

`Simplex` and Subex_LP (see [18]) solve the subproblems and determine ϕ_G and θ_G. The corresponding bases (in the primal and in the abstract sense) are unique, since the problem is by assumption non-degenerate and Subex_LP is applicable since the problem is feasible and bounded. Thus, given the same information set $\mathbb{G}$, both algorithms compute a basis in the respective sense with the same value. Agents exchange in both algorithms their locally optimal bases with neighbors, and thus the information available to an agent at a specific instant are dual. Thus, the two algorithms realize the same trajectories $\phi_B^{[i]}$ and $\theta_B^{[i]}$ and are dual to each other. □

6 Application of the Distributed Simplex to the Multi-Agent Assignment

We motivated the formulation of distributed linear programs with the multi-agent assignment problem. Table I clarifies now more precisely the relations between the two problem formulations. The assignment problem (1) is always primal degenerate, since a basis consists of $2N-1$ columns, while only N of the corresponding primal variables $x_{i\kappa}$ (representing the optimal assignment) are non-zero. As discussed in Remark 2.1, it is also often dual degenerate. The Distributed Simplex algorithm is therefore perfectly suited to solve the distributed assignment problem. This claim is supported by the fact that only very little data exchange between the agents is required.

Proposition 6.1 (Communication Requirements) *At each communication instant, every agent transmits at most* $\mathcal{O}(N \log_2 N)$ *bytes.*

PROOF. The integers $c_{i\kappa}$ can be encoded with 2 bytes. Then a column can be encoded with $2 + \lceil \frac{1}{4}(\log_2(N)+1) \rceil$ bytes. Thus, at each round at max $(2N-1) \cdot (2 + \lceil \frac{1}{4}(\log_2(N)+1) \rceil)$ bytes are sent out by each agent. □

We use random assignment problems to analyze by simulation the expected time complexity of the Distributed Simplex algorithm. Although our algorithm is shown to work in asynchronous networks, for the clarity of presentation we consider here synchronous communication. The time interval between subsequent information exchanges is called communication round. In the first simulation, shown in Figure 2, we consider random assignment graphs $\mathcal{G}_a$, generated with an edge probability $\frac{100}{N^2}$ and cost coefficients $c_{i\kappa} \in [0, 20]$. We compare two different communication schemes: an undirected cycle graph and random Erdõs-Rènyi graphs. For the latter, we use the probability $p = (1+\epsilon)\frac{\log(N)}{N}$, $(\epsilon > 0)$, for which the resulting graph $\mathcal{G}_c$ is almost surely connected [1]. Note that the diameter of a cycle graph grows with the number of edges, whereas the average diameter of Erdõs-Rènyi graphs is $\log(N)/\log(pN)$. For each configuration ten simulations are performed, and the average number of communication rounds (with the corresponding 95% trust intervals) are shown in Figure 2. For the Erdõs-Rènyi graphs, the number of communication

rounds remains approximately constant as the number of agents grows, whereas for the cycle graph it grows linearly with the number of agents. This can be explained with the fact that the diameter of the cycle graph grows linearly with the number of nodes, whereas the one of the random graph remains constant. The second simulation study,

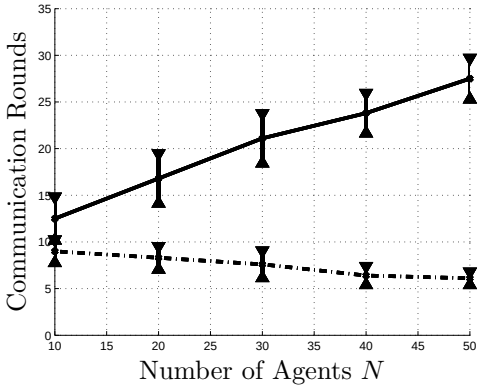

Fig. 2. Average number of communication rounds and the 95% trust interval for random multi-agent assignment problems with communication according to cycle graphs (solid) and Erdõs-Rènyi graphs (dashed).

shown in Figure 3, supports this observation. Here, $\mathcal{G}_a$ is a complete assignment graph with $N = 40$ agents, and the communication graph $\mathcal{G}_c$ is a directed k-regular graph. We vary $k \in \{1, \ldots, 39\}$ and perform for each configuration 15 problems with random cost coefficients. Figure 3 shows the average time complexity and the 95% confidence intervals over the diameter $diam(\mathcal{G}_c) = \lceil (N-1)/k \rceil$. The simulations support the following conjecture on the expected time

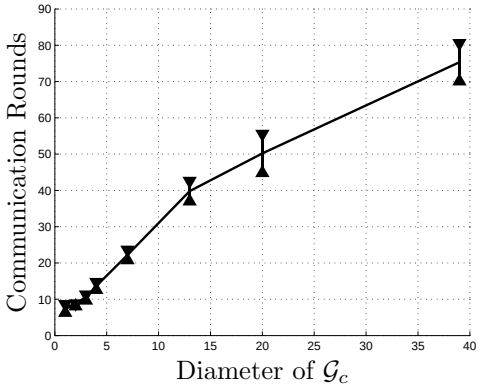

Fig. 3. Average number of communication rounds and the 95% trust interval for the complete $N = 40$ multi-agent assignment problem with communication according to directed k-regular communication graphs.

complexity of the Distributed Simplex algorithm.

Conjecture 6.2 *The average time complexity of the* Distributed Simplex *algorithm belongs, on a fixed communication graph $\mathcal{G}_c$, to* $\mathcal{O}(\mathrm{diam}(\mathcal{G}_c))$. □

7 Conclusions

We have proposed the notion of *Distributed Linear Programs*, as an extension of linear programs to a multi-agent setup. By utilizing a non-standard version of the classical simplex algorithm, a distributed algorithm, named Distributed Simplex has been derived for solving Distributed Linear Programs which are eventually primal and/or dual degenerate. The algorithm is proven to work in asynchronous networks and poses little requirements on the communication structure. The multi-agent assignment problem has been introduced as a relevant problem class for which the algorithm is especially well suited.

References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[2] D. P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14(1-4):105–123, 1989.

[3] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computations: Numerical Methods*. Athena Scientific, Belmont, Massachusetts, 1997.

[5] S. Bolognani and S. Zampieri. Distributed Quasi-Newton method and its application to the optimal reactive power flow problem. In *Proceedings of NECSYS 2010*, Annecy, France, September 2010.

[6] L. Brunet, H.-L. Choi, and J. P. How. Consensus-based auction approaches for decentralized task assignment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008.

[7] R. Burkard. Selected topics on assignment problems. *Discrete and Applied Mathematics*, 123:257–302, 2002.

[8] D. A. Castañón and C. Wu. Distributed algorithms for dynamic reassignment. In *IEEE Conference on Decision and Control*, pages 13–18, Maui, Hawaii, December 2003.

[9] G. Dantzig and P. Wolfe. The decomposition algorithm for linear programs. *Econometrica*, 29(4):767 – 778, 1961.

[10] H. Dutta and H. Kargupta. Distributed linear programming and resource management for data mining in distributed environments. In *IEEE International Conference on Data Mining*, pages 543–552, 2008.

[11] K. Fukuda, H.-J. Lüthi, and M. Namiki. The existence of a short sequence of admissible pivots to an optimal basis in LP and LCP. *International Transactions in Operational Research*, 4(4):273–284, 1997.

[12] B. Gärtner and E. Welzl. *Linear Programming-Randomization and Abstract Frameworks*, volume 1046 of *Lecture Notes in Computer Science*, chapter Symposium on Theoretical Aspects of Computer Science, pages 669–687. Springer, 1996.

[13] J. Hall and K. McKinnon. ASYNPLEX, an asynchronous parallel revised simplex algorithms. *Annals of Operations Research*, 81(24):27–50, 1998.

[14] C. Jones, E. Kerrigan, and J. Maciejowski. Lexicographic perturbation for multiparametric linear programming with applications to control. *Automatica*, 43(10):1808–1816, 2007.

[15] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, 1973.

[16] B. J. Moore and K. M. Passino. Distributed task assignment for mobile agents. *IEEE Transactions on Automatic Control*, 52(4):749–753, 2007.

[17] A. Nedic, A. Ozdaglar, and P. A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[18] G. Notarstefano and F. Bullo. Distributed abstract optimization via constraints consensus: Theory and applications. *IEEE Transactions on Automatic Control*, 56(10):2247–2261, October 2011.

[19] S. L. Smith and F. Bullo. Monotonic target assignment for robotic networks. *IEEE Transactions on Automatic Control*, 54(9):2042–2057, 2009.

[20] G. Yarmish and R. Slyke. A distributed, scalable simplex method. *Journal of Supercomputing*, 49(3):373–381, 2009.

[21] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas. A distributed auction algorithm for the assignment problem. In *IEEE Conference on Decision and Control*, pages 1212–1217, Cancun, Mexico, December 2008.

[22] M. Zhu and S. Martínez. On distributed convex optimization under inequality and equality constraints via primal-dual subgradient methods. *IEEE Transactions on Automatic Control*, 2009. to appear.