

3. On Coordination, Externalities, and Organization

Stanley Reiter

1. INTRODUCTION

Economic environments with nonconvexities or externalities—nonclassical environments—pose challenges for both economic theory and economic policy, simply because the best-understood economic institutions and mechanisms cannot be shown to perform satisfactorily in nonclassical environments. The challenge is to design institutions, mechanisms, or organizational arrangements that would perform well, or at least better than existing arrangements, in nonclassical environments.

The fundamental problem is to coordinate the activities of a set of economic agents. A coordination problem is associated with a decision function, whose value in any particular environment is determined by the combination of actions that each agent is to carry out. An example of an economic coordination problem is to make agents take actions that produce a Pareto-optimal allocation, whatever the prevailing environment.

Because information about the environment is necessarily dispersed among agents, and because the capacities of agents to communicate and to compute are limited, the task of figuring out what actions should be taken can be formidable. Even ignoring incentive issues (as is done here), the resources and informational costs required actually to “compute” the decision function, and to determine who should do what, could be immense in a large society. The approach of mechanism design has been to recognize that information about the environment is dispersed among agents, and that the information and enforcement costs of solving the coordination problem via decentralized mechanisms are less than for centralized mechanisms. In nonclassical environments, coordination can be much more complicated and costly than in classical environments, where decentralized solutions are known to be possible.

The nonclassical coordination problem addressed in this chapter is an

externality problem involving location. The many manifestations of this problem are typified by location choice for a set of industrial plants, and include as well locations for waste treatment plants, parks, and public facilities. A brief summary of a model in Reiter (1995) is presented to demonstrate how an organizational structure can emerge as the solution of an optimization problem. The "unknown" to be solved for is a variable with values that can be interpreted as different organizational structures.

2. THE DESIGN OF ECONOMIC ORGANIZATIONS

Design of economic organizations begins with the following elements. First, a set of economic environments is given, expressing the economic circumstances in which the organization is to function. In a classical setting of exchange among economic agents, the environment consists of the commodity space or consumption set, the preference structure for each agent, and the initial endowment of each agent. Second, there is a space of possible outcomes of economic activity, in this case the allocations of commodities to individual agents. Third, there are criteria for evaluating outcomes.

In the case of exchange, a familiar criterion is that the outcomes should be Pareto-optimal in the context of the prevailing environment. This criterion is represented by a correspondence (a set-valued function) associating to each environment its set of Pareto-optimal allocations (or trades). We shall restrict attention to criteria that are expressed by functions, i.e., are single-valued. A function that associates to each environment some Walrasian allocation would be an appropriate criterion in the exchange example. A function used to evaluate outcomes is called a *goal function*.

Generally, agents do not know which of the possible environments actually prevails. At most, individuals are able to observe or experience directly only part of the environment, such as their own endowment and preferences, and some aspects of technology. In other words, information about the prevailing environment is dispersed among economic agents, and no one knows it all.

Following Reiter (1995), an organization "computes" the goal function subject to constraints on the computational powers of the agents who carry out the computations. That is, an economic organization operating in a particular environment uses dispersed information to achieve the outcome(s) specified by the goal function. Thus, a mode of organization is

- (1) an algorithm for computing the function that characterizes the organization's desired behavior (which could be the goal function itself, or a decision function derived from it); and

- (2) an assignment of the steps of the computation to individual agents, the assignment being subject to constraints reflecting the technology of information processing and the capacities of the agents.

These concepts are illustrated below in the context of a nonclassical economic environment featuring externalities.

3. RESOURCE ALLOCATION WITH EXTERNALITIES

A class of problems of particular interest from the standpoint of design of economic organizations is the allocation of resources involving externalities, positive or negative. This class is even more interesting when the resources are indivisible. If so, the classical welfare theorems fail because of nonconvexity of the decision space, as well as indecomposability due to the presence of externalities. The analysis here follows Reiter and Sherman (1962).

We shall address the problem of locating factories in cities, when the returns to each factory depend on which other factories are located in the same city. The goal function to be maximized is the joint returns to all factories.

The resulting mathematical problem is maximization of a set-function over a class of partitions of a finite set. The partitions represent the alternative location possibilities. Many well-known examples of externalities fall into this class. For example, instead of locating factories in cities, the problem might be to assign cities to (shared) waste disposal facilities. In this case, each subset in a partition of the set of cities would consist of those cities sharing a common facility.

Here, the environment consists of

- the list of cities $\{1, 2, \dots, K\}$;
- the list of processes (also called factories or plants), $\bar{m} = (1, 2, \dots, m)$.

Each process $i \in \bar{m}$ is described by a vector of operating characteristics, θ^i . The present example has the following parameters:

- the number of plants, m ;
- the number of cities, K ;
- the numbers of components of the vectors θ^i , for $i = 1, \dots, m$.

The goal function $F(\cdot)$ is defined as follows. The payoff to process i may depend on the city in which it is located, as well as on which other plants are

located in that city. Thus, for each i there is a function $f_i^j(\theta^i, \theta^j)$ which represents the increase in i 's payoff when j is located in the same city as i , where θ^i and θ^j are the vectors of characteristics of i and j , respectively. Of course, j 's presence may help or harm i , so $f_i^j(\theta^i, \theta^j)$ may be positive, negative, or zero. A location pattern is represented by a partition of \bar{m} into K subsets, denoted

$$P_{\bar{m}} = \{P_1, P_2, \dots, P_K\}.$$

Let i be a factory in \bar{m}_i , and let $P(i)$ denote the set in the partition $P_{\bar{m}}$ that contains i . Then the total payoff to plant i when $P_{\bar{m}}$ is the location pattern is¹

$$\sum_{j \in P(i)} f_i^j(\theta^i, \theta^j),$$

and the total payoff to all plants—the social payoff—under location pattern $P_{\bar{m}}$ is²

$$F(P_{\bar{m}}) = \sum_{i \in \bar{m}} \sum_{j \in P(i)} f_i^j(\theta^i, \theta^j).$$

As explained below, an organization consists of an algorithm, expressed as a modular network for computing the desired outcome, and an assignment of the modules of the network to agents. In this case the algorithm computes a partition of the set of processes that maximizes the social payoff, $F(P_{\bar{m}})$, subject to restrictions on what each agent can (1) observe directly, and (2) compute.

We shall apply an algorithm from Reiter and Sherman (1962) to this problem. (See also Reiter and Sherman, 1965.) We assume that an agent may observe θ^i for at most one particular value of i and that, for any agent, f_i^i is an elementary function for all i and j .

The algorithm may be described informally as follows. One agent, say, agent 0, plays the role of coordinator. Agent 0 selects a K -set partition of \bar{m} , say, P , and announces it to agents $i = 1, 2, \dots, m$. Agent i is the manager of process or plant i . Let $P(i)$ denote the subset of P that contains plant i . Then agent i calculates the change in the total payoff that would result from moving plant i from $P(i)$ to P_k for $k = 1, 2, \dots, K$, in turn. Agent i proposes to move plant i to the first set P_k for which the change in total payoff is positive, i.e., for which

$$\sum_{j \in P_k} f_i^j(\theta^i, \theta^j) - \sum_{j \in P(i)} f_i^j(\theta^i, \theta^j) + \sum_{j \in P_k} f_j^i(\theta^i, \theta^j) - \sum_{j \in P(i)} f_j^i(\theta^i, \theta^j) > 0.$$

(The first two terms represent the change in the payoff to firm i that would result from the move, and the last two the effects of i 's move on all other plants in $P(i)$ and P_k .)

The proposed move leads to a new partition from which agent $i+1$ calculates whether or not to propose a move of plant $i+1$. This process continues in sequence through the agents until a partition, say P' , is reached from which no agent proposes to move her plant. Then P' is a locally maximal partition, but is not necessarily optimal. At this point the coordinator records the partition and its value $F(P')$, and then either repeats the process with a new starting partition selected at random, or stops and announces that P' is the solution location pattern. The coordinator uses a stopping rule to determine whether to continue the process of sampling from the distribution of locally maximal partitions, and the distribution of their payoff values.

The sampling process results from the fact that the algorithm just described organizes the set of K -set partitions of \bar{m} into a "forest" of disjoint trees (acyclic directed graphs, each having a root). The root of a tree corresponds to a locally maximal partition. Any probability distribution on the set of partitions induces a probability distribution on the set of locally maximal partitions, and also on the set of locally maximal payoffs. This structure is described in detail by Reiter and Sherman (1962), and in a more general context by Reiter and Sherman (1965).

It can be demonstrated that

- this algorithm converges to an optimal partition in a finite number of steps;
- the probability that an optimal partition is found in a "feasible" number of steps is "high";
- the probability that a "near-optimal" partition is found in very few steps (typically two or three) is "high".

In terms of organization design, consideration of the possible assignments of computations to agents under our assumptions about who may observe what makes it clear that the organization will not be decentralized under every efficient assignment. Since θ^i is not observed directly by agent $i \neq j$, it is not possible for agent i to calculate the value of $f_i^j(\theta^i, \theta^j)$ on the basis of his own information. Therefore the value of θ^j must be transmitted from the agent who observes j to the agent who computes f_i^j . This means that the number of variables to be communicated from one agent to another (called "crosslinks" in section 4.2) cannot be bounded independent of the number of parameters. Consequently the organization cannot be decentralized. This remains true even if it is possible to economize on the amount of communication by assigning the computation so that the values of the functions f_i^j are transmitted among the agents, rather than the values of all the parameters, θ^i , since even

with such an assignment it would be necessary to transmit the parameters needed to compute each f_i^j .

If agent i were to calculate only the change in the payoff to plant i resulting from a move to another city, omitting the effects of i 's move on the other plants, then the sequence of changes of location may not result in a locally maximal partition; in fact, it may not lead to a determinate partition. For example, suppose there are two cities and two plants, and that

$$f_1^1(\theta^1, \theta^2) = f_2^2(\theta^2, \theta^1) = 0, \quad f_2^1(\theta^1, \theta^2) = 1, \quad f_1^2(\theta^2, \theta^1) = -1.$$

Then it is easy to verify that if agent i considers only the effect of a move of plant i on the payoff to plant i , then the sequence of locations for plants 1 and 2 oscillates indefinitely.

For a problem of sufficient size (a large space of environments; see below), enough communication among different agents will be required to make it preferable to internalize the required communication. This result is, of course, not surprising in a problem in which nonconvexity and externalities are important features.

However, it is also the case that if the number of parameters is sufficiently large, then the problem is infeasible. In other words, there is no organization that would solve the problem!

4. THE MODULAR NETWORK MODEL OF COMPUTING

The model of computing on which the organization model of Reiter (1995) is based is the modular network. This model can not only express limitations on the computational powers of the agents, it can also be used to analyze the implications of those limitations for behavior and institutional structure. We present here a very brief exposition aimed at communicating the model and the ideas embodied in it with a minimum of technical apparatus. Formal definitions, theorems, and proofs can be found in Mount and Reiter (1990, 1994; see also 1982).³

There is a prescribed class of environments within which an organization can operate. The desired behavior of the organization is described by a decision function that relates the available information about the environment to the decision to be taken. This function may be the goal function itself, or a function derived from it. The task of managing the organization includes figuring out the decisions to be taken in the given environment, i.e., computing the decision function. Because of limitations on the information processing powers of individuals, the task of computing the decision function must in general be performed by more than one person. It is the task of the

designer of the organization to decide how the computation should be assigned.

As formalized by Reiter (1995), an individual, or agent, is characterized by two features, the environmental parameters that the agent can observe directly, and the elementary functions that the agent can evaluate. Generally, there are restrictions of the agent's observable environmental parameters; for example, an agent might be able to determine either a parameter of the production function, or a parameter of the demand function, but not both. Likewise, the set of elementary functions that the agent can evaluate may be limited. Finally, it is assumed that any elementary function that the agent can evaluate takes one unit of time, and the agent can evaluate only one function at once.

With these assumptions, the overall computing task is accomplished by assigning the modules of the network to individual agents in a way that reflects the underlying algorithm. Before going further into the problem of assigning tasks to agents, we consider the relationship of the underlying algorithm and the computing model.

4.1 Algorithms

An algorithm is represented by a directed graph with elementary functions assigned to the nodes. A (directed) arc from node A to node B expresses the fact that the function at node A must be evaluated and its value transmitted to node B before the function assigned to node B can be evaluated. Thus, the graph corresponding to a particular algorithm expresses the constraints that algorithm imposes on the order of performance of the elementary operations. Some operations can be carried out in parallel while others must follow one another in a fixed order.

An elementary function assigned to a node of the graph is called a *module*. A module can be visualized as a black box with possibly many input lines and one output line, taking one unit of time to compute its output from its inputs.

A *modular network* consists of modules wired together, subject to the condition that each input wire is connected to at most one output wire. Thus, a modular network is specified by an assignment of elementary functions to the nodes of a directed graph. An elementary operation (the evaluation of an elementary function) is assumed to take one unit of time, and counts as one step in the computation.

The class of elementary functions is a primitive of the modular network model, i.e., it plays the role of an undefined term in an axiomatic system. Thus, the set of elementary operations can be different in different applications of the model. For example, what is considered to be elementary may vary with the means available for computing. In some circumstances,

basic arithmetic and logical operations may be taken to be the only elementary operations, but in other circumstances—say, when the computing is to be done by a person equipped with a personal computer and a program for finding the roots of a polynomial of degree p in n variables—then finding such roots might be included as an elementary operation.

As this observation makes clear, the fact that the set of elementary functions is a primitive of the model gives the modeler control over the level of resolution of analyses carried out with the model. Once the set of elementary operations is specified, then the analysis of a function requires only that the computation be reduced to the level of elementary operations.

A commonly used notion of the complexity of a computation is the time it requires. When an algorithm to calculate a function is represented in terms of sequence(s) of operations, then the complexity of that function is related to the length of the longest sequence that might arise in the computation, or, alternatively, the average (in some sense) length of all possible sequences. Of course, the directed graph corresponding to an algorithm for computing a given function may contain loops, which would make it difficult to determine the complexity of the function. However, Mount and Reiter (1990, 1994) show that for every graph with loops that computes a given function G in time t , there is a loop-free graph, in fact a standard tree, that uses the same elementary functions and computes G in time t . We can therefore confine attention to algorithms whose graphs are trees.

The set of elementary functions provides a formal model of limitations on computational powers. For example, psychologists have pointed out that individuals can pay attention to only a limited number of things at any one time. (See, for instance, Miller, 1956.) This limitation on computational powers can be expressed by the condition that an elementary operation can be a function of no more than a specified number of variables.

Let F be a class of elementary operations. A modular network containing only elementary operations f in class F is called an F -network. If the number of arguments of an elementary function is restricted, say to r variables, each of which is at most d -dimensional, we call the network an (r, d) -network with modules in F . Where there is no ambiguity, we omit explicit reference to F . The possibility that $d > 1$ is important, because it allows the inclusion of conditional switches among the elementary functions. This extends the class of algorithms that can be modeled beyond the so-called straight-through algorithms.

The complexity of a function relative to F is the minimum time required to compute that function by an F -network. A function is not computable relative to F if there is no F -network that computes it in finite time.

Computation can be extended from integers and discrete functions to continuous variables and functions in a natural way. This extension, and

other issues related to the interpretation of the model and its application to human beings, are discussed in Mount and Reiter (1990), and, less formally, in Mount and Reiter (1994).

4.2 Assignments

An assignment of the modules to the network that expresses an algorithm to compute a decision function determines several properties that affect costs. First, it determines the length of time it takes to compute the decisions, called the *delay*. Second, it determines the number of agents employed in the computation. Third, it determines the amount of communication required to carry out the computation.

There are two types of communication:

- (1) an agent may evaluate a module and then use the results of that evaluation as an input to the next computational step he or she is assigned; or
- (2) an agent may use as an input the result of a computation performed by another agent.

Communication of type (1) is internal to the agent, and may be considered as retrieval of information from memory. It is represented in the assigned graph by an arc originating at one node assigned to an agent and terminating at a node assigned to the same agent. Such an arc is called a *selflink*. Communication of type (2) is represented by an arc joining nodes assigned to different agents, called a *crosslink*.

A further distinction relating to crosslinks can be useful. Two crosslinked agents may be in the same organization, or in different organizations. Reiter (1995) discusses the motivation for this distinction, as well as the basis for distinguishing informationally separate organizational units. We call a crosslink an *internal crosslink* if it is an arc between different agents in the same organization, and an *external crosslink* if it joins agents in different organizations. Part of the task of designing an organization is to decide which crosslinks to internalize and which to make external. The distinction is best expressed in terms of a cost function, of which the arguments are the delay, the number of agents employed in the computation, and the numbers of selflinks, internal crosslinks, and external crosslinks. External crosslinks may be converted to internal by incurring a fixed cost, interpreted as the cost of creating the infrastructure of internal communication; the cost per message for internal crosslinks is less than for external crosslinks.

It is shown in Reiter (1995) that cost-minimizing assignments must be efficient, where efficiency is defined to be a vector minimum of the vector

with four components—delay, number of agents, number of selflinks, and number of crosslinks. In fact, it suffices to look for efficient assignments in two dimensions—number of crosslinks and delay. Characterizations of efficient assignments and algorithms for generating them are given in Appendices I and II of Reiter (1995). An efficient assignment determines an Efficient Assigned Directed Acyclic Graph (EADAG). The EADAG solves the problem of how to compute the decision called for by the decision function embodied in the algorithm. Its structure reflects not only the algorithm, but also the constraints on the agents' computing and information processing powers. It is an organizational structure that solves the problem.

To interpret efficient assignments in terms of organizational structure, we must introduce the concept of a large class of environments. A class of environments is *large* if it contains environments characterized by an arbitrarily large number of parameters. For example, consider a pure exchange economy with two individuals and two goods. This economy constitutes a large class of environments if all convex continuous preference relations are included, but not if the only allowable preference relations are those determined by Cobb-Douglas utility functions, as the latter are determined by at most three parameters.

Suppose the possible environments form a large class. If an efficient assigned graph that computes the desired decision function can be split into component subgraphs such that the number of crosslinks between the subgraphs is independent of the number of parameters, then those components are interpreted as representing informationally distinct organizational units—i.e., separate organizations. Otherwise, decision costs can be reduced by internalizing the crosslinks between the component subgraphs—i.e., fusing them in a single organizational unit. In other words, it will pay to build the infrastructure needed to support the communication required to achieve the specified coordination. In still other words, it will pay to centralize rather than decentralize the organizational structure.

In the case of a pure exchange economy with two agents and two goods, the computation of a Pareto-optimal allocation (or trade) results in an assigned graph with only four crosslinks (two, if only verification of equilibrium is required). Since the number of parameters specifying continuous and convex preferences is unbounded, the number of crosslinks is independent of the number of parameters. In fact, this organization is the canonical example of decentralization.

NOTES

1. More generally the external interactions among factories could be written so that the profit to plant i would depend on the subset $P(i)$. Thus, the profit to any plant could involve interactions among triples and higher order groupings of factories. Our assumption that the payoff is additive in the bilateral interactions can be regarded as a second order approximation to more general external interactions.
2. To express the effect of the city on the payoff to plant i , we could introduce K additional "fictitious" plants, each one permanently located in one of the K cities. Then the effect of locating plant i in city k would be expressed by the function $f_k^i(\theta^i, \phi^i)$, where ϕ^i is the characteristic vector of the fictitious plant located in city k . For simplicity we ignore these fixed plants.
3. The modular network model is an extension of the neural network model of McCulloch and Pitts (1943) to cases in which the alphabet may be a continuum and the functions continuous or smooth. It should be noted that the McCulloch-Pitts neural network model is equivalent to the finite state sequential machine model in the sense that every function computable by either a finite state machine or a neural network can also be computed by the other. For details, see Arbib (1969).

REFERENCES

- Arbib, M.A. (1969). *Theories of Abstract Automata*, Prentice Hall, Englewood Cliffs, NJ, USA.
- McCulloch, W. and W. Pitts. (1943). 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics* 5, 115-133.
- Miller, G.A. (1956). 'The magic number seven, plus or minus two: Some limits on our capacity to process information', *Psychological Review* 63, 81-97.
- Mount, K. and S. Reiter. (1982). 'Computing, communication and performance in resource allocation', Presentation, Decentralization Conference, Minneapolis, MN, USA.
- Mount, K. and S. Reiter. (1990). 'A model of computing with human agents', Center for Mathematical Studies in Economics and Management Science Discussion Paper 890, Northwestern University.
- Mount, K. and S. Reiter. (1994). 'On modeling computing with human agents', Center for Mathematical Studies in Economics and Management Science Discussion Paper 1080, Northwestern University.
- Mount, K. and S. Reiter. (1996). 'A lower bound on computational complexity given by revelation mechanisms', *Economic Theory* 7, 237-266.
- Reiter, S. (1995). 'Coordination and the structure of firms', Center for Mathematical Studies in Economics and Management Science Discussion Paper 1121, Northwestern University.
- Reiter, S. and G. Sherman. (1962). 'Allocating indivisible resources affording external economies or diseconomies', *International Economic Review* 3, 108-135.
- Reiter, S. and G. Sherman. (1965). 'Discrete optimizing', *Journal of SIAM* 13, 864-887.