PERGAMON

Computers and Chemical Engineering 23 (1999) 341–355

Computers & Chemical Engineering

Approaches to asynchronous decentralized decision making

I.P. Androulakis*, G.V. Reklaitis

School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, USA

Received 11 February 1997; revised 22 September 1998; accepted 22 September 1998

Abstract

Motivated by the ideas of asynchronous relaxation algorithms this paper investigates optimal decision-making problems that exhibit decentralized characteristics. Such problems consist of a collection of interacting sub-systems, each one described by local properties and dynamics, joined together by the need to accomplish a common task which achieves overall optimal performance. Special properties of such systems that make them ideally suited for the framework of asynchronous computing are (a) the lack of a single overall objective describing the collective performance, and (b) the asynchronism in implementing topologically optimal decisions based on information which is local in space and time. A methodology for decentralized decision making is developed based on the solution of a series of sub-problems in which each minimizes a local objective while maximizing a common Lagrangian function, by generating independent approximations of an ascent direction in the space of the dual variables. The concepts are illustrated by means of motivating examples. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Asynchronous distributed computing; Decision making; Agreement algorithms; Decentralized operations

1. Decentralized decision making

Classical centralized models of decision making and computation deal with the situation in which a single decision maker possesses (or collects) all available information related to a certain system and performs some computational and/or makes a decision so as to achieve a certain objective. Our interest in *distributed decision making* stems from the fact that real-world systems are too large for the classical model of decision making to be applicable. There may be a multitude of decision makers none of which possesses all relevant knowledge because this is impractical, inconvenient, or expensive due to limitations of the system's communication channels, memory, computation and information processing capabilities. Aggregation of data might be undesirable or even impossible. In general, distributed decision making and support systems are more flexible and robust in absorbing various and sudden changes and can accommodate more gracefully failures in individual components (Bryant, 1993; Towill, 1993).

Although a more detailed description of distributed decision-making problems will be provided shortly we point out that in the content of the Chemical Process Industries, and in particular in Process Operations, several problems are distributed by nature. Multiple batch plant facilities may cooperate in order to supply different markets while competing for utilizing limited resources (Wilkinson et al., 1994), plant-wide operations such as decentralized scheduling systems, (Hasebe et al., 1994), rely on accumulating data at different rates and analyzing them in an independent but cooperative way. Furthermore, it is clear that future computer integrated manufacturing systems will become distributed (Cowdrick, 1991). Often, within the same organization or plant, independent decisions which are related via feasibility; or other, constraints are made (Malone, 1987).

Finally, the planning subproblem, known *as the supply chain management problem* involves the coordinated operation not only of the multiple plant and distribution sites of a given business unit but also the set of production sites of its suppliers of feed materials or intermediates, the production sites of the customers who use the products, and the logistical system which is engaged to transport all of these materials. Clearly, each of these entities (plant, suppliers, customers, and shippers) has its own characteristic decision variables, is related to other

* Corresponding author. Present address: Corporate Research Science Laboratories, Exxon and Research and Engingeering C., Roure 22 East, Annandale, NJ 08801, USA. Tel.: 908 7302-111; fax: 908 730 3344; e-mail: ipandro@erenj.com

entities through production and demand constraints, and has its own characteristic economic objective functions which is in general not commensurate with that of other entities and thus cannot be lumped into a meaningful explicit overall objective function. To achieve efficient supply chain management, a plan must be developed under which each entity is allowed to make independent decisions on a time-scale appropriate to its operation so as to optimize its own objective function while satisfying constraints which link its operation to those of the entities with which it communicates. Thus, as will be apparent from the subsequent discussion, supply chain management is inherently an *asynchronous distributed decision-making problem* which involves *asymptotic agreement with computation.*

These local decisions, although related, are made at different rates and points in time. Therefore, in all of the above systems, we have individual components related by physical constraints, such as material balances, input–output relations, different descriptions of the same physical reality, etc. Each individual component is characterized by its own local dynamics and rate of generating, exchanging, data. Mathematically speaking, these decision-making problems can be viewed as *independent sub-problems with overlapping decision spaces.* Our goal then is to identify the response of the sub-systems within the overlapping regions through a process of information exchange.

Formally, the decision problem has two different aspects. It is possible to formulate the problem as a distributed decision problem whereby one tries to choose an optimal distributed scheme subject to certain limitations, such as the amount of information that can be transferred, etc. Alternatively, one can choose to focus on distributed systems with prespecified structure in which each decision maker chooses an initial decision and iteratively improves this decision as more information becomes available from the environment or other decision centers. By this we mean that the ith decision maker updates, from time to time, a decision, x^i using some formula

$$x^i \leftarrow f^i(x^i, \mathfrak{I}^i), \tag{1}$$

where $\mathfrak{I}^i$ is the information available to the ith decision maker at the time of the update.

Mathematically speaking, since there is no strict sequence according to which computation and communication take place at various decision centers, the state of computation tends to evolve according to a point-to-set mapping and possibly in a probabilistic manner that may give rise to many other states depending on which of the processing centers executes iteration (1) next and depending on possibly random exogenous information made available to the decision maker during the execution of the algorithm.

In most situations the information $\mathfrak{I}$ of decision maker i contains some of the most recent decisions of the others. However, we allow the possibility that some decision centers compute more often that they exchange information, in which case the information $\mathfrak{I}^i$ may be outdated. Thus, any formulation should include asynchronous algorithms where there is no strict a priori sequence according to which iteration (1) is carried out at the various processors. Asynchronous algorithms (Bertsekas and Tsitsiklis, 1989), have several advantages over their synchronous counterparts. Synchronization protocols may require complex implementation and might introduce considerable overhead. Furthermore, in a synchronous algorithm the progress of computation is controlled by the slowest processor. Such bottlenecks were realized early and efforts have been made to mitigate them, for instance via dynamic load-balancing schemes. Finally, in situations where the problem data changes with time, synchronous algorithms require restart protocols that may complicate the implementation. From the point of view of numerical computation, synchronous iterative algorithms are easier to understand and their convergence can be more readily established than their asynchronous counterparts. Indeed some well-known algorithms simply do not converge when implemented in an asynchronous mode.

The challenge is thus twofold. First, we need to understand, analyze and make efficient use of the principles and ideas of asynchronous distributed computing. The process of analyzing and understanding the behavior of asynchronous computation will, in the future, be crucial, in developing new numerical algorithms that are able to combine the advantages of asynchronous algorithms while preserving desired convergence properties, or even improve them by making efficient use of the synergistic effects that may occur. As a next step, we should consider the potential advantages of asynchronous distributed decision making since, as will be shown shortly, such a framework presents a very natural way of modeling reality in several distinct areas.

As a first step we have attempted to characterize the dynamic behavior of some relaxed models of asynchronous algorithms in the presence of nonlinearities and a plethora of possible outcomes was presented. Based on these results (Androulakis and Reklaitis, 1995), the computational efficiency of asynchronous algorithms was challenged in most instances. Specifically, the presence of arbitrary nonlinearities was found to render convergence of asynchronous iterative algorithms extremely difficult for a variety of problems. The extreme measures that had to be taken in some cases made the distributed counterpart of ordinary sequential algorithms substantially slower in terms of the number of iterations as well as CPU time.

As the next step, we proceed to address a more general problem, that of asynchronous distributed decision making, in an attempt to take advantage of the inherent structure of certain classes of decision-making problems, for which the framework of asynchronous computing is very natural. In short, we will attempt to use the ideas and principles of asynchronous distributed computing as a modelling tool of physically realizable systems, and not simply as a solution methodology. Our argument is that asynchronous computing with all of its shortcoming, should not be introduced in situations where asynchronism is actually absent. On the other hand, a number of decision-making problems such as the optimal allocation of resources in a computing environment, the distribution of production among multiple plant sites, the co-operation of several teams of decision makers within a certain organization, etc., have the common characteristic of being inherently distributed. Furthermore, they exhibit the need to asynchronously implement decisions that affect each other only with respect to some common knowledge. Clearly, the satisfaction of a higher goal should be the ultimate focal point of such a decision making process.

The focus of this work will be primarily decision making problems in which decision making centers (agents) are not simply committed to achieve optimal local objectives, but are also trying to achieve a global, common, goal. A coordination mechanism will therefore be needed so as to guide local objectives towards this common goal. The nature of the coordination is precisely to drive the overall system towards eventual agreement which is a higher goal than local optimality.

In what follows, we will first introduce the ideas of distributed consensus and asymptotic agreement. We will discuss the key characteristics of a specific model of distributed computing in which multiple processors update the same variable. We will subsequently reformulate the problem of asynchronous distributed decision making and will derive an asynchronous solution of it based on the ideas of the asymptotic agreement algorithms. These constructions will be illustrated by means of simple examples.

2. Asynchronous relaxation algorithms

Since the concept of *asynchronous* iteration is fundamental to the developments to be presented in the following sections we will very briefly define the key ideas of asynchronous iterations so as to not only introduce the new concepts but also to make clear the important extensions to this type of computing that is important to the subsequent developments.

Let us consider the following fixed point problem:

Let $x = (x_1, \dots, x_n) \in \Re^n$, *and the nonlinear mapping* $F = (F_1, \dots, F_n) : \Re^n \to \Re^n$. *Given* $x_0 \in \Re^n$, *we wish to find a fixed point of* F, *i.e. a point* $x^* \in \Re^n$ *such that* $x^* = F(x^*)$.

If an iterative scheme of the form

$$x(k+1) = F(x(k)), \quad k = 1, \dots \tag{2}$$

is used in order to generate approximations to x^* then, it is well known that as long as the mapping F is a contractive one, i.e. $\rho(F(x) - F(y)) \leqslant \rho(x - y))$, $\forall x, y \in \Re^n$, then $\lim_{k\to\infty} F(x(k)) = x^* \in \Re^n$ and further x^* is unique.

Let us now consider a decomposition of Eq. (2) as follows:

$$x_i(k+1) = F_i(x(k)), \quad k = 1, \dots, \tag{3}$$

$$F_i : \Re^n \to \Re^{n_i}, \quad x_i \in \Re^n.$$

In such a case, it is natural to envision a situation in which, given p partitions of the domain, i.e. $\Re^n = \bigcup_{i=1}^{i=p} \Re^{n_i}$, the workload of iteration (3) is distributed among distinct processing elements, with each performing one of the p iterations. Let us further suppose that each processing element performs its elementary operations, i.e. *computation* and *communication*, according to a given schedule. In such a case we assume that for each processing element there exists a set of times $T_i \subset \{1\ 2, \dots\}$, such that for a given processing element i if $k \in T_i$, then i performs iteration (3), otherwise that processing node remains idle, i.e. does not perform any computation. Furthermore, in large computing systems, especially in distributed ones, such a work load distribution often introduces *communication delays*. In the presence of such delays, the *asynchronous* counterpart of Eq. (3) becomes

$$x_i(k+1) = \begin{cases} F_i(x^i(k - \varphi^i)(k); \alpha(k)), & k \in T_i, \\ x_i(k), & k \notin T_i \end{cases}. \tag{4}$$

In Eq. (4), x^i stand for the realization of x^* that is known to processing element i at time k, ϕ^i is a vector of retarded arguments, $0 \leqslant \phi^i_j \leqslant k$ quantifying the amount by which the information from node j reaching node i is outdated. Finally, α stand for a set of time-dependent, node-specific parameters needed in order to guarantee convergence of Eq. (4). Conditions for convergence of such a model, under various assumptions, have been derived, (Bertsekas and Tsitsiklis, 1989). It was further observed (Androulakis and Reklaitis, 1995), that the quantitative nature of the delays, ϕ^i, strongly affects the asymptotic behavior, and thus the convergence, of Eq. (4), for general non-linear problems. It should be noted that the reason for introducing the model of Eq. (4) was so as to overcome the effects of multiple communication barriers which are the results of differences in computational power in heterogeneous distributed systems and also the result of unexpected and unpredictable network congestion.

In a sense the difference between synchronous and asynchronous iterations is similar to the difference between *Gauss–Seidel* and *Jacobi* iterations. It was realized (Androulakis, 1993), that qualitatively the key

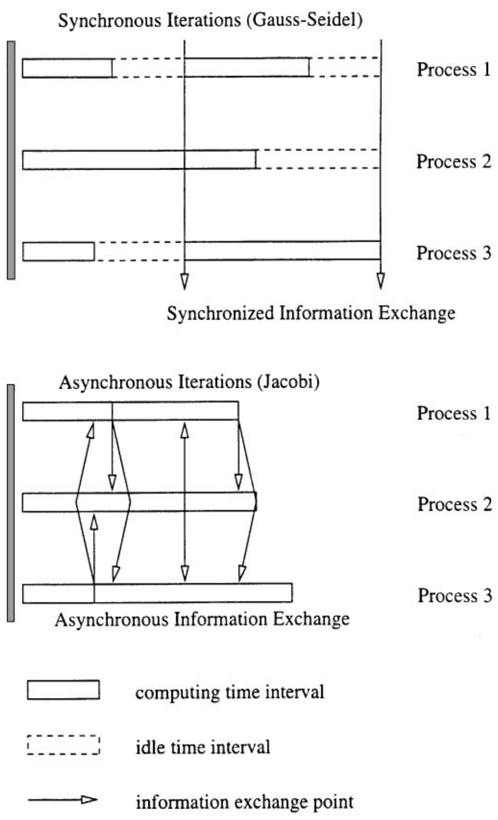
Synchronous Iterations (Gauss-Seidel)
Process 1
Process 2
Process 3
Synchronized Information Exchange
Asynchronous Iterations (Jacobi)
Process 1
Process 2
Process 3
Asynchronous Information Exchange
computing time interval
idle time interval
information exchange point

Fig. 1. Synchronous iterations (*Gauss–Seidel*) vs asynchronous iterations (*Jacobi*).

problem that the presence of delays introduce is the fact that *partial* solutions are identified by each processor performing the constituent iterations. In Fig. 1 we depict the "implementational" differences of the method that accentuate the differences between Gauss–Seidel and Jacobi iterations.

In a Gauss–Seidel setting, a global synchronization protocol is established which requires that certain processors may remain idle until all the computing element complete their corresponding computations. At that point only is communication among all processing elements allowed. On the other hand, in a Jacobi setting computation occurs as soon as an individual processing element has completed its assigned task. Communication occurs at time instances which may differ from processor to processor and depend on the local nature of the computing task.

3. Consensus in distributed systems

Although the concept of asynchronism, as introduced in Section 2, has interesting conceptual features, it also possesses a number of undesirable counter consequences. It has been argued (Androulakis, 1993) that the model of asynchronous relaxation can only be successfully applied to a limited number of problems, since the introduction of delays confounds the convergence characteristics of the original iterative schemes and, further, introduces severe implementational problems. Nevertheless, we strongly believe that we can use the basic concepts of asynchronous relaxation in order to address specific types of distributed decision making problems in which "asynchronism" is an integral part of the problem in the sense that any attempt to remove it would result in an oversimplification of physical reality. The model to be presented in what follows, expands on several of the ideas of asynchronous computing and derives a novel approach for describing decision-making problems which are inherently both distributed and asynchronous.

In this section, we will show that there exist optimal decision-making problems that are inherently *distributed* and *asynchronous*, thus making the framework of asynchronous distributed computing very well suited for their analysis. In doing so, we will introduce the idea of *asymptotic agreement* and also a model of distributed computing with *overlapping computation.*

3.1. Distributed decision making

A large number of optimal decision-making problems exhibit, to a certain extent, *decentralized* characteristics, namely, they involve a collection of sub-systems, each one characterized by its local properties and dynamics, joined together by the need to accomplish a certain common task in order to form the overall decision making problem. Clearly, a system would be perfectly decentralized if and only if there are no connections between two or more sub-systems. Such systems are informally defined *as the activity of independent agents making harmonious, non-conflicting decision* (Billard and Pasquale, 1995). A connection, expressed in the context of mathematical programming by means of constraints, represents a *potential flow of information from one sub-system to another.*

There are two main characteristics of such systems that make them ideally suited for the framework of asynchronous computing. First of all, an overall objective describing the collective behavior of all the sub-systems need not exist, but even if it does it does not have to be, or may not be, known to all of the sub-systems. As long as the effect of the interactions can be perceived by the local decision makers, the complex systems should still be driven to optimality, assuming of course that an optimal solution exists. The second characteristic, of no lesser importance, is that once an optimal decision has been made, by a certain decision maker and based on some information which is local in space and time, this decision has to be implemented with no further delay. Clearly, the first characteristic is present in any distributed system, whereas the second one emphasizes the need for an asynchronous operation. Observe, for instance, the situation depicted in Fig. 2d. We have a collection of users,

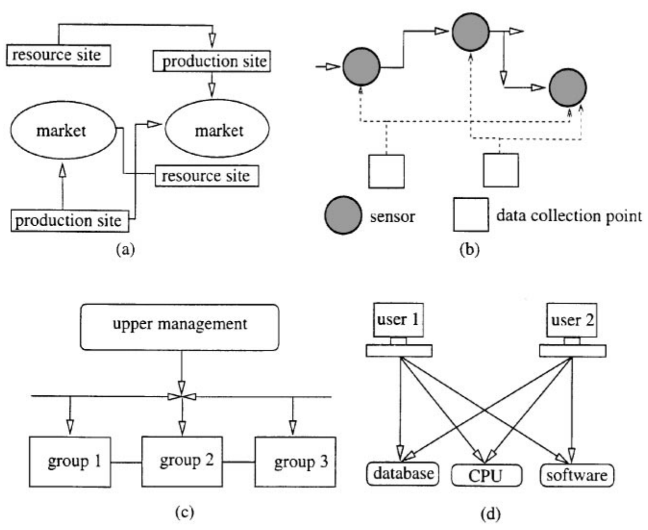

Fig. 2. Asynchronous and distributed decisions: (a) multi-site operations, (b) plant-wide operations; (c) "horizontal" flow of information, (d) sharing of computer resources.

effectively ignoring the existence of each other, attempting to access the same resources at different rates, for different time periods, and at different levels. Open computational ecologies (Ceccato and Huberman, 1989), offer the ultimate example in asynchronous and distributed decision making. Systems exhibiting those characteristics include:

- A collection of computer processes whose objective is to complete a certain task in an optimal way while, at the same time, competing for common resources.
- In an organization setting, Different decision makers within the same organization try to access and/or build a common database (Plouffe, 1987), different teams of agents seek to optimize the performance of their divisions while striving to meet specified goals, (Carlosson et al., 1992) often referred to as horizontal cooperation (Bond, 1990) different production sites have to distribute the production of certain goods among themselves so as to minimize a certain objective, while sharing some common resources or striving to meet the same due dates.
- In the Chemical Process Industries setting, cases (a) and (b) described in Fig. 2 often arise. Representative distributed operational decision-making problems include multiple batch plant operation where there is a need to coordinate resource consumption or maintain certain production levels, plant-wide control based on distributed observations and, so forth.

Clearly, these example systems are distributed in nature, and exhibit the need to reach optimal local decisions while also striving for consensus. What is even more important, is the fact that in all of these systems, once a local decision has been made (i.e. user decides to access a certain software item, or a certain number of processing nodes, a team of experts decides on a policy, a production site decides on certain production level, etc.), this decision has to be implemented with no delay in order to keep the collection of sub-systems operating. Therefore, the description of the decisions to be made is local, in the sense that the competing computer users do not need to know the details of each others problems, one team of experts does not necessarily need to know the exact objective of another team of experts, or the details of operation of a certain production facility are not important to some other facility. Thus, although a global model describing the overall operation might exist in principle, it does not have to be made explicitly known to all the members. Further, the asynchrony is introduced since the local decisions have to be implemented as soon as they are made, without waiting for approval from other entities at specific points in time.

3.2. Algorithms for asymptotic agreement

A key aspect of the previously discussed decision-making problems is *agreement*, that is, the attainment of consensus among decision makers on values of decision variables.

The associated agreement problem can be stated as follows:

a set of processors try to reach agreement on a common scalar value by exchanging tentative values and combining them by forming linear combinations of their estimates (Bertsekas and Tsitsiklis, 1989).

Pictorially, this situation is depicted in Fig. 3, where the first form of the problem involves no local variables u_i but the second form does. We first consider the former version.

3.2.1. Asymptotic agreement without computation

There exists a trivial solution to the problem of agreement without computation in which a processor is

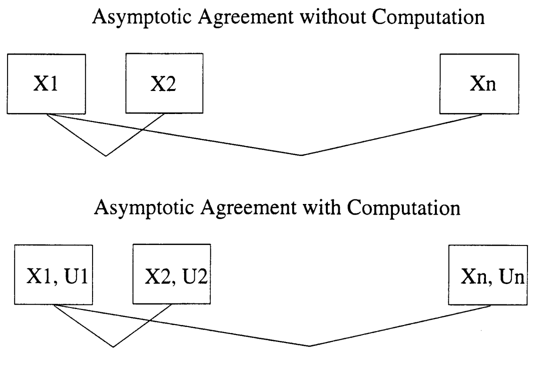
Asymptotic Agreement without Computation
X1
X2
Xn
Asymptotic Agreement with Computation
X1, U1
X2, U2
Xn, Un

Fig. 3. Distributed agreement problems.

selected and transmits its own value to all the others. Such a situation is not considered here for reasons to be explained latter.

One algorithm that addresses this form of the agreement problem can be defined as follows. Let the vector $x \in \Re^n$ and consider a set $P = \{1, 2, \dots, p\}$ of processors each having stored in its local memory the starting values of a sub-set of the coordinates of x. Let us denote by $\wp^i \subset \{1, \dots, p\}$ the subset of processing nodes having access to component i, i.e. x_i. We would like all of the processors having access to the same components to exchange messages and eventually agree on a common value. We further denote by $\Pi^p \subset N$ the set of components of x to which processor p has access. The common limit each node is expected to reach is defined as follows:

$$\min_{p \in \wp^i} x_i^p(0) \leqslant y_i \leqslant \max_{p \in \wp^i} x_i^p(0), \quad \forall i. \tag{5}$$

The agreement problem can be stated as the simultaneous solution of a set of optimization problems, as many as the number of processing nodes, with each one having the following form:

$$\min_{x_i^j i \in \Pi^j} \sum_{i \in \Pi^j} \left(x_i^j - \sum_{l \in \wp^i} \alpha_{jl}^i \tilde{x}_i^l \right)^2, \\ \sum_{l \in \wp^i} \alpha_{jl}^i = 1, \quad \alpha_{jl}^i \geqslant 0, \quad \alpha_{jj}^i \neq 0. \tag{6}$$

Analyzing Eq. (6), it is evident that each processing node j, has access to a sub-set of components $i \in \Pi^j$. There is also a set of processing node $l \in \wp^i$ that also have access to the same component i. Therefore, the current estimate of component i of node j, x_i^j, should approximate the weighted sum of the components of the other processing nodes, i.e. $\sum_{l \in \wp^i} \alpha_{jl}^i \tilde{x}_i^l$. By assuming that each node performs to completion such a minimization we observe that the local[1] solutions that are identified are

$$x_i^j = \sum_{l \in \wp^i} \alpha_{jl}^i \tilde{x}_i^l \tag{7}$$

or in vector form,

$$x^j = A^j x^L. \tag{8}$$

It is clear, that Eq. (8), via successive minimizations, defines a linear iterative map A, which has following important characteristics:

1. Since:
   (a) $A \geqslant 0$
   (b) $\sum_{l \in \wp^i} \alpha_{jl}^i = 1$
   A is a "stochastic" matrix.
2. By construction, if we follow the entries of A, there is always a path that leads from i to j. Therefore A is also "irreducible".
3. Based on the *Perron–Frobenious* Theorem, (Wilf, 1978), $\rho(A) = 1$ is an eigenvalue λ of A, and further it is essentially unique.
4. Because $\alpha_{jj}^i \neq 0$, A is "acyclic".

Therefore, the solution of Eq. (8), as a function of the iteration count, is A^k and $\lim_{t \to \infty} A^k = A^*$.

Based on the above the *underlying agreement* is convergent (8). Furthermore, because of all of the above properties Eq. (8) also belongs to the class of iterative schemes for which even their asynchronous counterpart, as presented in Section 2, is know to be convergent (Bertsekas and Tsitsiklis, 1989). Therefore, Eq. (7) now becomes

$$x_i^j(t+1) = \sum_{i \in T^i : l \in \wp^i} \alpha_{jl}^i \tilde{x}_i^l(\tau_{ij}^i(t)). \tag{9}$$

3.2.2. Asymptotic agreement with computation

We can now proceed one step further to analyzing the problem of achieving agreement when computation is also involved. In this case suppose that in addition to the variable x, on which the processing nodes have to agree, each processing node also has access to a set of "local" variables. Local variables are those associated with a specific node and whose values are not available to other nodes. A similar situation has been very elegantly analyzed in Tsitsiklis et al. (1986) where the assumption was made that all processing elements not only generate weighted sums of their estimates through a series of communication steps, but also generate, based on their own perception of the world, directions of change for their "local" variables. These directions, which can be gradient estimates for instance, are generated based on the assumption that *all processing nodes attempt to minimize the same functional.* This important and very restrictive assumption will be relaxed in our work. According to the development presented in Tsitsiklis et al. (1986), the formulation of Eq. (9) should be expanded as follows:

$$x_i^j(t+1) = \underbrace{\sum_{i \in T^i : l \in \wp^i} \alpha_{jl}^i \tilde{x}_i^l(\tau_{ij}^i(t))}_{\text{communication}} + \overbrace{\gamma_i^j(t) s_i^j(t)}^{\text{computation}} \tag{10}$$

The following powerful result was shown to hold.[2]

Proposition 1. Assume that iteration (10) is being performed. Further assume that

H1. *F is Lipschitz continuous.*
H2. $\exists K \geqslant 0; \nabla_j F(x^i(t)) E[s_j^i(t) \Im(t)] \leqslant -K \| \nabla_j F(x^i(t)) \|^2.$

[1]The term "local" as used in this context does not reflect the convex or nonconvex nature of the problem, rather it is used in a topological sense.

[2]This is a simplified version of the convergence result, the interested reader is advised to consult Tsitsiklis et al. (1986) for an excellent description of all the details.

Then it can be shown that[3]

$C1$. $\lim_{t\to\infty}\langle F(x^i(t))\rangle = F^*$.
C2. $\lim_{t\to\infty}\langle x_j^i(t) - x_j^l(t)\rangle = 0, \quad \forall i, l \in \wp^i$.
C3. $\lim_{t\to\infty}\nabla_j F(x^i(t)) = 0$.

Analyzing this key Proposition, we observe that the hypothesis part, H2, simply requires that as long as the *expected* direction $s_j^i(t)$ of update, given the previous history of updates, $\Im(t)$, up to time t, is a descent one. and as long as the underlying agreement is convergent, then: $C1$ the limit of the cost functional is the same for all participating nodes; $C2$ the estimates are the same for the nodes; and $C3$ the common limit is a minimum.

3.3. Limitation of the model of asymptotic agreement with computation

Clearly, although Eq. (10) is very powerful, it suffers from one major limitation. It requires that all processing nodes minimize the same cost functional. In the distributed decision-making setting which is of interest to us, this assumption requires that all of our decision makers have identical perceptions of the physical reality. This fact is not true in any problem which is decentralized. We can always of course reformulate it so as to fit into this description but this would be undesirable since most of the interesting and appealing properties of distributed decision making would be lost. All theories and algorithms for agreement and consensus break down when different and conflicting objectives are assigned to the decision makers. Such a situation can be illustrated through the following trivial example.

Example 1. Suppose that two decision makers would like to optimize their perceptions of the world, given by $f_1(x) = -x$ for one and $f_2(x) = x^2$ for the other, and assuming that the space of admissible values for their corresponding decision in [0, 1]. It is clear that these two objectives are not only different but also conflicting. We do know though, that this trivial problem has a solution. Nevertheless, if we allow one decision maker to minimize f_1 and the other to minimize f_2 there is not way that they will ever achieve agreement since the estimates that are being generated move one towards the upper and one towards the lower bound of the acceptable values for x. This example shows that the theories of asymptotic agreement are not applicable to agreement problems involving conflicting objectives.

In the next section, we will attempt to present a novel approach for distributed decision making problems. By combining the principles of asynchronous asymptotic agreement with Lagrangian duality theory in order to reformulate the problem in the common space of Lagrange multipliers we will develop an asynchronous algorithm for addressing certain classes of distributed decision making problems with conflicting objectives.

[3]The notation $\lim_{t\to\infty}\langle z\rangle = 0$ implies that z is 0 with probability 1 and in the mean square sense.

4. Asynchronous distributed decision making

4.1. Relations to mathematical programming

The systems under consideration exhibit a characteristic of cooperation in the sense that there exists a set of collective properties that has to be satisfied when the sub-systems, comprising the overall problem, reach optimality. By this we mean that the decisions made by each sub-system have to become locally optimal subject to the constraint that a certain set of hypothesis is satisfied. We should point out that such systems are reminiscent of the large-scale programming problems that generated the entire theory of decomposition methods. For instance, in Dantzig and Wolfe (1961), the properties of structured linear programming were very efficiently exploited to solve large LPs. Subsequently (Everett, 1966). Lagrange Multiplier constructions were employed to solve structured nonlinear as well as decentralized control problems (Lasdon and Schoeffler, 1966). Finally, the concept of "complicating variables" motivated (Geoffrion, 1972), the extension of the seminal work of Benders (1962), to the Generalized Benders Decomposition construction. The motivation behind this line of research is to identify parts of the problem that when fixed, would make the resulting problem "easier" to solve. Obviously "easy" can have different meanings as it might mean an LP with special structure, a convex NLP, or an IP with known solution. Even notions of using parallel processing were introduced as early as 1972 by Geoffrion.

Although the above decomposition constructions are not explicitly connected to this work, they must be mentioned because of conceptual similarities and the fact that some of these ideas will be used in what follows. The key differences between our approach and the earlier decomposition literature is basically that all of the above-mentioned methods assume that at some point in time, all partial results have to be collected in order to formulate a "master" problem whose task is to deal with the complicating part of the problem. This is clearly a synchronization bottleneck which we would like to avoid. There is no doubt that substantial speed-up could be achieved if one were to devise a parallel implementation of any of the above methods, as has been done very successfully for similar types of algorithms, (Zenios, 1994). Our principal interest is in addressing operational types of problems, i.e. problems where decisions are made at rates, which are different for each sub-system and have to be implemented without delay once they are made.

As was explained in the previous sections, we envision a situation were a collection of decision makers, each one operating within its own decision space, as defined by the particular problem specifications, has to come up with an optimal decision. The complexity characterizing these problems stems from the fact that the decision spaces are not disconnected regions, but rather overlapping sub-spaces as illustrated in Fig. 4. Because of such overlap, if two decisions made by different decision makers, say D_1 and D_2 take values of the corresponding decision variables from the overlap region, these decisions have to either be the same (same level of production, same due date etc.) or they have to satisfy certain property requirements (their sum must not exceed an upper bound if they represent some resource utilization etc.). Local decision are made independently in the beginning, and in later stages they will be re-adjusted so that, eventually, the connecting constraints are satisfied.

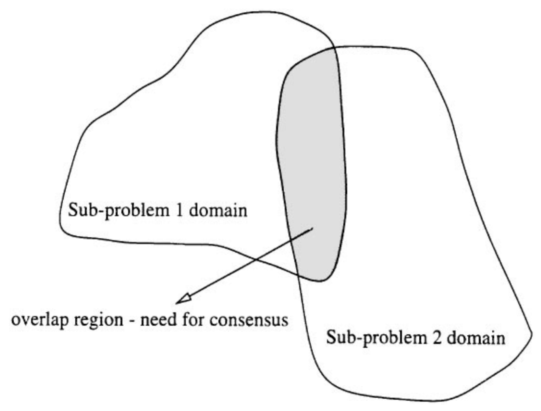

Fig. 4. Problem domain of distributed making.

4.2. *Two-level optimization and the Lagrangian dual problem*

Consider specific decision-making problems that are characterized by the following features:

- There exist n decisions that are to be made independently.
- Each local decision depends upon two distinct set of decision variables denoted by x and u respectively. The first set, x, is assumed to be local.

From a mathematical point of view, two x's belonging to different systems are not related via any constraint. The second set of variables, u, are considered to be the complicating set of decisions. From the mathematical point of view this implies that for a given component, say k, of the vector u^i, there exists a set $C_k^i \subset \{1, 2, \dots, n\}$ containing the indices j of those u^j, $j = 1, 2, \dots, n$ to which the kth component belongs.[4]

Based on the above, each decision-making problem is described by a local optimization problem of the form:

$$\begin{aligned} \min_{x^i, u^i} \quad & f_i(x^i, u^i) \\ \text{s.t.} \quad & x^i \in X^i \subset \Re^n, \quad X^i \cap X^j = \emptyset, \\ & u^i \in U^i \subset \Re^m, \quad U^i \cap U^j \neq \emptyset, \\ & g_j^i(x^i) \leqslant 0, \quad j = 1, 2, \dots, J^i, \\ & h_m^i(u_k^i \mid k \in C_k^i) \leqslant 0, \quad m = 1, 2, \dots, M^i. \end{aligned} \tag{11}$$

The sets J^i and M^i are local set of inequality and equality constraints, respectively. These need not be the same for different sub-systems. Note the fact that the complicating set of constraints, h, depends only on the complicating variables. This is a limitation that can be easily relaxed but, as will be discussed later, more elaborate updating schemes would be required. Qualitatively speaking the first set of implicit constraints expresses the local characteristics of the decision making processes, while the second set expresses the exchange of information between different decision centers.

[4]We slightly altered the notation used in Section 3. We do so because a more compact notation is easier to follow at this stage.

Example 1. We consider a simple case in order to make a few points clear. Consider the simplest form of such a decision-making process:

$$\begin{aligned} \min_{x,u} \quad & f_1(x_1, u_1) + f_2(x_2, u_2) \\ \text{s.t.} \quad & u_1 - u_2 = 0. \end{aligned} \tag{12}$$

Clearly, in the absence of the constraint, this would have been a perfectly decomposable optimization problem and thus no further analysis would be required. This would have corresponded to *harmonious non-conflicting decision.* In the context of mathematical programming, the variables u are termed “complicating” and usually “two level” optimization methods are used to solve such problems. These involves a lower level of decision making which selects values for the x's, and then a second level, at which the values of the u's are selected so as to improve the solution.

Basically, one can envision two major types of approaches for solving the high level decision making problem. One is based on solving the Lagrangian dual, to be defined shortly, directly (usually termed the “classical Lagrangian”), and others based of solving some transformation of the Lagrangian dual, say for example the Generalized Linear Programming Technique. Other methods are primarily based on the idea of Benders Decomposition, and basically solve a master problem which is a projection of the space of the complicating variables, u^i, on the feasible space.

All of these approaches have a synchronization bottleneck, as can be easily seen by the respective implementations. We will show that this bottleneck can be relaxed using the first approach, namely solving the Lagrangian dual directly, whereas the other two methods cannot since the master problems generated need, at least, a feasible solution (Minoux, 1986). If feasibility is not achieved, usually a perturbed problem has to be solved which will recover feasibility. Conceptually, this is very similar to the idea of back-tracing in the context of ODE integration, such as in shooting methods for example. Once a non-feasible solution, or one with high error, has been found, it is corrected using all the available information, and then the search proceeds. Of course, the idea of back-tracking is meaningless within the context of asynchronous computing (Androulakis, 1993), and thus such improvements have to be abandoned. Further, the synchronization bottleneck would prevent sub-systems from implementing their local policies independently. We will focus therefore on methods that address the Lagrangian Dual Problem directly.

By way of example, dualizing the constraint of Eq. (12), we obtain the Lagrangian relaxation:

$$\min_{x,u} f_1(x_1, u_2) + f_2(x_2, u_2) + \lambda(u_1 - u_2), \tag{13}$$

where λ is the Lagrange multiplier. Provided that the optimal value of λ, was known, then Eq. (13) can be solved to optimality as two separate problems yielding a solution to problem (12). In order to extend the validity of the method to cases in which the convex structure of Eq. (12) is not preserved and also to improve the computational efficiency of the method, a combination of a penalty and Lagrangian approach was proposed (Bertsekas, 1976), thus resulting in

$$\min_{x,u} f_1(x_1, u_2) + f_2(x_2, u_2) + \lambda(u_1 - u_2) + C(u_1 - u_2)^2. \tag{14}$$

Since it is clear that the presence of the cross-product term u_1u_2 will destroy the separable structure, the use of a first order Taylor Series expansion around a nominal point $(\bar{u}_1, \bar{u}_2)$ was suggested (Stephanopoulos and Westerberg, 1975), so as to preserve the separability. Thus, Eq. (14) becomes

$$\min_{x,u} f_1(x_1, u_2) + f_2(x_2, u_2) + \lambda(u_1 - u_2) + C(u_1^2 + u_2^2 - 2u_1\bar{u}_2 - 2u_2\bar{u}_1 + 2\bar{u}_1\bar{u}_2). \tag{15}$$

The classical methods for solving this problem belong to the family of two level algorithms described earlier. If we define the dual function as

$$\theta(\lambda) = \min_{x,u} \; f_1(x_1, u_2) + f_2(x_2, u_2) + \lambda(u_1 - u_2) + C(u_1^2 + u_2^2 - 2u_1\bar{u}_2 - 2u_2\bar{u}_1 + 2\bar{u}_1\bar{u}_2). \tag{16}$$

Then the dual problem, or upper-level optimization problem becomes

$$\min_{\lambda} \theta(\lambda). \tag{17}$$

It is obvious therefore that we have to perform the outer maximization in order to update the values of the Lagrange multipliers and thus to be able to move towards optimality.

As was explained earlier, there exist several ways for solving the dual problem. We will focus on the simplest choice, namely the steepest ascent method applied directly to the problem defined in Eq. (17). From classical nonlinear optimization (Bazaraa et al., 1993), we know that the dual function is characterized by a set of nice properties such as differentiability for the case where the underlying problem is differentiable, and has subgradients for the cases where it might be nondifferentiable. In the differentiable case, it can be shown, that the gradient of the dual function, i.e. the direction of steepest ascent, can be very easily computed as

$$s(\lambda) = u_1 - u_2. \tag{18}$$

Basically, the gradient with respect to a certain multiplier, or a subgradient in an ascent direction, is nothing but the constraint associated with that multiplier evaluated at the optimum point of Eq. (16). This is a fairly strong result since it provides us with an ascent direction, in the space of the dual variables, without actually having to solve the dual problem. A number of methods using the Lagrangian approach have been based on this idea.

We propose to combine such Lagrangian-based methods with the agreement algorithm and the model of distributed computing with overlapping computation as developed in Section 3 so as to derive convergent asynchronous approaches.

4.3. Asynchronous distributed solution of the dual problem

If the original decision making problem takes the form

$$\min_x \quad f(x)$$

$$\text{s.t.} \quad h(x) = 0, \quad x \in X, \tag{19}$$

then the penalized Lagrangian function associated with this problem will be

$$L(x, \lambda) = f(x) + \lambda h(x) + Ch(x)^2. \tag{20}$$

The first step of the procedure would be to minimize Eq. (20) with respect to the x's and the second step will be to solve the dual problem by maximizing the minimum of Eq. (20) with respect to the λ's. For solving the second problem, let us recall that

$$\lambda(k+1) = \lambda(k) + C(k)h(x(k)). \tag{21}$$

The index k is simply a counter reflecting the number of times (20) has been minimized. Note that updating λ according to Eq. (21) results in an increase of the dual function. The convergence properties of such schemes are well established (Bertsekas, 1976). The implementational details of this scheme will be discussed in the context of the first of the computational examples.

Within the framework discussed in Section 3, let us note the following:

- If $f(x)$ is a decomposable function, then Eq. (20) along with the expansion of the quadratic terms will provide us with independent sub-problems which can be solved separately.
- After each minimization sub-problem has been solved, the λ's have to be updated so that they generate an ascent direction within the dual space. We saw that identifying such ascent directions is as simple as evaluating a set of constraints.
- Recall that within the previous development, the constraints that we dualized contain the complicating variables, and that the Lagrange multipliers have to be known to all the decision-making sub-problems that are associated with a particular h_j, and therefore λ_j. As a result, we can envision a situation in which approximations of the same λ_j are made by different decision makers based on their local data. Each decision maker solves his own sub-problem and then can generate his own approximation of λ_j by evaluating the complicating constraints and thus generating (local) ascent directions. Recall that all Proposition 1 stated was that the direction of ascent generated by each individual processor had to be ascent/descent in the mean sense, i.e. the expected direction had to be ascent/descent. This fact implies that we can even absorb locally incorrect estimates. We can clearly see now that the model of distributed computing with overlapping computation, can be very nicely used in order to allow for the asynchronous implementation of the steepest ascent algorithm.
- In the previous development, we repeatedly mentioned the fact that a master problem, the dual, had to be maximized and that this should be the cooperative effort of all of the sub-problems. It is clear that we do not need an explicit representation of the dual problem, nor does it have to be known by all the decision makers. The only requirement is the ability of the local decision makers to generate moves that are expected to lead to ascent in the space of the dual variables.
- Finally, the initial decision-making problem was translated from the primal to the dual space. We required agreement in the space of the dual variables, and that was achieved in a distributed and asynchronous way.

4.4. Preliminary computational results

In this section we will discuss implementational details regarding the solution of the primal and dual problems, some important computational aspects related to the gradient ascent and penalty method used, as well as details regarding the use of multiprocessor machines.

As was explained earlier, the key task of each sub-problem is to provide estimates for the Lagrange multipliers associated with the complicating constraints. The primal problem given by Eq. (16) is a separable minimization problem, in the sense that each sub-problem, for fixed λ depends only on x^i, u^i. Solution of the separate minimization problems will thus yield the u's needed in order to construct the ascent direction (18). Obviously, the values of u^i are generated at different rates and they become available at different rates. The result is that the direction given by Eq. (18) is "expected" to be ascent, but this is a sufficient condition. Once an estimate of the ascent direction is obtained, using the values of the u variables as generated locally, the Lagrange multipliers λ are updated by each sub-problem following a scheme similar to Eq. (10).

The first part of the summation will be replaced by the corresponding agreement part of Eq. (10), according to which estimates of the Lagrange multipliers, as they are being communicated from other processing element will be summed, in a manner analogous to Section 3.2.1. The part corresponding to the computation step will be replace by a local evaluation of the constraint that corresponds to the particular Lagrange multiplier. Initialization of the multipliers is a key feature, but in the cases reported the starting values were $\lambda = 0$ for all sub-problems. Note that the zero values correspond to purely local solutions, i.e. the solutions without considering any interactions.

A very important question deals with the update of the penalty term $C(k)$, in Eq. (21). We used a relatively simple scheme in which the penalty parameter is updated only once a new value has been received and the update penalty parameter is "time" dependent:

$$C(k) = ab^{ck}, \quad a \approx 1, \quad \mathrm{b} \approx 1.0, \quad \mathrm{c} \approx 0.5. \tag{22}$$

Clearly, some experimentation is needed in selecting efficient values. Unfortunately, as will be discussed later, asynchronous iterations are not very well suited for developing corrective mechanisms. As a result, several '*ad hoc*" decisions have to be made.

The steepest ascent method for solving the dual problems was selected because it only incorporates first order information. From a sequential, or synchronous, point of view, second order methods would be more accurate. Note that using Hessian information about the Dual function would require knowledge about the effect of non local variables on the Lagrangian dual. We do risk to run into the problem of incorporating additional errors into our calculation in an attempt to reduce the computational effort. Finally, each local NLP was solved using the Generalized Reduced Gradient Method. The above procedures were used while solving each one of the following example problems.

Example 1 (revisited). From a compuational point of view, Example 1 is quite trivial. However, it will be used in order to illustrate some of the intrinsic properties and characteristics of the method developed with special emphasis on distributed decision making problems in which there is the need to achieve agreement under conflicting partial objective functions which produce opposing results. Recall that the objectives of the two decision makers are to minimize $f(x) = x$, for the first and $f(x) = -x^2$ for the second, and the agreement requires that at the final decision both these numbers are equal. Clearly, the centralized decision making problem is

$$\begin{aligned} \min_{x_1, x_2} \quad & x_2^2 - x_1, \\ \text{s.t.} \quad & x_1 - x_2 = 0, \\ & 0 \leqslant x_1, x_2 \leqslant 1. \end{aligned} \tag{23}$$

It should be clear that had we chosen to work in the space of the primal variables, i.e. x_1, x_2, we would never be able to reach agreement, unless each decision maker considers the entire problem, because the effects of the two optimal decisions point at opposing directions. If we generalize this problem, it is obvious that this is a problem in which even if the two decision makers share the identical information, agreement would never be reached simply because their objective are different. One approach would be to constantly change the sub-problems by introducing artificial constraints that would drive the sub-problems to consensus, but it is easy to see that such a method will fail. If, on the other hand, we view this problem from the dual perspective, we realize that the Lagrange functions has the form

$$x_1 + \lambda x_1 - x_2^2 - \lambda x_2. \tag{24}$$

We observe that this function is separable, and thus the task of maximizing the dual problem can be perceived as a unique problem, i.e. both precessors are aiming towards the same point which is the maximum of the dual function. The decomposition is somehow "artificial" and is achieved through the use of the Lagrange multiplier. Therefore, from the dual point of view, both decision makers are after the same objective, i.e. maximize the dual function. Further, they do not have to actually solve the dual problem, it suffices that they generate approximations of the dual variable by estimating an update direction in the dual space. Once agreement has been reached, then the separability of the dual problem is "exact" and the solution follows in an independent way. The two-stage problem is defined as follows:

$$\begin{aligned} &\min_{x_1, x_2} \quad x_2^2 - x_1, \\ &\text{s.t.} \quad x_1 - x_2 = 0, \\ &0 \leqslant x_1, x_2 \leqslant 1 \end{aligned} \quad \left\{ \begin{aligned} & x_1' = \arg\min_{x_1} \bar{x}_2^2 - x_1 + \lambda_1(x_1 - \bar{x}_2) \\ & \text{s.t.} \quad 0 \leqslant x_1 \leqslant 1 \\ & x_2' = \arg\min_{x_2} x_2^2 - \bar{x}_1 + \lambda_2(\bar{x}_1 - x_2) \\ & \text{s.t.} \quad 0 \leqslant x_1 \leqslant 1 \end{aligned} \right\} \quad \begin{aligned} \lambda_1 &= \alpha_{11}\lambda_1 + \alpha_{12}\lambda_2 + (x_1' - \bar{x}_2) \\ \lambda_2 &= \alpha_{21}\lambda_1 + \alpha_{22}\lambda_2 + (\bar{x}_1 - \bar{x}_2), \end{aligned}$$

It can be seen that the centralized problem is broken into two individual problems, each defined in its own domain. Communication between the two problems is in the form of locally generated estimates of the Lagrange multipliers, as well as estimates of the local solutions, i.e., $\bar{x}_1, \bar{x}_2$. It is important therefore to realize that by combining the ideas of Lagrangian duality with those of asymptotic agreement, we are able to extend the principles of distributed consensus even in problems where the decision makers have different objectives. This is a key property for general problems although so far the focus has been solely in problems in which the decision makers aim towards the same objective, i.e. minimize the same function. It is also important to mention that the partial sub-problems do not involve the complete solution of the dual problem, but simply the generation of directions of change of the dual variables which produce ascent with respect to the dual problem. The equations defining the updates of the Lagrange multipliers through the addition of the linear combination of the estimates and the local updates are the results of the application of the principles of agreement with computation.

The problem of consensus was thus moved from the primal space, in which it may not be well defined to the dual space, in which the problem was switched from minimizing the local objectives, which can be contradictory and thus generate oscillating sequences of solution points, to maximizing the dual function, which is a well defined problem because of its separable structure. In practically all of the decision making problems of interest, the partial objectives will be different and thus consensus, even when sharing identical information, is impossible. The combination of Lagrangian duality and agreement algorithms may offer an alternative to these classes of distributed decision making problems.

Example 2. The next simple example shows the effect of certain time-dependent decision parameters on the final agreed optimal point. Consider the following nonconvex problem:

$$\min \quad \{\alpha_1 + \beta_1 x^3 + \gamma_1 x + \delta_1 x\} + \{\alpha_2 + \beta_2 y^2 + \gamma_2 y + \delta_2 y\}$$

$$\text{s.t.} \quad x + y = B$$

This is a simplification of the following competitive environment. A certain market requires a certain quantity of both x and y, so that their sum must satisfy the constraints. The "production" of x and y depends on a fixed cost α, an operating cost βx^2, a storage cost γx and a transportation cost δx. The idea is to distribute the production of x and y so that the overall cost is minimized. It is assumed that each production site makes local and independent decisions that re-adjusted so that, eventually, optimality has been achieved while the requirement constraint, $x + y = B$, has been satisfied.

As mentioned earlier, one of the main characteristics of dynamic distributed decision making problems, is the fact that local changes can occur in real time, and thus the systems should be able to adapt their behavior in order to respond to those changes. In the present instance, the local changes are introduced via changes in the set of model parameters. The initial set of parameters of the model, scenario 1 of Table 1, undergoes two changes, first to scenario 2 and then to scenario 3. The first change is introduced at step 100 and the second at step 200. The results are depicted in Fig. 5, where the status of each decision makes is plotted as it is being recorded locally. Note that after about 50 computational steps, "steady state" values of the decision variables begin to be reached. As a results of the parameters changes introduced at step 100, a sharp disturbance is observed.

Table 1
Parameter values for 3 scenarios

Scenario	α_1	β_1	γ_1	δ_1	α_2	β_2	γ_2	δ_2
1	1	1	1	1	1	2	1	1
2	1	0.1	1	1	1	1	2	1
3	1	0.1	2	1	1	2	1	1

These disturbances are absorbed in a smooth way as more estimates become available through a process of information exchange, and new equilibrium points are identified. At step 200, the effects of the second set of parameter changes can be seen again and a new "steady state" is reached after some 50 steps.

It should be pointed out, that this system was simulated on two nodes of an *nCUBE/2* Hypercube machine. Each node was "simulating" one decision center. The simulation was asynchronous and thus the systems would reach the threshold of 100 computing steps at different time instances.

Example 3. This example illustrates a situation in which it is necessary to achieve local, as well as, global consensus. It assumed that independent decision makers are lumped into groups. Each decision maker has to identify an optimal policy and at the same time reach consensus with his teammates within the group. Further, global consensus has to be reached, in the sense that some decisions do have global effects, i.e. influence decision makers of other groups. Such problems arise in situations where a master planner has set some global goals that

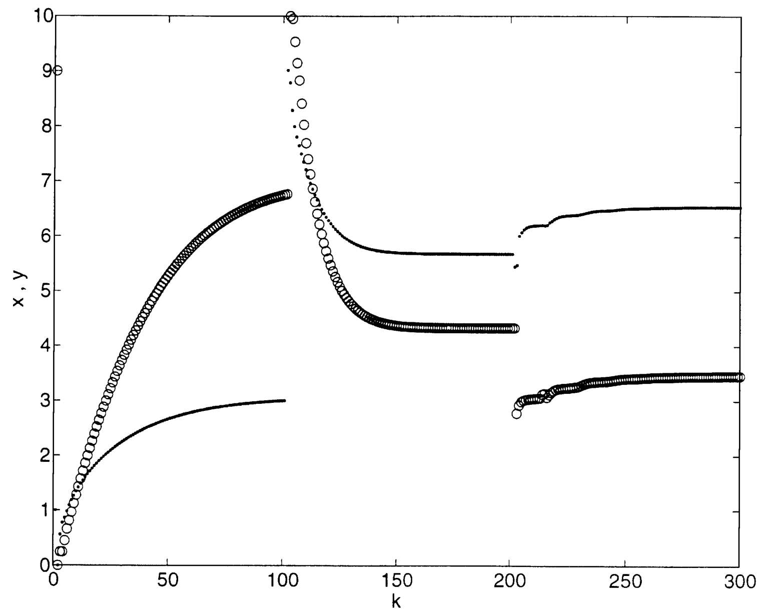

Fig. 5. Effect of time–dependent parameters on the distributed decision making, ○ = x, · = y.

have to be met (inter-group decisions), and within each group local decision are made to drive the systems to optimality (intra-group decisions). Obviously, local interactions produce global changes and vice versa. Local consensus might be the result of sharing certain resources or items. It might also be the need to produce a certain product at a given rate, or it might imply the need to produce different intermediates by the same due date. With respect to the global interactions, consensus may imply the need to maintain an optimal flow of a certain resource or final product, or might express the need to maintain a uniform rate of production. Clearly, depending on the particular instance, and provided that the interactions can be modeled as described earlier, the connecting constraints and/or variables have different meanings. The key idea is the fact that such optimal decision making problems are distributed by nature and are inherently asynchronous, in the sense that all decisions once made are implemented without delay.

Each local decision is assumed to be optimal point of the following optimization problem:

$$\begin{aligned}
\min \quad & A_i x_{i,0}^2 + B_i x_{i,1} + C_i x_{i,2} + (x_{i,3} + \tfrac{4}{i+1})^2 + x_{i,4} \\
& + x_{i,5} + 4x_{i,6} + 4x_{i,7} + 4x_{i,8} - x_{i,9} - x_{i,10} \\
\text{s.t.} \quad & - C_i x_{i,0} + x_{i,1}^2 + x_{i,2}^2 + x_{i,3}^2 - x_{i,4} + x_{i,5}^2 + x_{i,6}^2 \\
& + x_{i,7}^2 + x_{i,8}^2 + x_{i,9}^2 \leqslant 0, \\
& - A_i x_{i,0} + 2x_{i,1}^2 - x_{i,3} + 3x_{i,3}^2 + x_{i,4}^2 - x_{i,5} \\
& + x_{i,6} + x_{i,8} + x_{i,8}^2 + x_{i,10} \leqslant 0, \\
& A_i \in \{2, 6\}, B_i \in \{-25, -10\}, C_i \in \{10, 17\}.
\end{aligned}$$

The formulation gives the problem description for each sub-system. Different sub-systems are described by the same functional form using different, randomly chosen, values for the parameters A, B and C. A total of 16 sub-problems are thus created. The interactions between these 16 sub-problems can be represented via explicit equivalence constraints. One such set is as follows:

$$S_1 = \begin{bmatrix} x_{0,0} - x_{6,0} \\ x_{6,1} - x_{2,1} \\ x_{2,5} - x_{12,5} \end{bmatrix} = 0, \quad S_2 = \begin{bmatrix} x_{12,0} - x_{7,0} \\ x_{7,1} - x_{5,1} \\ x_{5,9} - x_{9,9} \end{bmatrix} = 0,$$

$$S_3 = \begin{bmatrix} x_{9,0} - x_{4,0} \\ x_{4,1} - x_{5,1} \\ x_{8,9} - x_{15,9} \\ x_{15,0} - x_{10,0} \end{bmatrix} = 0, \quad S_4 = \begin{bmatrix} x_{10,1} - x_{11,1} \\ x_{11,5} - x_{3,5} \\ x_{3,0} - x_{13,0} \\ x_{13,1} - x_{14,1} \\ x_{14,5} - x_1, 5 \end{bmatrix} = 0.$$

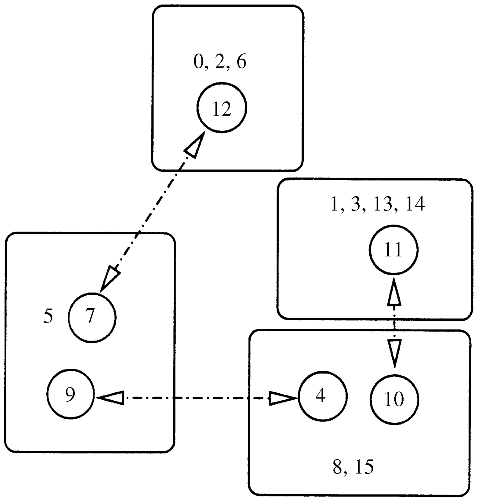

Fig. 6. Intersystem agreement.

The formulation is also depicted graphically in Fig. 6. The variables x are defined as $x_{subsystem,variable}$. In this particular instance, we have four major systems. In System 1 belong sub-systems 0, 2, 6 and 12. System 2 is composed of 5, 7 and 9. System 3 contains 4, 8, 10 and 15. Finally, System 4 includes 1, 3, 11, 13, 14. The interconnections require that sub-systems (7, 12), (4, 9) have to agree on the value of resource 0, and that sub-systems (10, 11) agree on the value of resource 1. The remaining of the constraints indicate local agreement requirements. Therefore, System 1 is composed of 40 variables for which 3 pairs must agree, System 2 has 30 variables, of which 3 pairs must agree, System 3 has 40 variables and 4 pairs that must agree and finally System 4 has 50 variables and 5 pairs which must agree.

The system was simulated on 16 nodes of an *nCUBE/2* machine. Each node, represents a decision maker continuously generating approximations by successive computation and communication. Fig. 7 shows a sample of convergence for two of the agreement constraints. Similar patterns are observed for all the connecting variables and thus only two typical examples are being presented. It is important to mention the fact that in operational-type problems, a decision sufficiently close to the optimum is very often desirable since accurate solutions, within ε, are very costly. Moreover, truly optimal decisions are almost never implemented since the operational conditions will very quickly change, thus requiring a new solution to the problem.

5. Concluding discussion

Distributed decision-making schemes, like the one just presented, make it possible to propagate local changes in

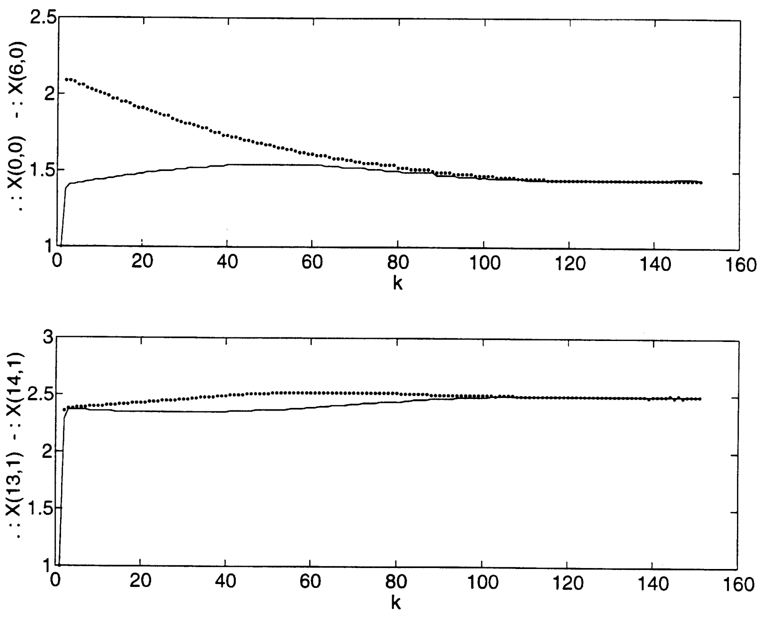

Fig. 7. Convergence of connecting variables.

order to cause global effects without the explicit description of such deviations. Furthermore, although the overall decision-making problem can be described by a single objective there is no need to do so. The only information required by the constituent components is the knowledge regarding the consensus, as far as the complicating variables are concerned. In our case each component minimizes a local objective while at the same time trying to maximize the common Lagrangian function by generating independent approximations of an ascent direction, in the space of the dual (Lagrange) variables. This direction is subsequently followed by each process (decision maker) in order to generate a new approximation of the common variable. Once again, we have to emphasize the fact that the consensus problem has been moved from the space of primal variables to the space of dual variables. The agreement is defined on the dual space, since this is the only way that we can guarantee reaching agreement. Notice that if we remain in the primal space, each decision maker would have to minimize a different objective. In this case, even in the presence of identical information, agreement is not possible (Tsitsiklis et al., 1986). By introducing the idea of the relaxed problems, and solving the resulting maximization problem, we were able to propose solutions for a certain class of distributed agreement problems, in which the decision makers utilize different objective functions.

We proposed the use of the ideas and principles of asynchronous computing as a modeling tool and not simply as a solution methodology. We identified a family of decision-making problems that can be inherently characterized as *distributed* and *asynchronous*. The systems are distributed since they essentially relate to decision-making problems where the optimal decision is the cooperative result of a set of independent decision makers, each one being characterized by its own, local, decision/utility function and its own decision space. The cooperation in those systems stems from the fact that parts of the decision spaces are overlapping and thus agreement among the different decision makers has to be achieved within the boundaries of the overlapped regions. The asynchrony of operation is due to the fact that these systems are essentially dynamic entities whose optimal policy has to be implemented continuously as soon as it has been identified.

Although the principles and ideas of asynchronous computing are almost 25 years old, they are still underdeveloped mainly due to their complexity. Thus, asynchronous algorithms are still in the development stage and this is the reason why the model of asynchronous computing has not yet enjoyed a widespread acceptance. In this work, we were able to not only make use of an interesting model, but also to present a novel way of addressing distributed decision making problems. The approach can undergo multidimensional extensions to any type of searching or learning algorithm that is inherently distributed and asynchronous. Furthermore, the simple reformulation that we proposed which makes use of “elementary” knowledge of Lagrangian duality, allowed us to address in a truly distributed and asynchronous way, distributed agreement problems in which the decision makers use different and often times conflicting objectives. To the best of our knowledge this is the first attempt to address this issue. Nevertheless, a lot of work remains to be accomplished. Some important questions that need to be addressed are:

1. We need to examine more elaborate updating schemes with respect to solving the dual problem to allow for the treatment of more general and complex connecting interactions. First, the restriction on the form of the complicating constraints has to be lifted and more sophisticated methods for updating the multipliers have to be developed. However, further research is required to effectively adapt similar methods to the asynchronous and distributed computational mode.
2. We need to address the problem of *hard* constraints. In Example 2, for instance, although the disturbances were very nicely absorbed, for a short period of time the connecting constraint was violated.
3. It is necessary to address the problem of existence of non-convexities in the model. The interest is not so much focused on identifying the *globally optimum* response as opposed to a *locally optimum* one, as it is to understand the interesting and rich behavior that nonconvexities induce in dynamically iterating systems in the presence of time delays. (Androulakis and Reklaitis, 1995).

Acknowledgements

The authors wish to thank Prof. E.N. Houstis for providing unlimited access to the *nCUBE/2* Parallel machine of the Computer Science Dept. at Purdue University.

References

Androulakis, I. P. (1993). *Asynchronous distributed decision making.* Ph.D. thesis, Purdue University.

Androulakis, I. P., & Reklaitis, G. V. (1995). Analysis of the spurious behavior of asynchronous relaxation algorithms. *Comput. Chem. Engng*, *19*, 827–845.

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (1993). *Nonlinear programming theory and applications.* New York: Wiley.

Benders, J. F. (1962). Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, *4*, 238–252.

Bertsekas, D. P. (1976). Multipler methods: a survey. *Automatica*, *12*, 133–145.

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and distributed computation: Numerical methods.* Englewood Cliffs, NJ: Prentice-Hall.

Billard, E. A., & Pasquale, J. C. (1995). Adaptive coordination in distributed systems with delayed communication. *IEEE Trans. Systems Man Cybernet. 25*, 546–554.

Bond, A. H., (1990). Distributed decision making in organizations. *IEEE Int. Conf. on Systems, Man, and Cybernetics* (pp. 869–901).

Bryant, G. B. (1993). Developments in supply management control systems. *FOCAPO Conf. Proc.*

Carlosson, C. D., Ehrenberg, P., Eklund, P., Fedrizzi, M., Gustafson, P., Lindholm, P., Markuryeeva, G., Riissanen, T., & Ventre, A. G. S. (1992). Consensus in distributed soft environments. *Eur. J. Oper. Res.*, *25*, 165–185.

Ceccato, H. A., & Huberman, B. A. (1989). Persistance of nonoptimal strategies. *Proc. Natl. Acad. Sci. USA*, *86*, 3443–3446.

Cowdrick, R. M. (1991). Distributed CIM — Planning for the future. *Comput. Ind. Engng*, *21*, 1–4.

Dantzig, G. B., & Wolfe, P. (1961). The decomposition algorithm for linear programming. *Egronometrika*, *29*, 767–778.

Everett, H. (1966). Generalized Lagrange multiplier methods for solving problems of optimum allocation of resources. *Oper. Res. 11*, 399–417.

Geoffrion, A. M. (1972). Generalized benders decomposition. *J. Optimi. Theory Appl.*, *10*, 237–260.

Hasebe, S., Kitakima, T., Shiren, T., Kurakami, Y., & Hashimoto, (1994). J. Autonomous decentralized scheduling system for single production line processes. AIChE National Meeting.

Lasdon, L. S., & Schoeffler, J. D. (1996). Decentralized plant control. *ISA Trans.*, *5*, 178–183.

Malone, T. W. (1987).1 Modeling coordination in organizations and markets. *Manage. Sci. 33*, 1317–1332.

Minoux, M. (1986). *Mathematical programming theory and applications.* New York: Wiley.

Plouffe, W. (1987). Databases in a distributed plant environment. In G. V. Reklaitis, & H. D. Springs (Eds.), *Computer aided process operations.*

Stephanopoulos, G., & Westerberg, A. W. (1975). The use of Hestenes' method of multipliers to resolve dual gaps in engineering systems optimization. *J. Optimization Theory Appl. 15*, 285–309.

Towill, D. FR., (1993). Supply chain dynamics — the change engineering challenge of the mid 1990s. *Proc. Institute Mech. Engineers*, *106*.

Tsitsiklis, J. N., Bertsekas, D. P., & Athans, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Trans. Automat. Control*, *AC-31*, 803–812.

Wilf, H. S. *Mathematics for the physical sciences.* New York: Dover, 1978.

Wilkinson, S. J., Shah, N., Pantelides, C. (1994). Scheduling of multisite flexible production systems. *A.I.Ch.E.* National Meeting.

Zenios, A. S. (1994). Parallel numerical optimization: Current status and annotated bibliography. *ORSA J. Comput. 1*, 20–42.