Asynchronous Teams

S. Talukdar

EDRC 18-42-93

ASYNCHRONOUS TEAMS

Sarosh Talukdar

Engineering Design Research Center
Hammerschlag Hall 2221 >
Carnegie Mellon University
Pittsburgh, PA 15213
Ph: (412) 268-8778
Fax: (412) 268-5229

Fourth International Symposium on Expert Systems Application to Power Systems, La Trobe University, Melbourne, Australia, January 4-8, 1993.

ASYNCHRONOUS TEAMS

Sarosh Talukdar
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

This paper has three purposes: first, to define a space of organizations for computer-based agents; second, to identify a fuzzy subset (called A3◁) of this space that contains organizations that are open, parallel and effective; and third, to illustrate how members (called asynchronous teams) of this subset can be realized. These teams are characterized by autonomous agents that work in parallel on populations of solutions. As such, they have features in common with a number of well known organizations. Blackboards, parallel implementations of genetic algorithms and certain recurrent neural nets are examples. But these examples occur at the very outermost fringes of the A3◁ subset. Its interior remains unexplored. We begin that exploration here.

Key-words: multiple agents, machine cooperation, distributed artificial intelligence, organizations.

1. INTRODUCTION

Loosely speaking, an organization is a scheme by which objects, such as birds, fish and humans, combine to form super-objects, such as flocks of birds, schools of fish and corporations. Functionally, an organization determines the manner in which its objects cooperate; whether they help or hinder one another in their work. Structurally, an organization can be thought of in terms of certain flows and constraints whose description is the first purpose of this paper. The second purpose is to identify a class of organizations that is well suited to software objects, such as simulation and optimization programs. Until recently, it was so horribly difficult to interconnect large programs that the builders of software systems had little time to bother with organization issues. Instead, programs were usually connected in series with the output of one becoming the input of another. Such serial organizations tend to be inflexible, difficult to maintain, and expensive to update.

The technology of non-serial organizations is in a primitive state: there are few widely accepted terms for describing their structures, even fewer models for visualizing these structures and virtually no systematic techniques for designing them. It will be several years before these deficits can be made good. The material in this paper is a modest start.

2. SOME ATTRIBUTES OF MACHINE COOPERATION

Our concern here is with mechanisms by which software objects can cooperate. Structurally, a software object is a body of code that has been encapsulated to distinguish it from other objects. Functionally, a software object may serve the purposes of storing data of processing (transforming) data. Most actual objects do some of each. Nevertheless, we will divide them into two fuzzy categories: data-objects and agents. A data-object is one whose primary function is to store data. All other objects are agents. Thus, data-objects appear from the outside to be passive, but may contain many procedures to store, retrieve, or even generate data; agents are outwardly active, though they may contain vast databases.

A super-object composed from software objects, is said to be distributable if it fits well in a network of computers, that is, if it allows its agents to work in parallel, if each agent uses only locally available data, and if the super-object's performance is relatively insensitive to communication delays.

2.1 Scale Efficiency

Consider for a moment a natural organization: any one of the 10,000 or so species of fish that form schools, say Silversides. Such schools grow easily; any Silverside that wishes can join. One of the bene-

nents: operators and controllers. The operators make transformations, the controllers decide: (a) what the operators will transform (select inputs), (b) when they will transform these inputs (schedule activities), and (c) what computers will be used (allocate resources).

Think of an agent as an aggregation of processes, one being an operator and the others being con* trollers for itself or other agents. If the agent contains controllers sufficient for making all its own decisions (for input selection, scheduling and resource allocation), then it is "autonomous." Otherwise, it is "supervised" and its supervisors are indicated by broken arrows in a directed graph whose nodes are agents and whose edges are supervisory relations. Such a graph will be called a Cf or control flow.

Let S(Df) be the space of all control flows. As before, we recognize two fuzzy subsets of this space: 4>(Cf), the subset of near-null control flows, and A(Cf), the subset of strongly cyclic control flows. A near-null control flow means there are virtually no supervisors; most, if not all, the agents are autonomous. Strongly cyclic control flows are, at the very least, unusual; I cannot think of an organization that uses them. But their complements, strongly acyclic control flows, are the hierarchical arrangements so common in human organizations.

3.4 State Spaces and Markings

Usually, a state space is defined so that each of its points represents one state of the system under consideration. Changes in state (dynamics) trace trajectories through such spaces. Here, we will also use a different and less condensed version of a state space-instead of a single point, each state will be denoted by a set or population of points, called a marking. More specifically, each shared memory is assigned its own state space. Each point in this space represents one data-object while the space as a whole represents all the objects allowed into the memory. Let Sj be such a space for the i-th memory and let Mj(t) be the subset of Sj that represents the data-objects that are actually in the i-th memory at time t. Then changes in state (dynamics) will cause changes in the shape of this marking. We will call the trace of such changes a state flow.

3.5 Activity Constraints

Consider the j-th agent. It, like every other agent, is capable of three types of action: reading, writing and computing. In general, it cannot take these actions whenever it wishes, but only when its supervisors and the contents (markings) of its input memories allow them. Thus, even an autonomous agent is not without external constraints on its actions. For instance, it may have to wait till certain information appears before it can start computing, and may have to stop when some essential piece of information is removed. The set of all such constraints for all the agents in a super-object will be called an Ac or set of activity constraints.

Let S(Ac) be the space of all Ac. We recognize one fuzzy subset of this space: <$>(Ac), the subset of "near-empty" Ac. An Ac is near-empty if its constraints are so loose or so few that they have little influence on when the associated agents read, write or compute. More specifically, if an Ac allows virtually all communications to be asynchronous (so no agent ever needs to interrupt its computing in order to wait for a communication) and if virtually all the agents compute when they wish (so they can work in parallel all the time if they so choose), then the Ac is near-empty.

3.6 **Modification** Constraints

Consider a super-object. If the structural changes required to insert or delete an object are confined to the object itself, then the effort involved is likely to be relatively low. In contrast, if the entire super-object has to be modified, the effort can be expected to be high.

Let Me be a set of constraints that must be satisfied in order to insert or delete an object from some super-object. An Me will be called near-empty if it requires only modifications to the object being added or deleted, and moreover, this object uses only locally available information. Let S(Mc) be the space of all Me and <(>(Mc) be a fuzzy subset of this space that contains Me that are near-empty.

3.7 A Structural Model

The upshot of the preceding material is that the structure of an organization can be thought of as a quadruple: 9 = (Df, Cf, Ac, Me) where Df is the organization's data flow, Cf is its control flow, Ac is its set of activity constraints, and Me is its set of modification constraints.

4. AN ORGANIZATION SPACE: S(6)

Let S(0) be the space of all 6. In other words:

S(9) = S(Df) X S(Cf) X S(Ae) X S(Mc).

One use for S(8) is in designing organizations. Suppose that the specifications of an organization are given. Then, one could search S(0) for entries that meet these specifications. Of course, it would be reasonable to first ask: for what sorts of problems does S(8) contain good solutions? are there some regions of S(9) that contain better solutions than others?

4.1 Limitations

Recall that 6, our model of an organization, represents communication systems by sets of shared memories. While this representation is adequate for computer-based organizations, it is probably inadequate for human organizations. Also, 6 is a static model; it provides a snapshot of an organization's

iteration are possible) and autonomous agents (that is, an organization with no supervisors, and more important, no hierarchy through which the errors and limitations of supervisors can be magnified). The subsequent material deals with members of the [A, $, ⇄] region which, for the purposes of brevity, will sometimes be called the A3⇄ region.

5.1 Definition

An A-Team (asynchronous team) is any member of the region: [A, 4, ⇄, $]. In other words, an A-Team is an organization whose structure is as follows:

- Df, its data flow is strongly cyclic, meaning that its agents can use feedback and iteration in developing solutions.
- Cf, its control flow, is near-null, meaning that virtually all its agents are autonomous.
- Ac, its set of activity constraints, is near-empty, meaning that its agents are free to act and interact when they wish. In particular, all of them can work in parallel all the time. Further, there is no predetermined schedule for exchanges of information; rather, exchanges are allowed to occur asynchronously (spontaneously).
- Me, its set of modification constraints, is near-empty, meaning that the addition of an agent requires only modifications to the agent, not to the rest of the organization. Also, all agents use only locally available data. Therefore, the addition of an agent requires relatively little effort and the organization is comparatively open.

5.2 Two Population Control Strategies

Clearly, the structure of an A-Team allows for anarchic behavior. Autonomous agents, each choosing what to do and when, if ever, to communicate with its team mates, can act at cross purposes. Surprisingly, there are simple strategies, not only to prevent this from happening, but to produce forms of cooperation so effective that they approach synergy. Two of these strategies are "herding" and "consensus-seeking." Both are essentially strategies for controlling the populations of data-objects produced by the agents in an A-Team. These agents can be divided into four categories:

• construction agents that produce new data-objects from old ones by adding information (the old objects may or may not be erased),
• modification agents that produce new data-objects by changing the information in old ones (the old objects may or may not be erased),
• deconstruction agents that produce new data-objects by deleting information in old ones (the old objects may or may not be erased), and
• destruction agents or destroyers that erase data-objects.

Consider an A-Team of construction, modification and deconstruction agents that do not erase any data-objects. If these agents work continuously and in parallel, as the A-Team structure allows them to do, the population of data-objects would explode. There are two ways to keep this from happening. The first is introduce a set of destroyers; the second, is to change the agents so they do their own erasing. "Herding" is a strategy of the first type. The destroyers are designed so they keep the population in check and also cause it to move along paths that eventually lead to profitable conclusions. "Consensus-seeking" is a strategy of the second type. When the population of data-objects has become large, most of the agents change mode from producing objects to erasing them till only a few objects remain.

Recall that the data-objects in each memory are represented by a set of points in a state space, and this set is called a marking. In "herding," the size of the marking usually expands until it reaches an upper limit after which it remains more or less constant as the marking moves about in the state space. In "consensus-seeking" the dynamics are quite different: the marking expands and then contracts rapidly.

Some natural and synthetic examples of herding and consensus-seeking are given below.

6. TWO NATURAL A-TEAMS

6.1 Lamellar Bone Growth: An Example Of Herding

Osteoblasts and osteoclasts are simple cells that work on long bones [3], [4]. They react to the stresses in the bones, averaged over the last **few** weeks. The osteoclasts converge on faces **that have** been under low stress or in tension and **proceed** to dissolve the bone there. The osteoblasts **add** bone to faces that have been under high compressive stress. The net effect is to optimize the shape and mass of the bone for the average load to which it has been exposed [3].

Clearly, the osteoblasts and osteoclasts form an A-Team: they work iteratively, are autonomous, decide for themselves when and what they will do, base these decisions on locally available information, carry out the decisions in parallel, communicate asynchronously through the results of their work, and finally, can join or leave the team at will.

Think of each potential addition or deletion of a piece of bone as an object and the set of all these objects as the state space for the osteoblasts and osteoclasts. Recall that a marking is the subset of **the** objects that actually exists at time t (that is, **the** actual bone). Under the object-creating actions of **the** osteoblasts and the object-destroying actions of the osteoclasts, the marking changes to adapt to its average historical load. It is as if **the** marking **were** being herded through state space by the osteoblasts and osteoclasts. When they achieve optimal results it is through the addition of bone exactly **where** it **was** most needed and the deletion of bone from **where** it was least used. If the balance between these processes is disturbed, as would happen if there were

full-fledged membership by their agents which are seldom autonomous in the sense used here. Neural nets are disqualified unless they are self-training, that is, unless they require little retraining when neurons are added or removed.

In 1983, Sam Pyo, Nino Vidovic and I developed an A-Team for solving sets of nonlinear algebraic equations [16]. This team used consensus-seeking as its method of cooperation. Since then teams have been built that rely more heavily on the herding strategy and solve six important and difficult types of problem: large travelling salesman problems [15], configuring task-specific robots [3], high-rise building design [17], the real-time control of electric networks [18], the diagnosis of faults in power systems [19], and multi-objective train scheduling. The teams for the first two types of problem are fairly mature and have been extensively tested. In both speed and solution quality they are clearly superior to standard techniques. The teams for the the other problem types are less mature and have, as yet, only been tested on small problems. However, it seems that they will also out perform standard methods.

In the succeeding material, I will briefly describe the two mature teams.

7.1 The **Travelling** Salesman **Problem (TSP): An Example of** Scale Efficiency

Given the locations of a number of cities, the TSP is to find a closed tour of minimum length that passes through every city once. Because this problem is NP-hard, algorithms that find optimal solutions in polynomial time are unlikely to exist. However, there are many polynomial-time heuristics that can find near optimal solutions. These heuristics are of two types: construction heuristics that transform partial tours (including the null tour) into complete tours, and modification heuristics that make changes in complete tours. By adding heuristics for deconstruction (breaking complete tours into partial tours) and destruction (eliminating complete and partial tours) one obtains the basis for a set of agents from which A-Teams can be assembled. More specifically, each heuristic is encoded to form an autonomous agent able to read from, and write to, memories containing partial and complete tours, as appropriate. All the agents work in parallel. The construction and modification agents add new tours to the complete-tour-memory. The deconstructors add new partial tours to the partial-tour-memory. The destroyers limit the populations in both memories by eliminating their weakest members. The other agents select their inputs randomly from their input populations. Souza has shown [15] that such teams can be scale efficient. Specifically, the speed and quality of the results produced by the team increases as more construction and modification agents are added. For instance, with a difficult 532 city problem and random starting points, the team usually finds the optimal solution in an hour or two. Working alone from the same starting points, its construction and modification agents take much longer, virtually never finding the optimal solution, and often finding quite inferior solutions. Further details can be found in [16].

7.2 Robots-on-Demand (ROD): An Example of Synergy

The ROD problem is: given a bin of robot parts, a robot task expressed as a set of kinematic and dynamic constraints, and a set of objectives (such as minimizing robot weight and deflection), find a good, if not optimal, robot design. The intent is to assemble a robot that is specialized for the task to be performed, and to disassemble this robot when the task is completed so the parts can be reused in another robot for the next task.

Sesh Murthy [3] has built an A-Team for designing three-segment robots for certain ROD problems. Each design specifies the values of twelve variables. Three of these variables are continuous and locate the base of the robot. The other nine are discrete and identify the motors, joints and links that constitute the robot's segments. Besides meeting the task-specific kinematic and dynamic constraints, three criteria are taken into account: minimizing weight, minimizing deflection and maximizing dexterity.

The team contains a single memory, 55 very simple and fairly dumb modification agents, and about 12 equally simple and dumb destroyers. The objects stored in the memory are robot designs. Each modification agent acts on one of the twelve variables that constitute a design and takes into account only one criterion. As a result, its actions usually produce improvements in this criterion but, as often as not, are accompanied by large degradations in other criteria. For instance, one of the agents attempts to reduce kinematic constraint violations by replacing the link in the robot's third segment with the next longer link from the bin of parts. Invariably, this replacement causes large degradations in deflection and dynamic constraint violations.

The little intelligence in each modification operator is concentrated in its input controller. This controller selects the next design for the agent to modify by performing a qualitative match of the design's needs and the agent's capabilities. In essence, the design is selected if it needs the improvement the agent offers and can afford the price (the accompanying degradations in other criteria).

Each destroyer also considers only a single criterion and its intelligence is also concentrated in its input controller. The selection process involves two steps. First the candidate design is tested with respect to the destroyer's criterion. If it fails, then the destroyer proceeds to the second step: flipping a coin that is biased so "tails" is more likely for designs that have failed badly. If the coin comes up "tails," the design is eliminated, most of the destroyer use obvious criteria, such as kinematic-constraint-violations. However, one of them uses an interesting criterion: distance

X|<+1 requires the application of every 0_n to every point in X|<. The result is a state flow that expands quickly to a considerable width before it begins to narrow as points are repeated. The reason for using destroyers and mechanisms such "consensus-seeking" is to narrow the state flow of (P3) at its widest part without significantly changing its termination.

9. SUMMARY

The space of computer-based organizations, S(<t>), can be partitioned into a number of fuzzy regions. One of these regions, A3<t, is notable for containing organizations which are open, parallel and distributable. These organizations, called A-Teams, use only autonomous agents.

Applications of A-Teams in a variety of domains have demonstrated that they can be extremely effective in solving very difficult problems. Moreover, this effectiveness can be achieved by fairly simple strategies. Two of these strategies are called herding and consensus-seeking. The former relies on a balance between agents that create solutions and agents that destroy them. The latter relies on social instincts (rules) implanted in each agent that cause it to align its approach with that of its most successful neighbor.

No doubt, there are many other strategies by which A-Teams can be made effective at problem solving. Rather than search for these strategies, my colleagues and I have been concentrating on the herding strategy and in particular, on technologies for deconstruction and destruction. I believe that in tackling difficult problems, the deconstruction of bad alternatives to recover their reusable parts and the destruction of hopeless alternatives to prevent their further consideration, are at least as important as the creation of new alternatives through construction and modification processes. However, much more is known about construction and modification processes than deconstruction and destruction processes. Therefore, deconstruction and destruction would seem to be fertile areas for research.

10. REFERENCES

[1] Partridge, B.L., "Structure and Function of Fish Schools," (Scientific American, vol. 24, no. 6, 1982, pp. 100-109.)

[2] Talukdar, S., Ramesh, V.C., Quadrel, R. and Christie, R., "Multi-agent Organizations for Real-Time Operations," (Proceedings of the IEEE, vol. 80, no. 5, May 1992.)

[3] Murthy, S., "Synergy in Cooperating Agents: Designing Manipulators from Task Specifications," (Ph.D. Dissertation, Carnegie Mellon University, September 1992.)

[4] Frost, H.M., 'The Laws of Bone Structure," (Charles C. Thomas, Springfield, Illinois, 1964.)

[5] Eberhard, M.J.W., "The social biology of polistine wasps," (Miscellaneous Publications, Museum of Zoology, Univ. of Michigan, 1969, vol. 140, pp. 1-101.)

[6] Rettenmeyer, C.W., "Behavioral studies of army ants," (Kansas Univ. Sci. Bulletin, 1963, vol. 44(9), pp. 281-465.)

[7] Wilson, E.O., "The Insect Societies", (Belnap Press of Harvard Univ.: Cambridge, MA, 1971.)

[8] Kalmus, H, "Vorversuche ũber die Orientierung der Biene im Stock," (Zeitschrift fũr Vergleichende Physiologie, 1937, vol. 24(2), pp. 166-187.)

[9] Weiss, B.A., and Schneirla, T.C., "Inter-situational transfer in the ant *Formica schaufussi* as tested in a two-phase single choice-point maze," (Behaviour, 1967, vol. 28(3-4), pp. 269-279.)

[10] Sudd, J. H., "How insects work in groups," (Discovery, London, June 1963, pp. 15-19.)

[11] Lindauer, M., "Ein Beitrag zur Frage der Arbeitsteilung im Bienenstaat," (Zeitschrift fũr Vergleichende Physiologie, 1952, vol. 34(4), pp. 299-345.)

[12] Emery, C, "Le polymorphisme des fourmis et la castration alimentaire," (Compte Rendu des Séances du Troisième Congrès International de Zoologie, Leyde, 1895, pp. 395-410.)

[13] Wilson, E.O., "Chemical communication among workers of the fire ant *Solenopsis saevissima,"* (Animal Behaviour, 1962, vol. 10(1-2), pp. 134-164.)

[14] Grassé, P.P., "Nouvelle expériences sur le termite de Mũller *(macrotermes mũlleri)* et considérations sur la théorie de la stigmergie," (Insectes Sociaux, 1967, vol. 14(1), pp. 73-102.)

[15] Talukdar, S.N., and Souza, P.S. de, "Scale Efficient Organizations," (Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernetics, Chicago, Illnois, Oct. 18-21, 1992.)

[16] Talukdar, S.N., Pyo, S.S. and Mehrotra, R., "Distributed Processors for Numerically Intense Problems", (Final Report for EPRI Project, RP 1764-3, March 1983.)

[17] Quadrel, R., "Asynchronous Design Environment: Architecture and Behavior", (Ph.D. Dissertation, Carnegie Mellon University, 1991.)

[18] Chen, C.L. and Talukdar, S.N., "Causal Nets for Fault Diagnosis", (4th International Conference on Expert Systems Application to Power Systems, Melbourne, Australia, Jan 4-8, 1993.)

[19] Talukdar, S.N.and Ramesh, V.C., "Cooperative Methods for Security-Planning", (4th International Conference on Expert Systems Application to Power Systems, Melbourne, Australia, Jan 4-8,1993.)