A Distributed Multi-Agent Command Governor Strategy for the Coordination of Networked Interconnected Systems

Alessandro Casavola, Emanuele Garone, Francesco Tedesco

Abstract—A novel distributed coordination strategy is presented for networked, locally regulated and possibly dynamically coupled, interconnected systems. Such systems are assumed to be connected via data links and subject to pointwise-in-time global constraints on some relevant variables of them, to be enforced as a coordination goal along the overall system evolutions. Such a coordination-by-constraint paradigm is accomplished by resorting to a novel distributed multi-agent Command Governor (CG) approach where each agent is in charge to locally modify, whenever necessary on the basis of a reduced amount of data exchanged with the other agents of the strategy, the prescribed set-points to the regulated subsystems so that the coordination constraints are always satisfied. The strategy is fully described and its main properties analyzed, especially for what it concerns the stability, feasibility and Pareto optimality of the solution. An liveness analysis of the strategy is also presented for the aspects related to the presence of undesirable Nash equilibria and/or deadlock situations and a discussion on the scalability of the solution with the problem dimension is reported. The constrained coordination of a network of interconnected water tanks is presented in a final example in order to show the effectiveness of the proposed strategy.

I. INTRODUCTION

The problem of interest here is the design of distributed supervision strategies based on Multi-Agent Command Governor (CG) ideas for networked interconnected systems in situations where the use of a centralized coordination unit is impracticable because requiring unrealistic or unavailable communication and/or computational infrastructures. A centralized solution to this problem has been recently proposed in [1], where the standard Feedback CG approach of [2] was extended to deal with networked applications, in the quite general networked context depicted in Fig. 1. There, the master station is in charge of supervising and coordinating the slave

Fig. 1. Centralized master/slave architectures

A. Casavola and F. Tedesco are with the Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi della Calabria, Via Pietro Bucci, Cubo 42-c, Rende (CS), 87036, Italy. E. Garone is with Université Libre de Bruxelles, 50 Av. F.D. Roosevelt, B-1050 Brussels, Belgium {casavola, ftedesco}@deis.unical.it egarone@ulb.ac.be

subsystems via a data network. In particular, \( r_i, g_i, x_i, y_i \) and \( c_i \) represent respectively: the nominal references, the feasible references, the states, performance-related and coordination-related outputs of the slave systems. In such a context, the supervision task can be expressed as the requirement of satisfying some tracking performance, viz. \( y_i \approx r_i \), whereas the coordination task consists of enforcing some pointwise-in-time constraints \( c_i \in C_i \) and/or \( f(c_1, c_2, ..., c_N) \in C \) on each subsystem and/or on the overall network evolutions. To this end, the supervisor (Master Agent) is in charge of modifying the nominal references \( r_i \) into the feasible ones \( g_i \) on the basis of current measures of the subsystem’s states, when the tracking of the nominal references \( r_i \) would produce constraint violations and hence loss of coordination. Examples of applications include the constrained coordination of groups of vehicles, large-scale chemical processes, supply-chain management systems and electrical power grids, just to mention a few. See [1] for further motivations, details and other examples of application.

In this paper we move towards distributed strategies for solving the above coordination task. The starting point are the novel Feed-Forward CG (FF-CG) ideas recently proposed in [11]. The distributed context under investigation is depicted in Figure 2, where the supervisory task is now distributed amongst several agents, which are assumed to be able to communicate amongst them and with the regulated subsystems as well.

A second substantial difference in Figure 2 is that the references modification is undertaken without using an ex-
plicit measure or estimate of the current states of the slave subsystems. In fact, only the commands \( g_i \) are exchanged amongst the computing agents. This difference is noticeable in distributed frameworks and makes the proposed approach innovative with respect to all existing distributed MPC approaches which explicitly make use of a measurement of the current subsystems’ states to compute the commands. See [3] for a comprehensive review of earlier and relatively recent MPC schemes. See also [4], [5] and [6], [7], [8] for more recent non-cooperative and, respectively, cooperative distributed MPC methods. In particular, distributed sequential MPC schemes, more akin to the present approach, have been presented in [8], [9], [10]

Because the above feature is built-in in the centralized FF-CG approach proposed in [11], this makes it a more attractive starting point than [1] for addressing extension towards distributed frameworks. It is also worth remarking that such an extension is far from trivial. On the contrary, it introduces many technical challenges which are here carefully investigated. In this respect, this paper makes clear several theoretical aspects of this novel sequential distributed scheme, only preliminarily introduced and addressed in [12], [13].

In order to address the supervision and coordination problems depicted in Figure 2, a novel sequential Multi-Agent FF-CG scheme is proposed and its properties are fully investigated. The term “sequential” is used here to emphasize the fact that only one agent at a time is allowed to update its command while all others are instructed to keep applying the last computed commands. On-line, when allowed, each agent updates its command by solving a local convex constrained optimization problem which depends on local data and on a small amount of data received only from the last agent who updated its command. Similar approaches have been presented in [9], [10] where a stabilizing problem for decoupled systems with coupling constraints has been solved by exploiting state measurements. A different perspective for sequential non-iterative methods is here investigated. In particular, it is shown how it is possible to conceive and design constrained distributed supervision schemes without an explicit measure of the state. In this respect, it is worth commenting that the above choice is justified by the fact that this paper focuses on supervision problems, where the aim is the set-point modification to fulfill the constraints, and not on control problems where the use of closed-loop state-feedback strategies would be of course mandatory.

This approach, although behaving increasingly slower for an increasing number of agents, is anyway of interest in all situations whereby the coordination problem consists of few and slow set-point adjustments, e.g. in all small/medium-scale situations where the set-points change infrequently or slower than the system dynamics. The main advantages of such a scheme are in its simplicity and in the low communication rates required for its implementation, remarkably lower than other distributed approaches -e.g. those based on consensus mechanisms. Its derivation is also instrumental to build up faster “parallel” versions of the scheme where, whenever possible, all agents update their own commands simultaneously. See [16] for a preliminary study on parallel distributed FF-CG schemes.

Non-cooperative game concepts are exploited to analyze the properties of this strategy. In particular, the standard properties of BIBO stability, Pareto optimality and feasibility of local solutions will be shown to hold under certain liveness conditions. In fact, it will be shown that, unlike centralized schemes, the proposed distributed supervising scheme may fail to converge to a Pareto-optimal solution under specific constraint structures because of the presence of deadlock situations, where the agents might get stuck in an undesirable Nash-equilibrium forever. A method to analyze the geometrical structure of the constrained set is proposed, which is able to establish if deadlock situations may occur. Such analysis consists of the verification of a particular geometrical property, here referred to as Constraint Qualification (CQ), for a finite number of points belonging to the boundaries of the constrained region. Finally it is shown that, for the case of each agent acting on a scalar reference, it is possible to compute arbitrarily accurate inner approximation of the constrained regions that is Constraint Qualified (CD) by construction and avoid from the outset the presence of deadlock situations with an arbitrarily small loss of performance.

The paper is organized as follows: some preliminary notations and definitions are introduced in Sect. II. System description and problem formulation are reported in Sect. II. In Sect. III, the centralized Feed-Forward Command Governor approach is briefly recalled, being it the starting point of the discussion on the novel distributed SS-CG approach presented in Sect. IV. Sect. V introduces the liveness analysis where it is shown that an additional assumption on the constraints structure is necessary for avoiding the presence of deadlock situations and, in turn, the Pareto-Optimality of the solutions in steady-state conditions. Sect. VI provides the main results on the stability, feasibility and liveness of the scheme. Furthermore a faster implementation of the proposed approach is briefly presented. Sect. VII and Sect. VIII conclude the liveness analysis providing respectively a computable way to check if the constrained region satisfies the required conditions to avoid deadlock situations and a geometrical method to compute arbitrarily accurate inner approximations of the constrained region, which satisfy that CQ conditions by construction in coordination problems where each agent manipulates a scalar reference. Finally, Sect. IX presents a complete coordination example on a network of interconnected cascaded water tanks with a detailed scalability and performance analysis and Sect. X ends the paper with some conclusions.

II. NOTATIONS AND DEFINITIONS

\( \mathbb{R}, \mathbb{R}_+ \) and \( \mathbb{Z}_+ \) denote respectively the real, non-negative real and non-negative integer numbers. The Euclidean norm of a vector \( x \in \mathbb{R}^n \) is denoted by \( \|x\| = \sqrt{x_1^2 + ... + x_n^2} \) whereas \( \|x\|_2^2, \Psi = \Psi^T > 0 \), denotes the quadratic form \( x^T \Psi x \). For given sets \( A, \mathcal{E} \subset \mathbb{R}^n \), \( A \sim \mathcal{E} := \{ a : a + e \in A, \forall e \in \mathcal{E} \} \) is the Pontryagin Set Difference.

**Definition 2.1:** (Admissible direction) - Let \( S \subset \mathbb{R}^m \) be a convex set and consider an arbitrary point \( g \in S \). The vector \( v \in \mathbb{R}^m \) represents an admissible direction for \( g \in S \) if there
exists a positive scalar $\bar{\lambda} > 0$ such that $(g + \lambda v) \in S, \lambda \in [0, \bar{\lambda}]$.

**Definition 2.2:** (Box in $\mathbb{R}^m$) A box is a convex polytope with all the hyperplanes characterizing its boundaries being parallel to the axes. More formally a box $\Omega(l, u)$ is defined as

$$\Omega(l, u) := \{ x \in \mathbb{R}^m : l \leq x \leq u \},$$

where $l$ and $u$ are real vectors of $\mathbb{R}^m$ and the inequalities hold componentwise. The minimum side of a box $\Omega$ is denoted as $\lambda_\Omega = \min_{i=1, \ldots, N} u_i - l_i$.

$x \mod y$ indicates the remainder of the integer division $x/y$ and $\text{co}\{S\}$ the Convex Hull of set $S$.

### III. System Description and Problem Formulation

Consider a set of $N$ subsystems $\mathcal{A} = \{1, \ldots, N\}$, each one of which being a LTI closed-loop dynamical system regulated by a local controller which ensures stability and good closed-loop properties when the constraints are not active (small-signal regimes when the coordination is effective). Let the closed-loop subsystem be described by the following discrete-time model

$$\begin{align*}
x_i(t+1) &= \Phi_{ii}x_i(t) + G_i g_i(t) + \sum_{j \in \mathcal{A}\setminus\{i\}} \Phi_{ij}x_j(t) \\
g_i(t) &= H_i^y x_i(t) \\
c_i(t) &= H_i^c x_i(t) + L_i g_i(t)
\end{align*}$$

(1)

where: $t \in \mathbb{Z}_+, x_i \in \mathbb{R}^{n_i}$ is the state vector (which includes the controller states in the case of dynamic regulation), $g_i \in \mathbb{R}^m$ the manipulable reference vector which, if no constraints (and no CG) were present, would coincide with the desired reference $r_i \in \mathbb{R}^m$, and $y_i \in \mathbb{R}^{m_i}$ is the output vector which is required to track $r_i$. Finally, $c_i \in \mathbb{R}^{n_i}$ represents the local constrained vector which has to fulfill the set-membership constraint

$$c_i(t) \in C_i, \forall t \in \mathbb{Z}_+,$$

(2)

$C_i$ being a convex and compact polytopic set. It is worth pointing out that, in order to possibly characterize global (coupling) constraints amongst states of different subsystems, the vector $c_i$ in (1) is allowed to depend on the aggregate state and manipulable reference vectors $x = [x_1^T, \ldots, x_N^T]^T \in \mathbb{R}^n$, with $n = \sum_{i=1}^N n_i$, and $y = [y_1^T, \ldots, y_N^T]^T \in \mathbb{R}^{m}$, with $m = \sum_{i=1}^N m_i$. Moreover, we denote by $r = [r_1^T, \ldots, r_N^T]^T \in \mathbb{R}^m$ and $c = [c_1^T, \ldots, c_N^T]^T \in \mathbb{R}^{n}$, with $n = \sum_{i=1}^N n_i$, the other relevant aggregate vectors. The overall system arising by the composition of the above $N$ subsystems can be described as

$$\begin{align*}
x(t+1) &= \Phi x(t) + Gg(t) \\
y(t) &= H^y x(t) \\
c(t) &= H^c x(t) + Lg(t)
\end{align*}$$

(3)

Hereafter, it is assumed that

**A1.** $\Phi$ is a Schur matrix.

From a distributed perspective, the above assumption implies, according to Figure 2, that the local regulators to each subsystem (1) are capable to asymptotically stabilize the overall system (3). There is a number of works dealing with the problem of design such kind of distributed/decentralized regulators, see e.g. [18], [19], [20].

The CG design problem consists of locally determining, at each time step $t$ and for each agent $i \in \mathcal{A}$, a suitable reference signal $g_i(t)$ which is the best approximation of $r_i(t)$ and such that its application in the place of $r_i(t)$ does not produce constraint violation, i.e. $c_i(t) \in C_i, \forall t \in \mathbb{Z}_+, i \in \mathcal{A}$.

Classical centralized solutions to the above stated CG design problem (see [2]) have been achieved by finding, at each time $t$, a CG action $g(t)$ as a function of the current reference $r(t)$ and measured state $x(t)$

$$g(t) := g(r(t), x(t))$$

(4)

where $g(t)$ is the best approximation of $r(t)$ under the condition $c(t) \in C, C \subseteq \{C_1 \times \ldots \times C_N\}$ being the global admissible region. In [14], the Feed-Forward CG (FF-CG) approach has been proposed, where a CG action having the following structure

$$g(t) = g(r(t), g(t-\tau), \rho(t-\tau))$$

(5)

was proved to have similar properties of the standard Feedback CG approach (4) when the action is computed every $\tau$ steps and kept constant between two subsequent updating. Notice in particular that the FF-CG action (5) is computed without using any knowledge of the state vector $x(t)$. However, in order to increase the performance under the transients, the scalar $\rho(t-\tau)$ is used, whose rationale will be fully clarified in the next section, to characterize the amount of allowed displacement between the current state $x(t)$ and the equilibrium state $x_{g(t-\tau)} := (I_n - \Phi)^{-1} Gg(t-\tau)$, where the state $x(t)$ would converge under the application of a constant set-point $g(t-\tau)$ for a sufficient number of steps. The main idea of the FF-CG approach is that, because of A1, the state $x(t)$ (and the c-constrained vector $c(t)$ as well) would not differ too much from the closed-loop steady-state equilibrium $x_{g(t-\tau)}$ if the CG action $g(\cdot)$ were changed "slowing enough" with respect to the system dynamics.

### IV. The Feed-Forward CG Approach

In this section we recall the basic ideas and notation of the FF-CG approach proposed in [11], which will be relevant for the forthcoming discussion. Let define, for a given $\delta > 0$, the sets:

$$C_\delta := C \sim B_\delta, \quad \mathcal{W}_\delta := \{ g \in \mathbb{R}^m : c_g \in C_\delta \}$$

(6)

where $B_\delta$ is the ball of radius $\delta$ centered at the origin. The set $\mathcal{W}_\delta$, which we assume non-empty, is the convex and closed set of all constant commands $g$ whose corresponding equilibrium points $c_g := H^c(I_n - \Phi)^{-1} Gg + Lg$ satisfy the constraints...
with margin $\delta$. Let introduce also the virtual evolution of the $c$-variable
\[
\dot{c}(k,x(t),g(t)) := H^c \Phi^k x(t) + \sum_{i=0}^{k-1} \Phi^{k-i-1} g(t) + I(t)
\]
along the virtual time $k$, from the initial condition $x(t)$ at time $k = 0$ under the application of a constant command $g(t), \forall k$.

The virtual $c$-variable evolution (7) can be rewritten as the sum of a steady-state component represented by $c_{g(t)}$ and of the transient evolution $H^c \Phi^k (x(t) - x_{g(t)}):$
\[
\dot{c}(k,x(t),g(t)) = c_{g(t)} + H^c \Phi^k (x(t) - x_{g(t)}).
\]
Because $g(t) \in \mathcal{W}_\delta$ and, in turn, $c_{g(t)} \in \mathcal{C}_\delta$ at each time $t$, a sufficient condition to ensure that the constraints are satisfied, although in a quite arbitrary and conservative way, is to ensure that the transient component is confined into a ball of radius $\rho_{g(t)}$
\[
\|H^c \Phi^k (x(t) - x_{g(t)})\| \leq \rho_{g(t)}, \forall k \geq 0
\]
where $\rho_{g(t)}$ represents the minimum distance between $c_{g(t)}$ and the border of $\mathcal{C}$. $\rho_g := \arg \max_{\rho} \rho$ subject to $B_{\rho}(c_g) \subseteq \mathcal{C}$. (10)
and $B_{\rho}(c_g)$ the ball of radius $\rho$ centered in $c_g$. Details on the computation of $\rho_{g(t)}$ can be found in [11].

Then, the FF-CG design problem translates into the problem of defining an algorithm that is able to select, at each time $t$, a reference value $g(t)$ such that (9) holds true for all $k \geq 0$. This has achieved in [11] by selecting a suitable integer $\tau$, referred to as a Generalized Settling Time, and a sequence of positive scalars $\rho(t)$ such that the following condition, stricter than (9), is satisfied at each time $t$.
\[
\|H^c \Phi^k (x(t) - x_{g(t)})\| \leq \rho(t), \forall k \geq 0
\]
If condition (11) were true at time $t - \tau$ and if a command $g(t-\tau)$ were constantly applied to the system, then the transient contribution from $t$ onwards could be bounded as follows
\[
\|H^c \Phi^k (x(t) - x_{g(t-\tau)})\| \leq \gamma \rho(t - \tau) \leq \gamma \rho_{g(t - \tau)}, \forall k \geq 0
\]
with $\gamma < 1$ because of asymptotical stability. Then, if the FF-CG action were computed every $\tau$ sampling steps and kept constant between two successive updating, at time $t$ our goal would be the one of selecting a new command $g(t)$ such that
\[
\|H^c \Phi^k (x(t) - x_{g(t)})\| \leq \rho(t), \forall k \geq 0
\]
be satisfied for some $\rho(t) > 0$. By introducing the $\tau$-step ahead incremental reference $\Delta g(t) := g(t) - g(t - \tau)$ and state $x_{\Delta g(t)} := x_{g(t)} - x_{g(t-\tau)}$, a sufficient condition for (13) to hold true is
\[
\|H^c \Phi^k x_{\Delta g(t)}\| \leq \rho_{g(t - \tau) + \Delta g(t) - \gamma \rho(t - \tau)}, \forall k \geq 0. \tag{14}
\]
Please note that the latter inequalities always hold true for $\Delta g$ in a sufficiently small ball centered in $\Delta g = 0$. Finally, the Feed-Forward CG selection algorithm can be written as follows

The FF-CG Algorithm

1.1. IF ($t = \kappa \tau, \kappa = 1,2 \ldots$)

1.1.1. SOLVE

\[
g(t) = \arg \min_{g \in \mathcal{W}} \| g - r(t) \|_2^2 \tag{15}
\]

SUBJECT TO \[ (g - g(t - \tau)) \in \Delta g(t) \]

1.2. ELSE $g(t) = g(t - 1)$

1.2.1 APPLY $g(t)$

3.1 UPDATE $\rho(t) = \gamma \rho(t - \tau) + \max_{k \geq 0} \| H^c \Phi^k (I - \Phi)^{-1} \Delta g(t) \|$ (16)

where

\[
\Delta g(t,\rho) = \{ \Delta g : \| H^c \Phi^k (I - \Phi)^{-1} \Delta g \| \leq \rho_{t+\Delta g-\gamma \rho,\forall k \geq 0} \}
\]
is the set of all possible $\tau$-step incremental commands $\Delta g$ which ensure (14) to hold true. It is worth noting that the sets $\mathcal{W}_\delta, \Delta g(t,\rho)$ and the generalized settling time $\tau$ can be computed from the outset as indicated in [11]. Then, the following properties can be proved:

Theorem 1: [14] - Let assumption A1 be fulfilled. Consider the system (3) along with the FF-CG selection rule and let an admissible command signal $g(0) \in \mathcal{W}_\delta$ be applied at $t = 0$ such that (9) holds true. Then:

1) the minimizer of (15), computed every $\tau$ steps, uniquely exists and can be obtained by solving a convex constrained optimization problem;
2) the set $\Delta g(t,\rho)$ is convex, closed, finitely determinable and nonempty. In particular, there exists a scalar $\eta^* > 0$ such that the set $\Delta g(t,\rho)$ contains a ball of radius $\eta^*$, i.e. $\exists \eta^* > 0 : \Delta g(t,\rho) \supseteq B_{\eta^*}, \forall g \in \mathcal{W}_\delta, \forall \rho \leq \rho_g, \rho \geq 0$;
3) the constraints are fulfilled for all $t \in \mathbb{Z}_+$;
4) the overall system is BIBO stable and whenever $r(t) = r$, the sequence of $g(t)$ converges in finite time either to $r$ or to its best steady-state admissible approximation: $g(t) \rightarrow \hat{r} := \arg \min_{g \in \mathcal{W}_\delta} \| g - r \|_2^2$.

\[ \square \]

V. THE DISTRIBUTED SEQUENTIAL FF-CG (S-FFCG) APPROACH

The goal of this section is to introduce the proposed sequential distributed FF-CG scheme. For simplicity, in the forthcoming analysis a simplified variant of the above described FF-CG method will be considered by setting $\rho(t) = \rho_g(t), \forall t > 0$. In this way, the computation of $g(t)$ depends on the previous applied $g(t - \tau)$ and current reference $r(t)$ only. Thus, point 3.1 of the FF-CG Algorithm can be skipped and the set $\Delta g(t,\rho)$ depends only on the current command $g$, that is
\[
\Delta g(g) = \{ \Delta g : \| H^c \Phi^k (I - \Phi)^{-1} \Delta g \| \leq \rho_{g+\Delta g-\gamma \rho,\forall k \geq 0} \}
\]
where $\rho_g$ is determined by means of (10). The above simplification, although conservative, leads to a simpler analysis and allows to highlight what are the true issues of the distributed coordination. In this work we assume that all agents are connected via a communication network. Such a network may be modeled by a communication graph defined as follows:

Definition 5.1: (Communication Graph) Let a set of $N$ subsystems be given. Then, a Communication Graph is an undirected graph $\Gamma = (A, \mathcal{E})$, where
where
\[ \Delta G_i^0(g) := \{ \Delta g_i : [0_{T_{m1}}^T, 0_{T_{m2}}^T, \ldots, \Delta g_i^T, \ldots, 0_{T_{mN}}^T] \in \Delta G(g) \} \]

is the set of all possible command variations for \( g_i \) when the commands of all other agents are frozen. The condition \((\kappa \mod N) == i \) in step 1.1, resembling a C instruction, is a notational expedient to indicate when in the algorithm each agent establishes if its turn to became the updating agent in the sequential procedure has come.

VI. MAIN PROPERTIES OF THE S-FFCG SCHEME

In this Section the main properties of the S-FFCG scheme will be described. The next definitions are instrumental to characterize situations of deadlocks that, unlike the centralized solution, may arise.

The rationale is that by acting one agent at a time, certain viable paths existing in the centralized scheme when the solutions are computed simultaneously, are precluded and the agents could get stuck indefinitely in certain Nash Equilibria without converging to a Pareto-Optimal solution. For the sake of clarity, the notions of Nash Equilibrium and Pareto Optimality are first recalled. Then, important notions and assumptions are introduced that are instrumental to characterize and understand the properties of the proposed distributed algorithm.

Definition 6.1: (Decision Set of agent \( i \)) - The Decision Set \( \mathcal{V}_i^S(g) \) of the \( i \)-th agent at a point \( g \in S \) is the set of all admissible directions along which such an agent could move, under the assumption that all other agents are maintaining their commands unvaried, in updating its action viz. \( \mathcal{V}_i^S(g) := \{ v \in \mathbb{R}^{m_i} : [v_1^T, \ldots, v_{i-1}^T, v^T, v_{i+1}^T, \ldots, v_N^T]^T \in S \} \) is an admissible direction for \( g \in S \).

Definition 6.2: (Viability property) - A point \( g \in S \) is said to be "viable" if, for any admissible direction \( v = [v_1^T, \ldots, v_N^T]^T \in \mathbb{R}^m \), \( v_i \in \mathbb{R}^{m_i} \) with \( \sum_{i=1}^N m_i = m \), there exists at least one agent \( i \) such that the sub-vector \( v_i \neq 0 \) belong to its Decision Set, i.e. \( v_i \in \mathcal{V}_i^S(g) \).

Definition 6.3: (Pareto Optimal Solution - PO) - Let vectors \( r_i, i = 1, 2, \ldots, N \) be given. Consider the following multi-objective problem:

\[
\min_g \{ \| g_{1} - r_{1} \|_{\Psi_1}, \ldots, \| g_{i} - r_{i} \|_{\Psi_i}, \ldots, \| g_{N} - r_{N} \|_{\Psi_N}^2 \} \quad \text{subject to } g = [g_{1}^T, \ldots, g_{i}^T, \ldots, g_{N}^T]^T \in \mathcal{W}_b \]

(21)

A solution \( g^* \in \mathcal{W}_b \) is a PO solution for (21) if there does not exist any vector \( g \in \mathcal{W}_b \), such that:

\[
\| g_{i} - r_{i} \|_{\Psi_i}^2 \leq \| g_{i}^* - r_{i} \|_{\Psi_i}^2, \forall i \in \{1, \ldots, A\} \quad \text{and} \quad \| g_{j} - r_{j} \|_{\Psi_j}^2 \leq \| g_{j}^* - r_{j} \|_{\Psi_j}^2 \leq \| g_{i}^* - r_{i} \|_{\Psi_i}^2, \text{ for some } j \in A.
\]

Definition 6.4: (Nash Equilibrium - NE) - A solution \( g^n \in \mathcal{W}_b \) is a Nash equilibrium for the optimization problem (21) if, for each agent \( i \in A \), there does not exist any sub-vector \( g_i \) such that \( g_i - g_i^T \notin \mathcal{V}_i^S(g^n) \) satisfying the following inequality

\[
\| g_{i} - r_{i} \|_{\Psi_i}^2 \leq \| g_{i}^* - r_{i} \|_{\Psi_i}^2
\]

(22)
Ideally, one would like that the solutions provided by the above S-FFCG Algorithm converge, after a certain time, to a NE $g_i$, $i = 1, ..., N$ which is also PO for such a set-point. As it will be clear soon, a sufficient condition to ensure the correct behaviour is that for any admissible configuration of $g$, the viability property holds with respect to the set of admissible applicable commands

$$\mathcal{W}_\delta \cap G(g(t - \tau))$$

where $G(g) := \{ g' \in \mathbb{R}^m : (g' - g) \in \Delta G(g) \}$. The following Lemma enlightens how the viability analysis of $\mathcal{W}_\delta \cap G(g(t - \tau))$ can be limited to the viability analysis of the set $\mathcal{W}_\delta$.

**Lemma 1**: Any point $g' \in \mathcal{W}_\delta \cap G(g')$ is viable in $\mathcal{W}_\delta \cap G(g')$ if and only if it is viable in $\mathcal{W}_\delta$.

**Proof** - A direct consequence of item 2 of Theorem 1 is that

$$\mathcal{W}_\delta \cap G(g') \supseteq \mathcal{W}_\delta \cap B^{g'}_{\eta^*}$$

where $B^{g'}_{\eta^*}$ is a ball with radius $\eta^*$ centered at $g'$. Hence, each decision set for $g'$ related to the generic $i$-th agent is representable as $\mathcal{V}_{t}^{\mathcal{W}_\delta \cap B^{g'}_{\eta^*}}(g')$. As a consequence, because inside $B^{g'}_{\eta^*}(g')$ all directions $d \in \mathbb{R}^m$ related to $g'$ are admissible, such a set reduces to

$$\mathcal{V}_{i}^{\mathcal{W}_\delta \cap B^{g'}_{\eta^*}}(g') = \mathcal{V}_{i}^{\mathcal{W}_\delta}(g')$$

which concludes the proof.

Finally, because all internal points of $\mathcal{W}_\delta$ enjoy the viability property (see proof of Lemma 2 in Section VI for details), the only problems could derive from points over the border of $\mathcal{W}_\delta$. As a consequence, the following assumption

**A2** Each point belonging to $\partial(\mathcal{W}_\delta)$ is viable, $\partial(\mathcal{W}_\delta)$ denoting the border of $\mathcal{W}_\delta$

is introduced to ensure that the agents get always a PO solution corresponding to the actual constant set-point. Although apparently limitative, the situation is ameliorated in section VI where a computable way to check in finite time if $\partial(\mathcal{W}_\delta)$ satisfies A2 and a geometrical method for computing suitable inner approximations of $\mathcal{W}_\delta$ satisfying A2 are presented. In order to clarify the matter, next Figures 4-5 depict some different viable and no-viable situations.

### A. Main Result

The following properties can be shown to hold true for the above stated S-FFCG scheme

**Theorem 2**: Let assumption A1 be fulfilled. Consider system (3) as the composition of $N$ subsystems in form (1) along with the distributed S-FFCG selection rule (19) and let an admissible aggregate command signal $g(0) = [g^{T}_1(0), ..., g^{T}_{N}(0)]^T \in \mathcal{W}_\delta$ be applied at $t = 0$ such that (9) holds true. Then

1) for each agent $i \in \mathcal{A}$, at each decision time $t = k\tau$, $k \in \mathbb{Z}_+$, the minimizer of (19) uniquely exists and can be obtained by locally solving a convex constrained optimization problem;

2) the overall system acted by the agents implementing the S-FFCG policy never violates the constraints, i.e. $c(t) \in \mathcal{C}$ for all $t \in \mathbb{Z}_+$;

3) under the further assumption A2 and whenever $r(t) \equiv \left[ r^{T}_{1}, ..., r^{T}_{N} \right]^T, \forall t$, with $r_i$ a constant set-point, the sequence of solutions $g(t) = [g^{T}_1(t), ..., g^{T}_{N}(t)]^T$ converges asymptotically to a Pareto-Optimal stationary (constant) solution of (21), which is given by $r$, whenever $r \in \mathcal{W}_\delta$, or by any other Pareto-Optimal solution $\hat{r} \in \mathcal{W}_\delta$ otherwise.

**Proof**

1) The existence of an admissible solution for each agent at each decision time $k\tau$ can be proved by simply remarking that $g_i(t) = g_i(t - \tau)$, is always an admissible, although not necessarily the optimal, solution for the prescribed problem at time $t$.

2) At each decision time $t = k\tau$, with $k \in \mathbb{Z}_+$, from a centralized point of view, a command $g(k\tau)$ complying with (16) is applied to the overall plant. By construction, the latter implies that the set-valued virtual predictions along the virtual time $i$ defined in (7) satisfy

$$\hat{c}(i, x(k\tau), g(k\tau)) \in \mathcal{C}, \forall i \in \mathbb{Z}_+,$$

Then, the statement is proved by simply noticing that the following inclusion

$$c(t) = \hat{c}(i, x(k\tau), g(k\tau)) \in \mathcal{C},$$

holds true for all time instants $t = k\tau + i$, $i \in \{0, 1, ..., \tau - 1\}$ and by repeating the same argument for all $k \in \mathbb{Z}_+$.

3) The stated property of convergence follows simply because the sequences of solutions $g_i(t)$ makes the sequences of local costs $\|g_i(t) - r_i\|_{\Psi_i}^2$ non increasing for any $i = 1, ..., N$ under constant set-points. In fact, it is not convenient for the agents to modify their actual optimal solutions if the costs cannot be decreased further on. To this end, let $g_i(t)$ be the S-FFCG action for the $i$-th agent at time $t$, solution of the optimization problem (19). As already discussed, $g_i(t)$ is still an admissible, though not necessarily the optimal, solution at time $t + \tau$. Hence, the sequences of costs $\|g_i(t) - r_i\|_{\Psi_i}^2$ are all non increasing, i.e.

$$\| g_i(t + \tau) - r_i \|_{\Psi_i}^2 \leq \| g_i(t) - r_i \|_{\Psi_i}^2$$

(26)
Then, we want to show that any stationary solution, viz. \( g(t) = g(t + 1) \) ∀ \( t \), is PO by proving that only PO solutions are stationary. To this end, let \( g'(t) = [g'_1(t), \ldots, g'_n(t)]^T \) be the actual solution at time \( t = \kappa \tau, \kappa \in \mathbb{Z}_+ \) which is assumed to be not PO. As a consequence, other admissible solutions exist which improve the costs. Supposedly, vectors \( v = [v_1^T, \ldots, v_n^T]^T \in \mathbb{R}^m \) would exist with \( g'(t) + v \) being admissible, such that

\[
\|g'_1(t) + v_1 - r_i\|_{\Psi_i}^2 < \|g'_i(t) - r_i\|_{\Psi_i}^2, \quad i = 1, 2, 3. \tag{27}
\]

happens to hold for all \( i \in \mathcal{A}' := \{ i \in \mathcal{A} : v_i \neq 0 \} \) with some of the above inequalities becoming strict for at least one index \( i \in \mathcal{A}' \). Because of the strict convexity of the norm \( \| \cdot \|_{\Psi_i}^2 \), the following inequality holds for all \( i \in \mathcal{A}' \) and for all \( \alpha \in (0, 1) \)

\[
(1 - \alpha)g'_i + \alpha g'_i(t) + v_i - r_i \leq (1 - \alpha)g'_i(t) - r_i + \alpha g'_i(t) + v_i - r_i = 0. \tag{28}
\]

Therefore, by means of straightforward algebraic manipulations, one arrives to

\[
\|g'_1(t) + \alpha v_i - r_i\|_{\Psi_i}^2 - \|g'_i(t) - r_i\|_{\Psi_i}^2 < 0, \quad \alpha \|g'_i(t) + v_i - r_i\|_{\Psi_i}^2 < \alpha \|g'_i(t) - r_i\|_{\Psi_i}^2. \tag{29}
\]

for all \( \alpha \in (0, 1) \). Because (27), the right-hand term in (29) is always negative. Then, one can state

\[
\|g'_1(t) + \alpha v_i - r_i\|_{\Psi_i}^2 - \|g'_i(t) - r_i\|_{\Psi_i}^2 < 0, \forall \alpha \in (0, 1) \tag{30}
\]

The latter may be interpreted as the fact that if the above admissible direction \( v \) did exist at \( g'_i(t) \), for each agent \( i \in \mathcal{A}' \) it would be strictly convenient to move towards \( g'_i(t) + \alpha v_i \), for a suitable value of \( \alpha \), from its previous solution \( g'_i(t) \).

The last step of the proof is to verify that in this situation, at least one agent is allowed to move from \( g'_i(t) \) along \( v_i \) despite the constraints. To this end, because of A2 and Lemma 1, \( v_i \) belongs to \( \mathcal{V}_i^\alpha(g'(t)) \) for a not empty subset \( \mathcal{A}' \subseteq \mathcal{A} \) of agents. Hence, according to the sequential S-FFCG updating policy, if at time \( t = k\tau \), the index \( (k \mod N) \in \mathcal{A}' \) then, because of (30), the agent \( i' = k \mod N \) will find convenient to move into \( g'_{i'}(t) + \alpha v_{i'}, \alpha \in [0, \bar{\alpha}] \). In fact, because of viability of \( g'(t) \) (see the above definition) \( v_{i'} \in \mathcal{V}_{i'}^\alpha(g'(t)) \) implies that a scalar \( \bar{\alpha} \in (0, 1) \) exists ensuring \( g' + [0, \ldots, \alpha T, \ldots, 0]^T \in \mathcal{W}_N \) for all \( \alpha \in (0, \bar{\alpha}) \), which concludes the proof.

**B. A faster Implementation - Fast S-FFCG**

The above algorithm could be very conservative because the agents need to wait for several time instants before being allowed to update their commands. In order to overcome such a drawback, one can consider a faster implementation where, every \( \tau \) steps the actions of all agents are sequentially evaluated within the same sampling time by following the same approach used in [23]. Nevertheless, although this *modus operandi* leads to a more performing algorithm, it requires more demanding CPU capabilities and low transmission times for large networks in order to guarantee that the entire decisional procedure could be been completed within the prescribed sampling period. The resulting algorithm takes the following form:

**Fast Sequential FF-CG Algorithm (Fast-S-FFCG)**

**AT EACH TIME F1**

1. IF \((t \equiv \kappa \tau, \kappa = 0, 1, \ldots)\)
   1.1 SET \( i = \kappa \mod N \)
   1.2 SET \( \text{cnt} = 0 \)
   1.3 WHILE(\( \text{cnt} \leq N \))

1.3.1 AGENT \( i \) RECEIVES \( g(t - \tau) \) FROM THE PREVIOUS AGENT IN THE CYCLE \( \mathcal{H} \)

1.3.2 AGENT \( i \) SOLVES (19)

1.3.3 AGENT \( i \) APPLIES \( g_i(t) \)

1.3.4 AGENT \( i \) UPDATES \( g(t) = [g_1^T(t - \tau), \ldots, g_n^T(t - \tau)]^T \)

1.3.5 AGENT \( i \) TRANSmits \( g(t) \) TO THE NEXT AGENT IN \( \mathcal{H} \)

1.3.6 SET \( \text{cnt} = + + \)

1.3.7 SET \( i = (i \mod N) + 1 \)

It is worth commenting that the properties stated in Theorem 2 still holds true for this different version of the S-FFCG method.

**VII. Liveness Analysis and Viability Verification**

In this section, some aspects of the viability property A2 will be clarified for the case of polyhedral \( \mathcal{V}_i \). In particular, a numerical procedure is presented for checking that this property is satisfied for all points of the boundaries of a given polyhedral set of constraints. In the following analysis, we refer to a generic convex polyhedron \( \mathcal{S} \subseteq \mathbb{R}^m \) expressed as a set of \( p \) linear inequalities \( a_j^T g - b_j \leq 0, j \in \mathcal{J} := \{1, \ldots, p\} \).

If all points on the boundaries of \( \mathcal{S} \) are viable we say that such
a set is *Constraint Qualified* (CD) and no deadlocks can take place under the stated distributed S-FFCG strategy when $\mathcal{W}_\delta$ coincides with such a $S$.

**Lemma 2:** Let a polyhedral set $S \subset \mathbb{R}^m$ be given and $b \in \mathbb{R}^p$. Consider also a generic point $g' \in S$ and the set $J' := \{ j \in J : a_j^T g' = b_j \}$ representing the set of active constraints, viz. the set of constraints satisfied by equality. Then, $g'$ is viable iff the following test fails.

**Test** - Find, if there exists, a vector $w = [w_1^T, \ldots, w_N^T]^T \in \mathbb{R}^m$ such that $\forall i \in A$, with $w_i \neq 0$, it exists at least one $j \in J'$ satisfying:

$$
\begin{align*}
A(g' + w) & \leq b \\
\{ a_j^T [0^T, \ldots, w_i^T, \ldots, 0_N^T]^T > 0, \end{align*}
$$

(31)

where $a_j^T$ and $b_j$ denote rows of the matrix $A$ and, respectively, entries of vector $b$.

**Proof**

1) **Sufficiency - Test** failure $\Rightarrow$ viability of $g'$: Assume that (31) has no solution. Then, for all $w \in \mathbb{R}^m$ such that $A(g' + w) \leq b$ there exists at least one index $i \in A$, with $w_i \neq 0$, satisfying

$$
a_j^T [0^T, \ldots, w_i^T, \ldots, 0_N^T]^T \leq 0, \forall j \in J' \tag{32}
$$

The latter condition implies that for any admissible displacement $w$, all the active constraints are satisfied for at least $i \in A$, i.e.,

$$
a_j^T (g' + [0^T, \ldots, w_i^T, \ldots, 0_N^T]^T) \leq b_j, \forall j \in J', i \in A \tag{33}
$$

Moreover, because of convexity of $S$, the latter still holds for all $\lambda \in (0, 1)$, that is

$$
a_j^T (g' + [\lambda_0^T, \ldots, \lambda w_i^T, \ldots, 0_N^T]^T) \leq b_j, \forall j \in J', i \in A \tag{34}
$$

Thus, it only remains to see what happens for the constraints $a_j^T g' < b_j$, $\forall j \in J \setminus J'$. Because all of them are strict inequalities, it is always possible to find a $\lambda \in (0, 1)$ such that:

$$
a_j^T (g' + [0^T, \ldots, \lambda w_i^T, \ldots, 0_N^T]^T) \leq b_j, \forall j \in J \setminus J', \forall i \in A \tag{35}
$$

The latter, together with (34), implies that

$$
a_j^T (g' + [0^T, \ldots, \lambda w_i^T, \ldots, 0_N^T]^T) \leq b_j, \forall j \in J, i \in A \tag{36}
$$

or equivalently that $w$ belongs to the decision set $\mathcal{V}_j^S(g')$, $i \in A$, which finally implies viability. Please notice that if $g'$ is an internal point of $S$, i.e. $A g' < b$, for all admissible $w \in \mathbb{R}^m$, the set $J'$ is empty. Hence (35) is sufficient to state that $g'$ is viable in $S$.

2) **Necessity - Test** failure $\Leftarrow$ viability of $g'$: We prove the Necessity of the Lemma by proving that if **Test** succeeds then $g'$ is not viable. The success of **Test** would imply the existence of an admissible displacement $w'$ such that, for all $i \in A$ with $w'_i \neq 0$, there exists at least one index $j \in J'$ such that $a_j^T [0^T, (w'_i)^T, \ldots, 0_N^T]^T > 0$. As a consequence, in $g'$ it would exist an admissible displacement $w'$ such that, for all $i \in A$, $w'_i \neq 0$ at least one constraint $j \in J'$ is violated along the sub-direction $w'_i$

$$
a_j^T (g' + [0^T, \ldots, (w'_i)^T, \ldots, 0_N^T]^T) - b_j > 0
$$

or, equivalently, that $w'$ does not belong to the $i$-th decision set of $g'$, $\mathcal{V}_j^S(g')$, which implies that $g'$ is not viable. $\square$

It is worth pointing out that the direct application of the above conditions for checking the viability of all points of $S$ would give rise to a cumbersome numerical procedure. Hereafter, a few geometrical results are presented to show that the viability of all points of $S$ can be established by only checking a finite number of points of its boundary $S$. This to end, the usual notion of *face* of a polyhedron is recalled.

**Definition (Face of a polyhedron)** - Let a convex polyhedron $S \subset \mathbb{R}^m$ expressed by a set of $p$ linear inequalities $a_j^T g - b_j \leq 0, j \in J := \{1, \ldots, p\}$ be given. Any region $\mathcal{P} := \{g \in S : a_j^T g - b_j = 0, j \in J' \subset J\}$ is said to be a *face* of $S$. Moreover, the quantity $m - |J'|$ represents the order of $\mathcal{P}$. $\square$

Based on the above definition, vertices are 0-order faces, facets are $(m - 1)$-order faces, ridges are $(m - 2)$-order faces and so on. Then, consider this preliminary result

**Lemma 3:** Let a convex polyhedron $S$ and a $(m - k)$-order face $\mathcal{P}$ with $k \in \{1, \ldots, m - 1\}$ be given. Then, $w \in \mathbb{R}^m$ is an admissible direction for all points of the interior of $\mathcal{P}$, say it $In(\mathcal{P})$, iff it is an admissible direction for at least one point $g$ of $In(\mathcal{P})$.

**Proof** - To prove the statement it is enough to show that the admissibility of a direction in $S$ only depends on the active constraints. To this end, consider a generic point $g \in In(\mathcal{P})$. Because it belongs to the interior of $\mathcal{P}$, we have a part of the inequalities that are strictly satisfied and a part that are not, viz.

$$
\begin{align*}
a_j^T g & = b_j, \forall j \in J' \\
a_j^T g & < b_j, \forall j \in J \setminus J'.
\end{align*}
$$

(37)

Next, assume $w \in \mathbb{R}^m$ be an admissible direction at $g \in In(\mathcal{P})$. It is also useful to remind that an admissible direction $w = [v_1^T, \ldots, v_m^T]^T$ at $g$ implies that there exists a $\lambda > 0$ which satisfies

$$
a_j^T (g + \lambda w) - b_j \leq 0, \forall \lambda \in (0, \lambda], \forall j \in J \setminus J'.
$$

(38)

Then, because (37) and (38), one has that

$$
a_j^T w \leq b_j, \forall j \in J \setminus J'.
$$

(39)

Now, consider a different internal point $g' \neq g$. Note also that the condition (37) holds true for $g'$ as well. As a consequence, $w$ in (38) is still admissible at $g'$ because:

1) - for all $j \in J'$, conditions (37) and (39) imply that

$$
a_j^T g' + \lambda w \leq b_j, \lambda > 0;
$$

(40)

2) - for all $j \in J \setminus J'$, the scaling factor $\lambda > 0$ in (40) always exists $\forall w \in \mathbb{R}^m$ regardless of conditions (37) and (39). $\square$

Based on the above Lemma 3, the following result can be stated which allows one to verify in an easy way the CQ of $S$.

**Lemma 4:** Let a convex polyhedron $S$ be given and let $\mathcal{P}_S$ denote the set of all faces of $S$. Then, each point of $S$ is viable, that is the constraint set $S$ is CD, iff for each element $\mathcal{P} \in \mathcal{P}_S$ there exists at least a viable point $g \in In(\mathcal{P})$.

**Proof** - The first observation is that all points belonging to the interior of $S$ are viable because of Lemma 2. Then, it only remains to investigate the viability of points of the faces $\mathcal{P}_S$. In particular, we prove that points on each face of $\mathcal{P}_S$ are
viable by mathematical induction. To this end, we assume the viability of points on 0-order faces (vertices) of $P_S$. Then, it is sufficient to show that all points of a $(m-j)$-order faces are viable provided that all points in the interior of each $(m-j)$-order face are viable and all points on $(m-j-1)$-order faces are as such.

With this aim, consider a $(m-j)$-order face $P$ and assume that all points of $\partial(P)$ are viable, $\partial(P)$ consisting of $(m-j-1)$-order faces only. We only need to investigate the viability of points belonging to $In(P)$ provided that a single viable inner point $g \in In(P)$ exists. As proven in Lemma 3, all points of the interior of each face share the same admissible directions. As a consequence, for all $v = [v^T_1, ..., v^T_N]^T$ admissible at $g$, if $v_i \in V^S_i(g)$ (i.e. $[0_1, ..., v_i, ..., 0_N]$) is admissible at $g$ then $v_i \in V^S_i(g')$ for all $g' \in In(P)$, that means that all points belonging to $P$ are viable iff $g$ is viable.

The above results allow one to introduce the following numerical procedure to check the viability all points belonging to the boundaries of a polyhedron $S$.

**Constraint Qualification (CD) test for polyhedrons $S$**

1.1 COMPUTE $P_S$
1.2 SET $Pl := \emptyset$
1.3 FOR EACH $P \in P_S$
1.3.1 SELECT $g \in In(P)$
1.3.2 APPEND $g$ TO $Pl$
1.4 SET check := viable
1.5 FOR EACH $g \in Pl$
1.5.1 PERFORM Test
1.5.2 IF Test fails
1.5.2.1 SET check = not viable
1.5.2.1 BREAK
1.6 RETURN check

**Remark - 1** The Test involves, in the worst case, checking $|J| \times N$ inequalities (31) for each point collected in $Pl$. Then, the above algorithm can necessitate a huge number of iterations to finish. However, because this problem has to be solved off-line, the computational burden does not represent an obstructive limitation.

**VIII. Viable Convexified Multi-Box Approximations**

In this section, we describe a method to find arbitrarily accurate viable convexified inner-approximations of a no CD polyhedron in the case that all agents have mono-dimensional decision sets, viz. $m_i = 1, \forall i \in A$ and $A := \{1, ..., m\}$. Consider, for a no viable polyhedron $S$, a multi-box inner approximation $M(S) \subset S$. That is, according to [22], a collection of full-dimensional boxes $\{\Omega_1, \Omega_2, ..., \Omega_{m_1}\}$ such that

1) the intersection between any two boxes is not full-dimensional;
2) the union of all boxes in $M(S)$ is contained in $S$;

We will denote with $\lambda_{M(S)} = \min_{j=1, ..., m_1} l_{ij}$ the length of the minimum side of the multi-box approximation $M(S)$. The numerical method described in [22] can be used to find multi-box inner approximations $M(S)$ of $S$. Interestingly enough, it is possible to prove that under the Assumption $m_i = 1, \forall i \in A$ the convexified set $S' := \text{co}\{M(S)\}$ is always CD. To this end, let us first define the following notion of strong viability.

**Definition 8.1:** (Strong Viability) - Consider the case where all agents have mono-dimensional decision sets, viz. $m_i = 1, \forall i \in A$. A point $g \in S$ is "strongly viable" if, for each agent $i \in A$ there exists a real $\lambda_i > 0$ such that either $e_i$ or $-e_i$ are admissible directions at $g$ in $S$, i.e. $g + \lambda_i e_i \in S$ or $g - \lambda_i e_i \in S$, where $e_i$ denotes the $i$-th vector of the canonical basis of $\mathbb{R}^m$.

Please note that strong viability of $g \in S$ implies its viability. Then, it is possible to prove the following preliminary result:

**Lemma 5:** Let $g' \in \mathbb{R}^m$ and $g'' \in \mathbb{R}^m$ be two strongly viable points of $S' := \text{co}\{M(S)\}$. Then, any convex combination $g = \gamma g' + (1-\gamma) g''$, $\gamma \in [0, 1]$, of them is a strongly viable point of $S'$. In particular, for all $i \in A$, given $v_i' \in \{e_i, -e_i\}$ and $v_i'' \in \{e_i, -e_i\}$ and a positive scalar $\lambda > 0$ such that $g' + \lambda v_i' \in S'$ and $g'' + \lambda v_i'' \in S'$, $\forall \lambda', \lambda'' \in [0, \lambda]$, the following conclusions can be drawn out.

- if $v_i' = v_i''$, then
  $$ g + \lambda v_i' \in S', \forall \lambda \in [0, \lambda] $$

- if $v_i' = -v_i''$, then
  $$ g + \lambda v_i' \in S', \forall \lambda \in [0, (1-\gamma)] $$

**Proof** - Let us consider the following convex combination

$$ \gamma(g' + \lambda v_i') + (1-\gamma)(g'' + \lambda v_i'') $$

By convexity, $\forall \lambda', \lambda'' \in [0, \lambda]$ the latter point always belongs to $S'$. Then, if $v_i' = v_i''$, the latter becomes

$$ g + (\gamma \lambda' + (1-\gamma) \lambda'') v_i' \in S', \forall \lambda', \lambda'' \in [0, \lambda] $$

which is equivalent to

$$ g + \gamma v_i' \in S', \forall \lambda = \gamma \lambda' + (1-\gamma) \lambda'' \in [0, \lambda] $$

By using the same argument, when $v_i' = -v_i''$, (41) becomes

$$ g + \lambda v_i' \in S', \forall \lambda = \gamma \lambda' - (1-\gamma) \lambda'' \in [-(1-\gamma)\lambda, \gamma \lambda] $$

Finally, the viability property of each point $g \in S'$ can be proved as follows:

**Theorem 3:** Assume $m_i = 1, \forall i \in A$, let $S' = \text{co}\{M(S)\}$ be a convexified inner approximations of $S$ and $\lambda_{M(S)}$ be the minimum side of the related multi-box inner approximation. Then, any $g \in S'$ is strongly viable and hence viable. Moreover, for any point $g \in S'$ and for any $i \in A$

$$ g + \lambda v_i \in S', \forall \lambda \in [0, \lambda_+] $$

$$ g - \lambda v_i \in S', \forall \lambda \in [0, \lambda_-] $$

hold true with $\lambda_+ \geq \lambda_{M(S)}$.

**Proof** - The main observation is that, by construction, if $g$ is a vertex of the one of the box $\Omega_j$, then, for any $i \in A$, there exists a $v_i \in \{e_i, -e_i\}$ such that $g + \lambda v_i \in \Omega_j \subseteq S', \forall \lambda \in [0, \lambda_{M(S)}]$. Then, because $S'$ can be described as the convex hull of all vertices of all boxes of the multi-box inner approximations, each point belonging to $S'$ can be expressed
as a convex combination of such vertices, then the use of Lemma 5 concludes the proof.

In conclusion, a no CD polyhedron $W_{\delta}$ can always be approximated by a CD convexified inner approximation $W'_{\delta} = \text{co}\{\mathcal{M}(W_{\delta})\}$ in the case each agent governs a scalar reference. In this case, the S-FFCG problem can be rewritten as follows

$$g_i(t) = \arg\min_{g_i} \| g_i - r_i(t) \|_{\Psi_i}^2 \quad \text{subject to} \quad \left\{ \begin{array}{l} g(t) = [g_1^T(t - \tau), ..., g_N^T(t - \tau)]^T \in W'_{\delta} \\
(g_i - g_i(t - \tau)) \in \Delta G_i^\delta(g_i(t - \tau)) \end{array} \right. \ (46)$$

where the set $W'_{\delta}$ is used in the place of $W_{\delta}$. Notice also that each set $\Delta G_i^\delta$ is not subject to modification because it represents local constraints in the optimization problem (46). Then, its fulfillment does not depend on the global command vector $g$.

**Remark - 2** In order to improve the accuracy of the proposed method, the convexified inner approximation could be performed more effectively by approximating only the parts of the border of $W_{\delta}$ that contains no viable points. This possible improvement will be faced in future research activities.

The use of convexified multi-box approximations $W'_{\delta}$ in the place of $W_{\delta}$ adds to the S-FFCG scheme a **finite-time** convergence property. To prove this property we need to introduce first the following preliminary result that shows that if the "maximal admissible displacement" along one direction is smaller than a certain minimal distance, then, by performing such a displacement, the number of active constraints increases.

**Lemma 6**: Let $S' = \text{co}\{\mathcal{M}(S)\}$ be described by the set $J$ of non-redundant inequalities $a_jg \leq b_j$ and $J_0 \subseteq J$ denote the set of all active constraints for $g \in S'$, i.e. the set of constraints such that $a_jg = b_j$. Then, the cardinality of the set of active constraints for $g + \lambda v_i$ is bigger than the cardinality of $J_0$, i.e.

$$|J_{g+\lambda v_i}| > |J_0|$$

if there exist $i \in A$, $v_i \in \{-e_i, e_i\}$ and a scalar $\bar{\lambda} < \lambda_{\mathcal{M}(S)}$ such that: $g + \lambda v_i \in S'$ for all $\lambda < \bar{\lambda}$ and $g + \lambda v_i \notin S'$ for all $\lambda > \bar{\lambda}$.

**Proof -** By exploiting Theorem 3, the fact that $\bar{\lambda} < \lambda_{\mathcal{M}(S)}$ implies that $g - \lambda v_i \notin S'$ for all $\lambda > \bar{\lambda}$. Moreover, the fact that $g$ has both $v_i$ and $-v_i$ as admissible directions implies that the vector $v_i$ is orthogonal to hyperplanes related to active constraints in $g$, i.e., $a_jv_i = 0, \forall j \in J_0$. Then

$$a_j(g + \lambda v_i) = b_j, \forall j \in J_0$$

which implies $J_g \subseteq J_{g+\lambda v_i}$. The proof is completed by noticing that, because $g + \lambda v_i \notin S'$ for all $\lambda > \bar{\lambda}$, a new active constraint is added at $g + \lambda v_i$.

Thanks to the above Lemma, the following Theorem can be stated.

**Theorem 4**: Let $W'_{\delta} := \text{co}\{\mathcal{M}(W_{\delta})\}$, a convexified multi-box inner approximation of $W_{\delta}$ computed with a **minimum box side** $\lambda_{\mathcal{M}(W_{\delta})}$, be expressed as the intersection of $|J|$ non-redundant inequalities $Ag \leq b$. Then, all properties stated in Theorem 2 hold true under the same assumptions and, in addition, whenever $r(t) \equiv [r_1^T, ..., r_N^T]^T, \forall t \geq t'$, with $r_i$ a constant set-point, the sequence of solutions $g(t) = [g_1^T(t), ..., g_N^T(t)]^T$ converges in a **finite time** to a PO (stationary) (constant) solution of (21), which is given by either $r_i$, whenever $r_i \in W_{\delta}$, or by any other PO solution $\hat{r} \in W_{\delta}$ otherwise.

**Proof -** Being $W'_{\delta} = \text{co}\{\mathcal{M}(W_{\delta})\}$, assumption A2 holds and the scheme (46) fulfills the conditions of Theorem 2. The remainder of this proof will be focused on proving the finite-time convergence only. Define $\alpha = \min\{\lambda_{\mathcal{M}(W_{\delta})}, \eta^*\}$, with $\eta^*$ the same as in item 2 of Theorem 1. This quantity represents the minimum between the minimum side of the multi-box approximation and the "minimal displacement" that is always guaranteed by the constraint $(g_i - g_i(t - \tau)) \in \Delta G_i^\delta(g_i(t - \tau))$. Moreover, without loss of generality we can normalize each $i$-th cost weight $\Psi_i = 1$. This expedient allows us to make a link between the $i$-th cost and $\Delta g_i(t)$.

In order to prove finite-time convergence, the first step is to note that if $g(t'') = g(t'' + N\tau)$ then the agents would already have reached a PO solution. Then, it remains to show that, if no already in a PO solution, the number of consecutive decision "cycles" (one cycle is the time elapsed between two commands updating for a single agent) for which all agents moves less than the finite quantity $\alpha$ in the solution space is finite. This can be proved by noticing that

- For the $i$-th agent it is convenient to move of a quantity $\varepsilon < \alpha$ only in the case that, doing this kind of displacement, it is able to reach its own optimum $r_i$. Clearly, for a fixed $r_i$ this may happen at most 1 time for each agent.
- According to Lemma 6, the only remaining case where an $i$-th agent moves of a quantity $\varepsilon < \alpha$ is when a new constraints is added to the set of active constraints $J'_0$. Because inequalities are non-redundant, the maximum number of active constraints is limited by $N$. Thus, the maximum number of consecutive moves of this kind is $N$.

**Remark - 3** The Constraint Qualification of convexified multi-box approximations $S'$ has been proved only for mono-dimensional decision sets. In [17], a counter-example is available which shows that the presented convexified inner approximation does not give any guarantee of viability in the multidimensional case, i.e. $m_i > 1$ for some $i \in A$, for $m \geq 4$. However, it should be remarked that, even in this case, no equilibria that are not PO have been found. This is due to the fact that the notion of viability introduced in Definition 6.1 is sufficient to guarantee the absence of no PO equilibria but not necessary. The conjecture is that multi-box approximations also ensure convergence to PO solutions for $m_i > 1$, although the proof would probably require a much more involved notion of viability.

**A. Illustrative Example**

In this section, a short example is presented in order to clarify the matter and show the effectiveness of the proposed multi-box approximation method. The two-dimensional
approximated by 91 boxes (Figure 7) by using the algorithm presented in [22] which, on the same machine, took 37 seconds to terminate its execution. The resulting CD polytope $S'$, that does not contain $g'$ in its convex hull, is depicted in Figures 8-9.

Consider the water tank network depicted in Figure 10. The system consists of the interconnection of four cascaded two-tank sub-systems, each of them described by the following non-linear equations

$$\begin{align*}
\rho S_i h_i &= -p A_i \sqrt{2gh_i} + u_i \\
\rho S_i^2 h_i^2 &= -\alpha_i p A_i^2 \sqrt{2gh_i^2} + p A_i \sqrt{2gh_i^2} + \sum_{j \in \mathcal{S}} \alpha_j p A_j^2 \sqrt{2gh_j}
\end{align*}$$

where $u_i$ is the water flow supplied by the pump whose command is the voltage $V_i$, $i \in \mathcal{A} := \{1, \ldots, 4\}$. Moreover, for each $q = 1, 2$, $S_i^q$ are the tank cross sectional areas, $h_i^q$ the

polytopic constraint set $S$ of Figure 6 is considered. It is characterized by the following five inequalities

$$Ag \leq b$$

where

$$A = \begin{pmatrix}
-0.2693 & 0.9630 \\
0.3288 & -0.9444 \\
-0.9874 & 0.1584 \\
0.9877 & -0.1563 \\
0.9837 & 0.1797
\end{pmatrix}, \quad b = \begin{pmatrix}
6.1218 \\
-0.0681 \\
-1.7645 \\
8.4683 \\
10.7936
\end{pmatrix}$$

The CQ test presented in section V has been performed on it and the answer was achieved in 0.04 seconds by means of the Multi-Parametric Toolbox (MPT) (please see [21] for details) with MATLAB 2009b\textsuperscript{®} installed on a Intel Core\textsuperscript{™}2\textsuperscript{®} Quad machine. The polytope $S$ results no CD because of the presence of the vertex $g' = [1.908, 0.7355]^T$, which is not viable as illustrated in Figure 6. Then, $S$ has been inner-approximated by 91 boxes (Figure 7) by using the algorithm presented in [22] which, on the same machine, took 37 seconds to terminate its execution. The resulting CD polytope $S'$, that does not contain $g'$ in its convex hull, is depicted in Figures 8-9.
water level in the tanks, $A_i^j$ the section of pipes connecting the tanks, and $g$ and $\rho$ the gravity constant and the water density respectively. Their values are specified in Tables 1-2. Finally, parameters $\alpha_1 = 1, \alpha_2 = \alpha_3 = \alpha_4 = 1/2$ model the splitting water flows between upper and lower tanks.

With $S^i$ we denote the set of subsystems which provide water to the downstream tank of the $i$-th subsystem; in our case $S^1 := \{2\}, S^2 = \{3\}, S^3 = \{4\}$ and $S^4 = \emptyset$. Each cascaded two-tank subsystem has a related decision maker or agent in charge of regulating the levels $h_i^j(t)$, $i \in A$, by properly modifying their set-points and by exchanging relevant data with the other agents by means of the communication graph depicted in Figure 11.

Local decentralized tracking LQ output feedback controllers ([18]) are implemented, which act properly on the incoming water flows $u_i(t)$ so that the offset property $A_2$ is satisfied. A simple static equation is used to model the relationship between the input voltage $V_i(t)$ and the incoming mass of water.

$$u_i(t) = \begin{cases} V_i(t) & \text{if } V_i(t) \geq 0 \\ 0 & \text{if } V_i(t) < 0 \end{cases}$$

The following local and global constraints are to be enforced at each time instant

$$h_i^1 \leq h_i^1 \leq \overline{h}_i^1, \quad h_i^2 \leq \overline{h}_i^2, \quad 0 \leq h_i \leq V_{\text{max}}, \quad \forall i \in A, \quad |h_i^1 - h_{i+1}^1| \leq 5 \text{ cm}, i = 1, 2, 3$$

The system is linearized around the equilibrium $\overline{V}_i = \overline{u}_i^{eq} = 2$, $i \in A$, $\overline{h}_i^1 = 32 \text{ cm}$ and discretized with sampling time $T_c = 0.8 \text{ sec}$.

The reported simulations investigate the behavior of the overall system when the desired set-points to the water levels of the downstream tanks have the profiles depicted in Figure IX (red dashed line). At the beginning, the desired references $r_i = 32 \text{ cm}, i \in A$ correspond to an equilibrium. At time $t = 30 \text{ sec}$, the reference $r_1$ related to the downstream tank of subsystem 1 is changed from 32 cm to 42 cm. At the same time, also the reference $r_2$ is modified from 32 cm to 34 cm. These values are kept constant until time instant $t = 400 \text{ sec}$ when they are changed back to their initial values. Simultaneously, the desired references $r_3$ and $r_4$ change their values at time $t = 300 \text{ sec}$ from 32 cm to 27.85 cm and, respectively, 28.5 cm. After that, these new values are kept constant up to time $t = 800 \text{ sec}$, when they are brought back to the previous values.

In Figures 14-15, the constrained vector responses can be observed. It is important to note that such a vector violates the constraints at several time instants when no CG unit is used. On the contrary, this never happens when a CG unit is used. In particular, the responses of the centralized classical (Feedback) CG and Feedforward FF-CG schemes are contrasted with those corresponding to the proposed distributed S-FFCG approach.

The evolutions of the downstream water levels are depicted in Figure 12, while in Figure 13 the various CG actions are reported. The centralized CG and FF-CG schemes have similar coordination performance. On the contrary, the distributed S-FFCG exhibits, as expected, a slower response to changed conditions. Nevertheless, the related performance, especially during the equilibrium phases, are quite good even if compared to the centralized algorithms. Although the performance of a centralized solution, especially those based on the direct measure of the state, would be expected to remarkably outperform the decentralized solution here, on the contrary, the difference is modest as it can be observed in Figures 14-15.
A. Investigation on the scalability properties of the methods

In order to better appreciate the scalability properties and the real advantages of S-FFCG, further tests have been carried out for an increasing larger water network. In particular, a system ranging from 4 (8 tanks) up to 12 (24 tanks) subsystems (agents) have been considered with different system partitions: 1) each agent governs only one group of up- and down-flow tanks and 2) one agent governs more groups (up to four) of up-and down-flow tanks. Moreover, for each type of system several simulations have been performed also taking into account, beyond the methods already considered, the fast-S-FFCG method presented in Section VI.B. Notice also that results and the various strategies reported for the system partitioned as indicated in the above case 2) and table III will be referred to as “aggregated” S-FFCG (Ag-S-FFCG) and Fast “aggregated” S-FFCG (Fast-Ag-S-FFCG).

Simulation results have been reported in Figures 17-18. It is evident in Figure 17 that one of the main advantages of such schemes are in the low amount of data exchanged for its implementation, remarkably lower than those pertaining to other distributed approaches - e.g those based on consensus-based mechanisms. In the Figure 18. A the residual cost

$$\hat{J} = \frac{1}{T} \sum_{k=0}^{T} \|r(k) - g(k)\|^2$$

has been reported for each method, where $T$ is the amount of simulation time steps. Notice that results are related to the worst-case of step-changing reference signals. Lower differences are expected for reference signals changing more smoothly.

The sequential nature of the S-FFCG approach reveals in the linearly decreasing performance with the number of agents. Interesting enough, its faster implementation Fast-S-FFCG exhibits a better behavior. In this respect, even the aggregated versions of the S-FFCG methods present better performance.
Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.

Table III

<table>
<thead>
<tr>
<th>N° of subsystems in the Network</th>
<th>Agents Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2 (2 subs per agent)</td>
</tr>
<tr>
<td>6/9</td>
<td>2/3 (3 subs per agent)</td>
</tr>
<tr>
<td>8/12</td>
<td>2/3 (4 subs per agent)</td>
</tr>
<tr>
<td>10</td>
<td>2 (5 subs per agent)</td>
</tr>
</tbody>
</table>

Finally, in the last Figure 18.B, the scalability of the S-FFCG method in terms of computational time per agent can be appreciated. Notice how the related computational burdens have a very slow increment with the number of agents. Such a property could represent a clear advantage with respect to centralized implementation, especially w.r.t. the standard CG scheme, whose required CPU time exceeds the sampling period when the number of agents is greater than 9.

From Figure 18.B, the Fast-S-FFCG scheme seems to enjoy the same results in terms of CPU time per agent. However, for reasons detailed in Section VI.B, in this case the reported CPU time is the summation of all agents’ CUP times (essentially, it is computed as: N-of-agents × CPU-time-of a-single-agent). In fact, for this scheme, the last updating agent has to wait the conclusion of the updating activities of all other agents before completing its updating.


