Dilemmas in Computational Societies

Natalie S. Glance and Tad Hogg

Rank Xerox Research Centre
6 chemin de Maupertuis
38240 Meylan, France
glance@xerox.fr

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, U.S.A.
hogg@parc.xerox.com

Abstract
World-wide interlinked computer networks are forming the foundation for computational societies of software agents. Already, these new societies have encountered problems endemic to human communities, such as overusing common resources with thrashing over virtual memory and competition by packets for network time. Unlike with human societies, these inefficiencies can be overcome by re-working the algorithms governing the protocols. However, the public good problem, in which a common good is available to all regardless of contribution, can arise computationally in more subtle ways. We discuss how this can happen using Braess’ Paradox and demonstrate that adding resources to a computational system can counterintuitively lower the overall performance. This is thus a case in which distributed algorithms are provably unable to achieve globally optimal performance. We illustrate our claim using a genetic algorithm and computational ecosystem.

Introduction
We are in the midst of constructing vast societies of computational agents [18, 9, 34, 35, 13]. These agents often act autonomously using locally available information. Often, such decentralized systems behave more robustly, by responding rapidly to changes and performing well, than traditional ones based on a single central controller. They are also simpler to design and incrementally modify. On the other hand, these agents, like people in human societies, face the difficult task of performing well in spite of incomplete, imperfect, and changing information about their environment. What lessons might human societies offer designers of computational ones?

For example, the provision of public goods causes difficulties for autonomous agents [12]. These goods are available to all, regardless of whether an individual contributes (“cooperates”) or not (“defects”). Common examples are national defense and cleaning up the environment. Specifically, the dilemma arises when

\[ B > c > b \]  

(1)

where \( B \) is the benefit to the entire group of an additional individual’s contribution, \( c \) is the cost to the individual of cooperating, and \( b \) is the individual benefit received from cooperating. Agents in this situation are tempted to defect since their cost for cooperating exceeds the benefit they receive. When everyonereasons similarly no public good is produced, and everyone is worse off because the overall social benefit of cooperation exceeds the combined costs. Thus, individually rational behavior leads to an overall suboptimal result. Human societies often address this problem with mechanisms to enforce contribution, such as taxation, or the creation of new markets which at least partially restrict benefits to those that contribute. Cooperation can also be maintained if the benefits are concentrated in small groups with long-range plans [10] or if interactions are repeated [1, 2]. There are also a variety of mechanisms to evolve cooperation in such cases [30].

The nature of the computational society determines whether such dilemmas can be avoided. At one extreme, the agents perform tasks for a variety of users with differing goals, ranging from communicating over the network to sharing available workstations for distributed processing to searching a database. In this case, the dilemma can be avoided by converting public goods to private ones via simple computational pricing and accounting mechanisms [24] coupled with adequate security to prevent cheating by actively malicious programs such as computer viruses.

At the other extreme are societies of agents all designed with a single goal in mind, e.g., solving a problem via parallel processing. Naturally, all the agents benefit by producing the desired final solution. While this sharing of benefits creates an opportunity for a social dilemma, one might expect that it could be avoided by simply programming in virtues that are lauded but under-represented in human history: altruism, cooperation, and rationality. Thus, public goods seem an excellent example of a problem which should not exist in societies of computational agents that are designed to address a single problem.

However, the public good problem is as much a computational dilemma as a social one. It is, on the one
hand, a concrete example of the general mathematical fact that locally optimal actions often result in suboptimal global performance. At the same time, determining the globally superior choice by central fiat may be computationally intractable. In some cases, the overall minimum may actually be provably impossible to find within a distributed setting since individual choices will take the system away from the minimum instead of towards it — this is the computational analogue of a social dilemma. Computational social dilemmas thus stand in contrast to other hard problems whose hilly solution landscapes cause methods such as neural nets [26], simulated annealing [21] and greedy heuristics for NP-hard search problems [8] to become trapped in local minima, making the overall minimum hard, but not impossible, to find.

This discussion raises the following issues: in the absence of central control, what forms do public goods problems take, how can they be recognized, and how might they be resolved. In this paper we consider these questions in the context of Braess’ Paradox, described in the next section, which reveals how social dilemmas can arise in purely computational domains. The consequences are then illustrated with a genetic algorithm [16, 6] and computational ecosystem [18].

**Braess’ Paradox**

A particularly subtle version of the social dilemma is Braess’ paradox [4, 19] in which adding resources to a system can actually lower performance. Fig. 1 shows a simple example. Agents enter the network from the bottom and choose among various paths, moving between nodes along the indicated links. The traffic is described by the fraction of agents taking each link, and the cost for moving on a link can depend on that link’s flow. Intuition suggests that adding a new link between nodes B and C would always improve performance. However, the following argument shows this is not always the case.

At equilibrium the flow through the network is distributed so that each agent incurs the same cost and so that any change by one agent would increase its cost, given that none of the other agents change their routes. The overall equilibrium is reached by local decisions in which each agent minimizes its own cost given the behavior of the other agents. When the link between nodes B and C is absent, half the agents choose the left-hand path and half choose the right-hand path. This results in optimal flow: the average cost per agent is minimized and equals 1.25.

When the horizontal link with cost $1/4 < x < 1/2$ is present, an agent traveling from node A to D now has four paths to choose from (instead of only two). The most expensive of these is path ACBD which costs $c_{ACBD} = 2 + x$ and is consequently taken only in error. Path ABD costs the agent $c_{ABD} = 1 + f_1/2$ (where $f_1$ is the fraction of agents following arc AB), path ACD costs $c_{ACD} = 1 + f_2/2$ (where $f_2$ is the fraction following arc CD) and path ABCD costs $c_{ABCD} = f_1/2 + x + f_2/2$. The overall global average cost is then given by the sum of costs on each path weighted by the fraction of agents following that path:

$$C_{global} = (f_1 - f_x) c_{ABD} + (1 - f_1) c_{ACD} + f_x c_{ABCD} = \left(\frac{3}{2} - f_1 + f_1^2\right) + f_2^2 + f_x(x - f_1)$$

(2)

Given perfect information, an agent will always prefer path ABCD to the other two since $f_1/2 + x < 1$ and $f_2/2 + x < 1$ independent of the congestion levels $f_1$ and $f_2$. Thus, when each agent minimizes its own costs, they all take path ABCD, giving $f_1 = f_x = 1$ and an average cost of $1 + x$ which is greater than the average cost of 1.25 without the link. Thus when agents individually minimize cost, additional resources counterintuitively lead to lower overall performance, as illustrated in Fig. 2. The global minimum of Eq. 2 occurs at $f_1 = 1 - x$, $f_x = 1 - 2x$ giving

$$C_{min} = 1 + x - x^2$$

(3)

Braess’ Paradox can be readily mapped onto the social dilemma of Eq. 1. First, we map the average transit cost per agent into overall utility: $U_{total} = -n C_{global}$, where $n$ is the number of agents flowing
through the system. Then we associate the change in transit cost due to an agent’s path choice with the cost for cooperation and the change in average transit cost per agent with an increase or decrease in utility. We will show that the system can benefit overall when the agent picks a path with higher transit cost. The change in total utility will then reflect the sum of the global benefit and the individual’s cost due to cooperation:

$$\Delta U_{total} = B - c$$  \hspace{1cm} (4)

Consider the case \( f_x > f_1 - x \), where \( f_x \) is the fraction of agents traveling along arc BC (an analogous construction exists for \( f_x < f_1 - x \)). If we associate cooperation with the individual action that optimizes global performance and defection with the action that optimizes local performance at the expense of the global good, then, in this case, an agent at node B cooperates by following arc BD and defects by following arc BC. The cost \( c \) to the agent for cooperating is the difference between the costs of paths ABD and ABCD, using \( f_x = 1 - f_1 + f_x \):

$$c = \frac{f_1}{2} - \frac{f_x}{2} + \frac{1}{2} - x.$$  \hspace{1cm} (5)

Since \( f_x \leq f_1 \) and \( x \leq 1/2 \), the cost of cooperation is always positive, making cooperation a sacrificial move for the agent. On the other hand, if the agent does cooperate (incurring cost \( c \)), then the average transit cost per agent goes down, increasing overall utility

$$\Delta U_{total} \approx -n \frac{dC_{global}}{df_x} \left( \frac{-1}{n} \right) = -f_1 + f_x + x$$  \hspace{1cm} (6)

ignoring terms of order \( 1/n \), because when the agent cooperates, it reduces \( f_x \) by \( 1/n \). From Eq. 4, the global benefit \( B \) is then given by

$$B \equiv \Delta U_{total} + c = -\frac{f_1}{2} + \frac{f_x}{2} + \frac{1}{2}$$  \hspace{1cm} (7)

In this case, the individual benefit \( b \) of cooperation is zero. Since \( f_x > f_1 - x \), we get \( B > c > b \) so the new link creates a social dilemma.

The difference in average cost per agent due to individual actions can also be understood using Fig. 3 for \( x = 0.5 \). Individual defections move the system to increase \( f_x \), which reduces an individual’s cost but always increases total cost when \( f_x > f_1 - x \). If agents individually optimize their paths the system moves away from the global minimum \( f_1 = f_2 = \frac{1}{2}, f_x = 0 \) to the clearly inferior state \( f_1 = f_x = 1 \) with each agent choosing to follow the new link between nodes B and C. Thus, the distributed decision-making of the agents moving through the network by optimizing individual preferences takes the system far from the optimal overall state which an omniscient central controller would readily locate.

**Computational Examples**

**Genetic Algorithms**

To see how this paradox can affect a computational society, we use a genetic algorithm (GA) \([16, 6]\). This is a search method based on analogy with biological evolution. GA’s breed collections of structures (often represented by bit strings) to find those that minimize a given cost function, or, equivalently, maximize a fitness function. Specifically, we use a GA to solve the complete network flow problem of Fig. 4 in which Braess’
Fig. 4. A larger network embedding Braess’ paradox. Alternative choices and their costs (e.g., traffic delay) for flow through the network from bottom to top are indicated. The boxed region incorporates the paradox, giving lower throughput with the horizontal link than without it when agents choose the path that minimizes their individual cost. Decision points are marked by the circles.

Paradox (Fig. 1) is embedded. In our example, the structures manipulated by the GA encode the paths taken by the agents: one structure per agent. The GA evolves the collection of structures in a direction that tends to decrease the individual transit cost for the agents. Since Braess’ Paradox is embedded within the network, including the extra link BC should result in decreased global performance when the agents make local decisions.

Fig. 5. Behavior of genetic algorithm for a case without dilemma (top) and with dilemma (bottom). The cost of the extra link in the network of Fig. 1, present in the dilemma case, was set to $x = 0.5$. With or without the dilemma, the cheapest path through the network costs 4.25, but this solution is found only when the extra link is not present. The plots show the transit costs through the network as a function of generation number (in tens). The points give the average cost while the bars indicate the range of costs. We used a population of size 10, with crossover rate 0.5, mutation rate 0.01, and a generation gap of 0.1 (i.e., exactly one agent bred per generation). Six bits were used to represent the path taken by each agent through the network. A publicly available genetic algorithm implementation [11] was used to conduct the experiments. In each generation exactly one agent is bred, corresponding to asynchronous choices by the agents. Thus the entire population of ten agents reproduces once every ten generations.

Typical results are shown in Fig. 5: with the additional link the GA is unable to find the optimal state. To show that this was not exceptional, we repeated this experiment 100 times running the GA 2000 generations each time, and averaged the cost over all the agents for generations 1500 to 2000. The extra link always lowered the performance of the GA even though the cheapest path through the network costs 4.25 per agent with or without the extra link.
Excluding two experiments which had not converged to their final configuration by generation 1500, the mean cost for the case without the extra link (no dilemma) was 4.275 with standard deviation 0.0076. Adding the link increased the average cost to 4.51 with standard deviation 0.0056. (The average costs are higher than predicted by the cost function of Fig. 2 — 4.25 without the link, 4.5 with the link — because mutations introduce fluctuations which always increase cost once the population has converged.) Thus, the additional link on average increased the average cost by 0.235, almost the full 0.25 predicted by Braess’ paradox. The difference is somewhat smaller than expected because the mutation-caused fluctuations induce more variation in the no dilemma case, since the average fitness is then very sensitive to the distribution of agents among the left and ride-hand paths within the boxed-in region of the network.

We should note that, if one did not recognize the existence of the dilemma in this situation, the lowered performance could be easily misinterpreted as resulting from random variations in the population of agents. In this case one might be tempted to continue the runs hoping for eventual improvement or else conclude that the decreased performance results from a local minimum and hope that restarting from a new initial condition would result in a better final result. For this case, containing the social dilemma, neither of these common reactions would in fact be beneficial. In these cases, social dilemmas offer an alternative perspective on why problems may be difficult for GA’s to solve [7].

The results would be very different if the GA bred structures that solved the global problem of finding a set of structures to minimize the total cost to the whole group of agents. However, the GA would then be solving the problem of finding the optimal strategy for a central control mechanism. Furthermore, such an approach is possible only if the overall cost function is known, which will not generally be the case in a distributed setting.

**Computational Ecosystems**

An additional illustration of the effect of Braess’ Paradox is given by computational ecosystems [18]. These consist of agents, with computational tasks to perform, and various resources with which to accomplish their tasks. The benefits associated with the resources are computational measures of performance, such as the time to complete a task or accuracy of the solution. The suitability of a resource can depend on how many other agents are using it. For instance, additional load on a machine can reduce the execution speed of each task on that machine. Alternatively, agents using a resource could assist one another in their computations. For example, if the overall task could be decomposed into many communicating subtasks, the agents may be better off using the same computer rather than running more rapidly on separate machines and then being limited by slow communications. When the system is large and the information on the suitability of various resources is imperfect, delayed and changing it is not feasible to use a central agent to allocate resources. Instead, flexibility and responsiveness to local changes requires that autonomous agents asynchronously select among available resources. A simple model of these societies assumes each agent uses one resource at a time, evaluates its choice at a rate α and selects that resource it perceives to be best. These choices lead to a range of dynamical behaviors including simple equilibria, continual oscillations and chaos.

To see the consequence of Braess’ Paradox, we can use the costs of the various paths of Fig. 1 as the costs of resources in a computational ecosystem, i.e., use resources 1, 2 and 3 whose costs correspond to paths ABC, ACD and ABCD respectively. Let $F_i$ be the fraction of agents using resource $i$ and $C_i$ the respective cost. These are related to the fractions using the various links in Fig. 1 by $F_1 = f_1 - f_2$, $F_2 = 1 - f_1$ and $F_3 = f_2$, which are also the corresponding fractions for each path as used in Eq. 2. Since $F_1 + F_2 + F_3 = 1$ we can express the costs using only $F_1$ and $F_2$ as

$$C_1 = \frac{1}{2}(3 - F_2)$$

$$C_2 = \frac{1}{2}(3 - F_1)$$

$$C_3 = 1 + \frac{1}{2}(F_1 + F_2)$$

and the overall average cost for the system is given by $C = \sum F_i C_i$. Let $\rho_i$ be the probability that resource $i$ is perceived to be the best one when an agent makes a choice. Then $\sum \rho_i = 1$ and the $\rho_i$ depend on the $F_i$ because the resource costs do. The dynamical behavior is then given by [18]

$$\frac{dF_i}{dt} = -\alpha(F_i - \rho_i)$$

The simplest case is when the agents have full information on their individual costs and there are no delays in the information. In that case they will always select that resource with the lowest cost giving $\rho_1 = 1$ when $C_1$ is the lowest cost, and zero otherwise. When there are only two resources (or equivalently $x$ is so large that resource 3 is never selected), $\rho_1 = 1$ when $C_1 < C_2$, corresponding to $F_1 < F_2$, and similarly $\rho_2 = 1$ when $F_1 > F_2$. Thus, starting from an initial condition of all agents using resource 1, i.e., $F_1 = 1$,
we have $F_1(t) = e^{-\alpha t}$ and $F_2(t) = 1 - F_1(t)$ until these values reach 1/2, at which point they remain at that value. When the third resource is added, each agent will find it has a lower cost and select it, i.e., $\rho_3 = 1$. Thus $F_1$ and $F_2$ will decay exponentially toward zero. This behavior is illustrated in Fig. 6, along with its consequence for the overall performance.

While previous studies of the 3-resource case have shown how oscillations and chaotic behaviors can lower performance in some cases [20], we now see that public goods problems can also arise. Since this will not always be readily apparent from the cost functions [14], especially in more complex cases that include nonlinearities, it may be easy to confuse the lowered performance due to a social dilemma with that due to uncertainty or delays in the available information.

**Discussion**

In summary, additional resources can paradoxically decrease overall performance when the following basic ingredients exist: locally optimized choices, the sharing of benefits among the agents, and resources whose costs depend on the level of use. In addition to the illustrations we presented, there are a variety of other computational societies with a single overall goal that have these ingredients.

Coevolutionary systems, where the costs facing an individual depend inherently on the behavior of others, are an example. These systems arise naturally in biological and economic contexts and are increasingly used in the computational arena. For example, GA's can help a robot learn in specific environments through the evolution of a set of behavioral rules [32]. When a distributed GA is applied to a society of robots [17, 27], the process becomes coevolutionary. For instance, the robots might need to learn strategies for dividing tasks among themselves, so that all of the robots do not flock to the highest priority task.

Another example is cooperative problem solving in which information obtained by an agent working on one part of the problem is passed to others [3, 15, 5]. For instance, the crossover operator of GA's, which takes two "parent" agents and recombines them to produce new agents, can be interpreted as an exchange of hints. Hints can be very useful in cooperative problem solving, but often require agents to "explore" possibilities that do not directly produce solutions in order to obtain useful hints. Moreover, the benefit of a new hint depends on which other hints have already been uncovered. In such cases, the benefit of an agent's choices depends on those made by others.

These problem solvers can also include learning mechanisms such as chunking [22] which automatically add extra links in a problem space [25] representing the newly learned operations. While this often improves performance [29], density-dependence can arise if the choice of which link to follow in a particular instance is made based on how useful it was in the past as measured by the number of other agents that used it. In this situation, a dilemma could arise during learning, resulting in unexpected decrease in performance.

A computational ecology example is given by recent proposals for flexible manufacturing [33]. These introduce distributed decision-making using auctions to allocate tasks to machines. In these cases agents must determine the best machine to use, given the nature of its task and the current load on the machines.

Finally, density-dependence might be intentionally introduced to improve a computational method. For instance, many search algorithms face the problem of premature convergence to a single state of low, but not optimal, cost. To avoid this, steps are taken to maintain a diverse set of agents. In GA's, this
can be done by increasing the mutation rate, but this random variation also destroys useful structures. More sophisticated methods for GA's [23, 31] and other methods [28] rely on explicitly keeping the agents apart in solution space. This can be viewed as a density-dependent modification to the cost function, i.e., the desirability for a state depends on how many other agents are already "near" it. In this case, the cost function is defined by the configuration of the agents themselves, giving a coevolutionary system.

These examples show that density-dependence can arise from the nature of the problem itself or be introduced intentionally into the solution method. The resulting coevolutionary systems, in which autonomous agents attempt to coordinate their activities, can face social dilemmas. More generally, a resource can be interpreted as anything that expands the space of solutions available for exploration by a group, so the addition of new strategies to a collection of problem-solving agents is akin to adding new resources. This suggests that computational social dilemmas can occur in a variety of situations.

We have seen a variety of computational societies where social dilemmas can naturally arise, and this possibility should be considered when designing computational societies. This is especially important since, as we have seen, a lowered performance can easily be misinterpreted as due to other, more commonly recognized, causes. Our observations lead to two general issues for future work.

The first is, given situations in which the dilemmas are possible, how commonly are the density-dependent resource costs arising in practice such as to produce dilemmas. While best addressed by examining a collection of real problems, some insight is also possible from theoretical models of computational problems [8] and ecosystems [14].

The second open issue is how social dilemmas can be effectively addressed when they do occur, without losing the advantages of decentralization. In contrast to human societies, the ability to explicitly program the incentives of computational agents can help eliminate social dilemmas. Individual agents can be informed of the effect of their actions on the rest of the community and programmed to take that information into account when making decisions. However, this will not always be possible because individual agents often cannot feasibly determine the global consequences of their choices and thus cannot make the overall optimal choices. Other options are suggested by institutions used in human societies. However, the very different relative costs of recording usage information and transacting trades in computational and human societies suggests that mechanisms that are too unwieldy for human use may prove beneficial in computational societies. As human and computational societies interact ever more closely, we can expect that interesting new mixed strategies will evolve inside the overlap between the two worlds.

Acknowledgments
We thank Stephen Freeman for examples of solved social dilemmas in computational contexts.

References


