Avoiding Braess' Paradox through Collective Intelligence

David H. Wolpert
NASA Ames Research Center
Moffett Field, CA 94035
kagan@ptolemy.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Moffett Field, CA 94035
dhw@ptolemy.arc.nasa.gov

Tech Rep No. NASA-ARC-IC-1999-124

December 8, 1999

Abstract

In an Ideal Shortest Path Algorithm (ISPA), at each moment each router in a network sends all of its traffic down the path that will incur the lowest cost to that traffic. In the limit of an infinitesimally small amount of traffic for a particular router, its routing that traffic via an ISPA is optimal, as far as cost incurred by that traffic is concerned. We demonstrate though that in many cases, due to the side-effects of one router's actions on another routers performance, having routers use ISPA's is suboptimal as far as global aggregate cost is concerned, even when only used to route infinitesimally small amounts of traffic. As a particular example of this we present an instance of Braess' paradox for ISPA's, in which adding new links to a network decreases overall throughput. We also demonstrate that load-balancing, in which the routing decisions are made to optimize the global cost incurred by all traffic currently being routed, is suboptimal as far as global cost averaged across time is concerned. This is also due to "side-effects", in this case of current routing decision on future traffic. The theory of COllective INtelligence (COIN) is concerned precisely with the issue of avoiding such deleterious side-effects. We present key concepts from that theory and use them to derive an idealized algorithm whose performance is better than that of the ISPA, even in the infinitesimal limit. We present experiments verifying this, and also showing that a machine-learning-based version of this COIN algorithm in which costs are only imprecisely estimated (a version potentially applicable in the real world) also outperforms the ISPA, despite having access to less information than does the ISPA. In particular, this COIN algorithm avoids Braess' paradox.

1 INTRODUCTION

The problem of how to control routing across a network underlies a vast array of real-world problems including internet routing, voice/video communication, traffic flows, etc. In its general form, the problem is how to optimize the flow of certain entities (e.g.,

information packets, cars) from sources to destinations across a network of routing nodes. Here we are concerned with the version of the problem in which "optimization" consists of minimizing aggregate cost incurred by the entities flowing to their destinations. To ground the discussion, we will consider the case where the entities being routed are packets.

Currently, many real-world network routing solutions to this particular problem are based on the Shortest Path Algorithm (SPA), in which each routing node in the network maintains estimates of the "shortest paths" (i.e., minimal total incurred costs) from it to each of its destinations and at each moment satisfies any routing requests by sending all its packets down that shortest path. Many algorithms exist for efficiently computing the shortest path in the case where the costs for traversing each component of every path at any given time are known. In particular, there exist many such algorithms that can be applied when node-to-node path-cost communication is available and the costs for traversing each component are unvarying in time (e.g., Dijkstra's Algorithm [1, 3, 9, 10]. Real-world SPA's apply such algorithms to estimated costs for traversing each component of every path to generate their estimated shortest paths.

Consider the case where for all paths from a particular node to a particular destination, the costs that would be incurred by that node's routing all its current traffic along that path is known exactly to that node (the information being stored in that router's "routing table"). Clearly if a non-infinitesimal amount of traffic is being routed by our node, then in general its sending all that traffic down a single path will not result in minimal cost incurred by that traffic, no matter how that single path is chosen. However if it must choose a single path for all its traffic, then tautologically the SPA chooses the best such path. Accordingly, in the limit of routing an infinitesimally small amount of traffic, with all other nodes' strategies being a "background", such a router's running SPA is the optimal (least aggregate incurred cost) routing strategy *for that particular routing node considered individually.*

One might hope that more generally, if the node must allot all of its traffic to a single path, then its choosing that path via the SPA would be the *globally* optimal choice of a single path, at least in the limit of infinitesimally little traffic. This is not the case though, because in using the SPA the node is not concerned with the deleterious side-effects of its actions on the costs to other nodes [15, 26]. In the extreme case, as elaborated below, if all nodes were to try to minimize their personal costs via SPA's, then the nodes would actually *all* receive higher cost than would be the case under an alternative set of strategies. This is an instance of the famous Tragedy Of the Commons (TOC) [12].

Deleterious side-effects need not be restricted to extend over space; they can also extend over time. Indeed, consider the algorithm of having all routers at a given moment make routing decisions that optimize global cost incurred by the traffic *currently being routed*, an algorithm often called "load-balancing" (LB). By definition, LB avoids the

deleterious side-effects over space that can result in the TOC for the costs incurred by the traffic currently being routed. However, due to side-effects over time, even conventional LB is often suboptimal as far as global cost averaged across time is concerned. Intuitively, one would have to use "load-balancing over time" to ensure truly optimal performance.

In this paper we are concerned with how to address these kinds of deleterious side-effects, and thereby improve performance. In particular, we are interested in ways of doing this that result in better performance than that of the ubiquitous SPA.

Now use of the SPA obviously provides no guarantees, even for personal cost of the router using it, if the path-estimates of the nodes are incorrect. Such inaccuracy is the rule rather than the exception in many practical applications. Typically those estimates will be in error because node-to-node communication is not instantaneous, and therefore routing tables may be based on out of date information. More generally though, even if that communication were instantaneous, the cost to traverse a component of the network may be different by the time the packet arrives at that component.

In this paper we do not wish to investigate such topics, but rather to highlight the issue of side-effects. Accordingly we "rig the game" in favor of the SPA by constructing our simulations so that the first potential cause of routing table inaccuracy does not arise, and the second is minimized. We do this in our experiments by using an *Ideal* Shortest Path Algorithm (ISPA) which has direct access to the shortest path at each moment. Note that this ISPA provides an upper bound on the performance of any real-world SPA.

In general, even without side-effects, determining the optimal solution to a flow problem (e.g., determining what the loads on each link need to be to maximize throughput on a non-cooperative data network) can be nontractable [1, 20]. Therefore, we will concern ourselves with providing *good* solutions that avoid the difficulties the ISPA has with side-effects. It is not our aim here to present algorithms that find the best possible ("load-balanced over time") solution.

We will base our solutions on the concept of Collective Intelligence. A "COllective INtelligence" (COIN) is any pair of a large, distributed collection of interacting goal-driven computational processes among which there is little to no centralized communication or control, together with a 'world utility' function that rates the possible dynamic histories of the collection [26, 25]. In this paper we are particularly concerned with computational processes that use machine learning techniques (e.g., reinforcement learning [14, 23, 22, 24]) to try to achieve their goal, conventionally represented as maximizing an associated utility function. We consider the central COIN design problem: *How, without any detailed modeling of the overall system, can one set utility functions for the individual components in a COIN to have the overall dynamics reliably and robustly achieve large values of the provided world utility?* In other words, how can we leverage an assumption that our learners are individually fairly good at what they do? In a routing context, this question reduces to what goals one ought to provide to each router so that each router's greedily pursuing those goals will maximize throughput ("incentive

engineering"). For reasons given above, we know that the answer to this question is not provided by SPA's goals — some new set of goals is needed.

In Section 2 we discuss the SPA's deficiencies and in particular their manifestations in Braess' paradox. We also demonstrate the suboptimality of load-balancing in that section. We then present Collective Intelligence in Section 3, discuss the routing model we will use in our experiments, and show how the theory of COINs can be applied to that model to provide an alternative to shortest path algorithms. In Section 4 we present simulation results with that model that demonstrate that in networks running ISPA, the per packet costs can be as much as 32 % higher than in networks running algorithms based on COIN theory. In particular, even though it only has access to imprecise estimates of costs (a handicap that does not hold for ISPA), the COIN-based algorithm almost always avoids Braess' paradox, in stark contrast to the ISPA. In that the cost incurred with ISPA's is presumably a lower bound on that of an SPA not privy to instantaneous communication, the implication is that COINs can outperform such real-world SPA's.[1]

2 Suboptimality of Shortest Path and Load-Balancing

In this section we first demonstrate the suboptimality of an SPA when we have multiple nodes making simultaneous routing decisions, where neither node knows ahead of time the other's choice, and therefore does not know ahead of time exactly what the costs will be. We then demonstrate that such suboptimality can hold even when only one node is making a decision, and it knows what decisions the others have previously made. Next we present Braess' paradox, a particularly pointed instance of these effects. (See [2, 7, 6, 18] for other discussion of Braess' paradox in SPA routing.) We end by demonstrating the suboptimality of conventional load-balancing when cost over time is what's of interest.

2.1 SPA when multiple routers are simultaneously making decisions

Perhaps the simplest example of how individual greed on the part of all nodes can lead to their collective detriment occurs when two nodes determine that their shortest path is through a shared link with a limited capacity, while both have a second option that is slightly less preferable. In such a case, their using the common link degrades the performance of *both* parties, since due to limited capacity the performance of that link will quickly fall below that of their second option.

More precisely, consider the case where, given a load x, the shared link has a cost given by x^3, and where each router has a second option where the cost is given by $2x$. Acting alone, with a single packet to send, they would both send that packet through

[1] A brief synopsis of the COIN algorithm discussed here was presented in a space-constrained article [26]; this paper presents full details and applies the algorithm to Braess' paradox as an illustration of the suboptimality of the SPA.

packets, use of SPA will lead to a wrong routing decision.

2.2 SPA when only one router is making a decision

Consider the network shown in Figure 1. Two source routers X and Y each send one packet at a time, with X sending to either intermediate router A or B, and Y sending to either B or C. This type of network may arise in many different topologies as a subnetwork. Accordingly, difficulties associated with this network can also apply to many more complex topologies.

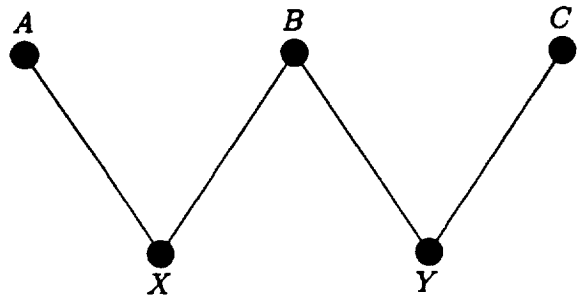


Figure 1:

Let x_A, x_B, y_B, and y_C, be the packet quantities at a particular fixed time t, at A, B, or C, and originating from X or Y, as indicated. At t, each source has one packet to send. So each of our variables is binary, with $x_A + x_B = y_B + y_C = 1$. Have $V_i(z_i)$ be the cost, per packet, at the single instant t, at router i, when the total number of packets at that instant on that router is z_i. So the total cost incurred by all packets at the time t, $G(\vec{x}, \vec{y})$, equals $x_A V_A(x_A) + (x_B + y_B)V_B(x_B + y_B) + (y_C)V_C(y_C)$.

In an ISPA, X chooses which of x_A or $x_B = 1$ so as to minimize the cost *incurred by X's packet alone*, $g_X(\vec{x}) \equiv x_A V_A(x_A) + x_B V_B(x_B + y_B)$. (Real-world SPA's typically try to approximate this by having X choose either A or B according to whether $V_A(0)$ or $V_B(y_B)$ is smaller, where those two values can be estimated via pings, for example.) In doing this the ISPA ignores the $y_B V_B(x_B + y_B)$ term, i.e., it ignores the "side effects" of X's decision.

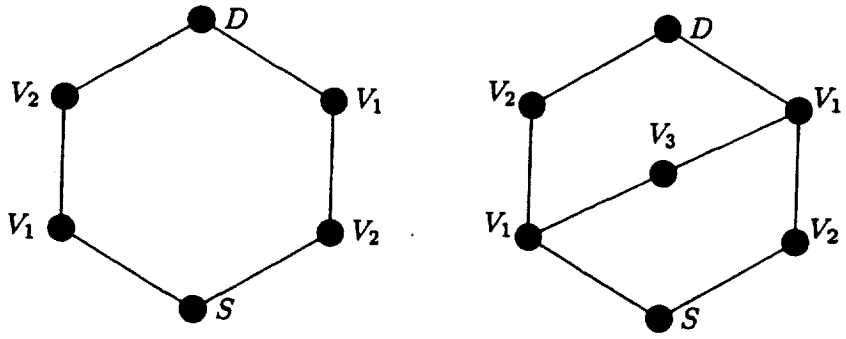
The right thing to do of course is instead have X minimize $G(\vec{x}, \vec{y})$, or more precisely, the components of $G(\vec{x}, \vec{y})$ that depend on X. Writing it out for this case, X should act to minimize $x_A V_A(x_A) + (x_B + y_B)V_B(x_B + y_B)$. Due to the constraint that $x_A + x_B = 1$, this means sending down A iff $V_A(1) < (y_B + 1)V_B(y_B + 1) - y_B V_B(y_B)$, which differs from the ISPA result in that X is concerned with the full cost of going through router B, not just the portion of that cost that its packet receives.

In the context of this example, this G-minimizing algorithm constitutes "load-balancing" (LB). Note that so long as $\text{sgn}[V_A(0) - V_B(y_B) - y_B V_B'(y_B)] \neq \text{sgn}[V_A(0) - V_B(y_B)]$, even

in the limit of infinitesimally small traffic (so that $x_A + x_B$ equals some infinitesimal δ), ISPA and LB still disagree.

2.3 Braess' Paradox

Braess' paradox [2, 7, 6, 14, 16] dramatically underscores the inefficiency of the ISPA described above. This apparent "paradox" is perhaps best illustrated through a highway traffic example first given by Bass [2]: There are two highways connecting towns S and D. The cost associated with traversing either highway (either in terms of tolls, or delays) is $V_1 + V_2$, as illustrated in Net A of Figure 2. So when $x = 1$ (a single traveler) for either path, total accrued cost is 61 units. If on the other hand, six travelers are split equally among the two paths, they will each incur a cost of 83 units to get to their destinations. Now, suppose a new highway is built connecting the two branches, as shown in Net B in Figure 2. Further, note that the cost associated with taking this highway is not particularly high (in fact for any load higher than 1, this highway has a lower cost than any other highway in the system). The benefit of this highway is illustrated by the dramatically reduced cost incurred by the single traveler: by taking the short-cut, one traveler can traverse the network at a cost of 31 units ($2\ V_1 + V_3$). Adding a new road has seemingly reduced the traversal cost dramatically.



Net A Net B

Figure 2: Hex network with $V_1 = 10x$; $V_2 = 50 + x$; $V_3 = 10 + x$.

However consider what happens when six travelers are on the highways in net B. If each node uses an ISPA, then at equilibrium each of the three possible paths contains two travelers.[2] Due to overlaps in the paths however, this results in each traveler incurring a cost of 92 units, which is higher than than what they incurred *before* the new highway was built. The net effect of adding a new road is to increase the cost incurred by *every* traveler. This phenomenon is known as Braess' paradox.

[2]We have in mind here the Nash equilibrium for this problem, where no traveler (or equivalently, no router) can gain advantage by changing strategies.

2.4 The Suboptimality of Load-Balancing

As mentioned before, LB considers side-effects of current routing decisions on other traffic currently being routed. However because it does not consider side-effects of routing decisions on future traffic, even LB may not optimize global cost averaged across all time, depending on the details of the system. Here we present an existence proof of this, by explicitly constructing a situation where conventional LB is suboptimal.

Consider a system with discrete time, in which the node X under consideration must route one packet to the (fixed) destination at each time step (cf. Section 2.2 above). Presume further that no traffic enters any of the nodes X sends to except for X. (So that traffic coming from X is the sole source of any costs associated with X's outbound links.) Let $S(t)$ be the number of times our node sent a packet down some link A in the W time steps preceding t, and take $s(t) = A, B$ to mean that the router uses link A or B, respectively, at time t. Model queue backups and the like by having the cost to send a packet down link A at time t be $C_A(S(t)/W)$, and have the cost for our router to instead send the packet down link B be $C_B(1 - S(t)/W)$, For simplicity we assume that both $C_A(.)$ and $C_B(.)$ are monotonically increasing functions of their arguments.

Restrict attention to nodes that work by having $s(t) = A$ iff $S(t) \leq k$ for some real-valued threshold k. The LB algorithm will choose $s(t) = A$ iff $C_A(S(t)/W) \leq C_B(1 - S(t)/W)$. So the LB algorithm's behavior is indistinguishable from this kind of threshold algorithm, with k set so that $C_A(k/W) = C_B(1 - k/W)$. (We implicitly assume that $C_A(.)$ and $C_B(.)$ are chosen so that such a solution exists for $1 < k < W - 1$.) The question is what k should be to optimize total averaged cost across time, and in particular if that k is the same as k_{LB}, the k that LB uses.

Now as we go from one time step to the next, the routing decision made W time steps ago drops out of the computation of $S(t)$, while the routing decision just made is newly included. In general, $S(t+1) = S(t)+1$ if the router just used A at time t and used link B at the time W time steps into the past. On the other hand, $S(t+1) = S(t) - 1$ if the router just used B and used A W time steps ago, while $S(t+1) = S(t)$ if the routing decision just made is the same as the routing decision W time steps ago. So in general, $S(t)$ can only change by -1, 0, or +1 as we go from one time step to the next.

Consider cases where $1 < k < W - 1$, so that eventually the router must choose an A, and at some subsequent time t^* the router switches from A to B. At that time $s(t^* - 1) = A$ and $s(t^*) = B$. This implies that $S(t^* - 1) \leq k, S(t^*) > k$. Define the value $S(t^* - 1)$ as k^*. Note that $S(t^*) = k^* + 1$, and $k - 1 < k^* \leq k$.

Now for any time t', if $S(t') = k^* + 1$, $s(t'+1) = B$, and the only possible next values are $S(t'+1) = k^*$ or $S(t'+1) = k^*+1$, depending on the old decision $s(t-W)$ that gets dropped out of the window. Similarly, if $S(t') = k^*$, $s(t'+1) = A$, and the only possible next values are $S(t'+1) = k^*$ or $S(t'+1) = k^* + 1$, again depending on the old decision being dropped. So we see that once $S(t') \in \{k^*, k^*+1\}$, it stays there forever.

This means that because of the relationship between k and k^*, in any interval of W consecutive time steps subsequent to t^*, the number of packets sent along A by router X must be $\in (k-1, k+1]$,[3] and therefore the number sent along B must be $\in [W-(k+1), W-(k-1))$. Each time that a packet is sent along A the cost incurred is the cost of link A with average traffic level $S(t)/W$, $C_A(S(t)/W)$. Similarly, each time the link B is chosen, the cost incurred is $C_B(1-S(t)/W)$. Since $S(t) \in \{k^*, k^*+1\}$, and both $C_A(.)$ and $C_B(.)$ are monotonically increasing, the cost for sending the packet down link $A \in (C_A((k-1)/W), C_A((k+1)/W]$, and that for sending it down link B is contained in $[C_B(1-(k+1)/W), C_B(1-(k-1)/W))$.

Now we know that the choice of A must have average frequency (across all time) between k^*/W and $(k^*+1)/W$. Similarly, B will have average frequency between $(1-(k^*+1)/W)$ and $1-k^*/W$. Accordingly, the average cost is bounded above by

$$\frac{k^*+1}{W} C_A\left(\frac{k+1}{W}\right) + \left(1-\frac{k^*}{W}\right) C_B\left(1-\frac{k-1}{W}\right), \quad (1)$$

where the first term provides the maximum possible average cost for using link A, while the second term independently provides the maximum possible average cost for using link B. (Note that the actual cost will be lower sincethe two frequencies in this bound, one for A and one for B, cannot both have the values indicated.) Because $k-1 < k^* \leq k$ and since $1-\frac{k-1}{W} = 1+\frac{2}{W}-\frac{k+1}{W}$, our upper bound is itself bounded above by

$$\frac{k+1}{W} C_A\left(\frac{k+1}{W}\right) + \left(1+\frac{2}{W}-\frac{k+1}{W}\right) C_B\left(1+\frac{2}{W}-\frac{k+1}{W}\right). \quad (2)$$

The optimal k will result in an average cost lower than the minimum over all k of the upper bound on average cost, given in Equation 2. So the average cost for the optimal k is bounded above by the minimum over k of this upper bound. Lable this argmin of Equation 2 k'.

Since other values of k besides k_{LB} result in behavior equivalent to LB, it does not suffice to simply test if $k' = k_{LB}$. Instead let us evaluate some lower bounds in a simiolar fashion to how we evaluated upper bounds. Using the average frequencies discussed above, the average cost is bounded below by:

$$\frac{k^*}{W} C_A\left(\frac{k-1}{W}\right) + \left(1-\frac{1}{W}-\frac{k^*}{W}\right) C_B\left(1-\frac{k+1}{W}\right), \quad (3)$$

where the first term provides the minimum possible average cost for using link A, while the second term provides the minimum possible average cost for using link B. Again, because $k-1 < k^* \leq k$, the term is Equation 3 is further bounded below by

$$\frac{k-1}{W} C_A\left(\frac{k-1}{W}\right) + \left(1-\frac{2}{W}-\frac{k-1}{W}\right) C_B\left(1-\frac{2}{W}-\frac{k-1}{W}\right). \quad (4)$$

In particular this bound holds for the average cost of the LB algorithm:

$$\frac{k_{LB}-1}{W} C_A\left(\frac{k_{LB}-1}{W}\right) + \left(1-\frac{2}{W}-\frac{k_{LB}-1}{W}\right) C_B\left(1-\frac{2}{W}-\frac{k_{LB}-1}{W}\right), \quad (5)$$

[3]Note that it is possible to send $k+1$ packets along A, but not $k-1$ packets.

where as before k_{LB} satisfies $C_A(k_{LB}/W) = C_B(1 - k_{LB}/W)$.

By appropriate choice of $C_A(.)$ and $C_B(.)$, we can ensure that the lower bound on the cost with the LB algorithm (Equation 5 evaluated with $k = k_{LB}$) is higher than the upper bound on the average cost incurred by the optimal algorithm (the minimum over k of Equation 2).[4] That is, the best possible average cost achieve by load balancing will be worse than the worst average cost that could arise through the optimal routing strategy. This establishes that LB does not engage in optimal routing.

3 COIN-based Routing

One common solution to these type of side-effect problems is to have particular components of the network (e.g., a "network manager" [15]) dictate certain choices to other nodes. This solution can incur major brittleness and scaling problems however. Another kind of approach, which avoids the problems of a centralized manager, is to provide the nodes with extra incentives that can induce them to take actions that are undesirable to them from a strict SPA sense. Such incentive can be in the form of "taxes" or "tolls" added to the costs associated with traversing particular links to discourage the use of those links. Such schemes in which tolls are superimposed on the nodes' goals are a special case of the more general approach of replacing the goal of each node with a new goal. These new goals are specifically tailored so that if they are collectively met the system maximizes throughput, with no additional *a priori* concern built into a particular node's goal for the SPA-type cost incurred by that node's packets. Intuitively, in this approach, we provide each node with a goal that is "aligned with the global objective, independent of that goal's relation to the SPA-type cost incurred by the traffic routed by that node.

In this section, we summarize the theory of such systems, which are called Collective Intelligences [24, 22]. We then use that theory to present an algorithm that only uses limited knowledge of the state of the network (in particular knowledge that is readily available to routers in common real data networks) to make routing decisions. At each router, this algorithm uses a Memory Based (MB) machine learning algorithm to estimate the value that a private utility (provided by the COIN formalism) would take on under the different candidate routing decisions. It then makes routing decisions aimed at maximizing that utility. (We call this algorithm an MB-COIN.)

[4] For example, for $C_A(x) = x^2$ and $C_B(x) = x$, balancing the loads on A and B — setting $C_A(S(t)/W) = C_B(1-S(t)/W)$ — results in $(S(t)/W)^2 = 1-S(t)/W$, leading to $k_{LB}/W = \frac{\sqrt{5}-1}{2} = .618$. For $W = 1000$, the associated lower bound on average cost (Equation 5) is $.617(.617)^2 + (.998 - .617)^2 = .380$. On the other hand, with C_A and C_B given as above, Eq 2 is $(\frac{k+1}{W})^3 + (1 + \frac{2}{W} - \frac{k+1}{W})^2$. Differentiating with respect to k and setting the result to zero leads to $\frac{k'}{W} = -\frac{1}{3} - \frac{1}{W} + \frac{\sqrt{28+48/W}}{6}$. For a window size of $W = 1000$, this yields $k'/W = .548$, a different result than k_{LB}. Plugging into Equation 2, the upper bound on the performance with k' is $(.549)^3 + (1.002 - .549)^2 = .371$, which is less than .380.

3.1 COIN Formalism

In this paper we consider systems that consist of a set of nodes, connected in a network, evolving across a set of discrete, consecutive time steps, $t \in \{0, 1, ...\}$. Without loss of generality, we let all relevant characteristics of a node η at time t — including its internal parameters at that time as well as its externally visible actions — be encapsulated by a Euclidean vector $\zeta_{\eta,t}$. We call this the "state" of node η at time t, and let $\underline{\zeta}$ be the state of all node across all time.

World utility, $G(\underline{\zeta})$, is a function of the state of all nodes across all time. When η is an agent that uses a machine learning (ML) algorithm to "try to increase" its **private utility**, we write that private utility as $\gamma_\eta(\underline{\zeta})$. We assume that $\underline{\zeta}$ encompasses all physically relevant variables, so that the dynamics of the system is deterministic (though of course imprecisely known to anyone trying to control the system). Note that this means that *all* characteristics of an agent η at $t = 0$ that affects the ensuing dynamics of the system must be included in $\zeta_{\eta,0}$. This includes in particular the algorithmic specification of its private utility (typically in the physical form of some computer code) if it has one,

As elaborated below, the mathematics is generalized beyond ML-based agents through an artificial construct: **personal utilities** $\{g_\eta(\underline{\zeta})\}$. Our goal as COIN designers is to maximize world utility through the proper selection of personal utility functions. Intuitively, the idea is to choose personal utilities that are aligned with the world utility, and that also have the property that it is relatively easy for us to configure each node so that the associated personal utility achieves a large value. (In particular, for ML-based agent nodes, it is often possible to induce a large value of personal utility simply by assigning that utility to the agent as its private utility, so that the agent then ensures the desired large value.) In for this paper, we restrict attention to utilities of the form $\sum_t R_t(\underline{\zeta}_{,t})$ for **reward functions** R_t. In particular, as shown below, overall network throughput is expressible this way.

We need a formal definition of the concept of having personal utilities be "aligned" with G. In this regard, consider systems where the world utility is the sum of the personal utilities of the individual nodes. This might seem a reasonable candidate for an example of "aligned" utilities. However such systems are examples of the more general class of systems that are "weakly trivial". It is well-known that in weakly trivial systems each individual agent greedily trying to maximize its own utility can lead to the tragedy of the commons [] and actually *minimize* G. Evidently, having $G = \sum_\eta g_\eta$ is not a formal instance of "aligned" utilities; some alternative formalization is needed.[5]

A more careful alternative formalization of the notion of aligned utilities is the concept of "factored" systems. A system is **factored** when the following holds for each agent η

[5]Note that in the simple network discussed in Section 2.1, the utilities are weakly trivial, since $G(\vec{x}, \vec{y}) = g_x(\vec{x}) + g_y(\vec{y})$. This provides another perspective on the suboptimality of ISPA in that network.

individually: A change at time 0 to the state of η alone results in an increase for $g_\eta(\underline{\zeta})$ if and only if it results in an increase for $G(\underline{\zeta})$.

For a factored system, the side effects of a change to η's $t = 0$ state that increases its personal utility cannot decrease world utility. In game-theoretic terms, optimal global behavior corresponds to the agents' reaching a personal utility Nash equilibrium for such systems [10]. In this sense, there can be no TOC for a factored system. As a trivial example, a system is factored for $g_\eta = G \; \forall \eta$.

Define the $(t = 0)$ **effect set** of node η at $\underline{\zeta}$, $C_\eta^{eff}(\underline{\zeta})$, as the set of all components $\zeta_{\eta',t}$ which under the forward dynamics of the system have non-zero derivative with respect to the state of node η at $t = 0$. Intuitively, η's effect set is the set of all components $\zeta_{\eta',t}$ which would be affected by a change in the state of node η at time 0. (They may or may not be affected by changes in the $t = 0$ states of the other nodes.)

Next for any set σ of components (η', t), define $\mathrm{CL}_\sigma(\underline{\zeta})$ as the "virtual" vector formed by clamping the σ-components of $\underline{\zeta}$ to an arbitrary fixed value. (In this paper, we take that fixed value to be $\vec{0}$ for all components listed in σ.) The value of the effect set **wonderful life utility** (WLU for short) for node η is defined as:

$$WLU_\sigma(\underline{\zeta}) \equiv G(\underline{\zeta}) - G(\mathrm{CL}_{C_\eta^{eff}(\underline{\zeta})}(\underline{\zeta})). \tag{6}$$

In other words, the WLU for the effect set of node η is the difference between the actual world utility and the virtual world utility where all node-time pairs that are affected by node η have been clamped to a zero state while the rest of $\underline{\zeta}$ is left unchanged.

Since we are clamping to $\vec{0}$, we can view η's effect set WLU as analogous to the change in world utility that would have arisen if node η "had never existed". (Hence the name of this utility - cf. the Frank Capra movie.) Note however, that CL is a purely "fictional", counter-factual operation, in the sense that it produces a new $\underline{\zeta}$ without taking into account the system's dynamics. The sequence of states the node η is clamped to in the definition of the WLU need not be consistent with the dynamical laws embodied in C. This dynamics-independence is a crucial strength of the WLU. It means that to evaluate the WLU we do *not* try to infer how the system would have evolved if node η's state were set to 0 at time 0 and the system evolved from there. So long as we know $\underline{\zeta}$ extending over all time, and so long as we know G, we know the value of WLU.

If our system is factored with respect to personal utilities $\{g_\eta\}$, then we want each $\zeta_{\eta,0}$ to be a state that induces as high a value of personal utility as possible (given the initial states of the other nodes). Assuming η is ML-based and able to achieve large values of most any private utility specified in $\zeta_{\eta,0}$, we would likely induce such a state of high personal utility if η's private utility were set to that associated personal utility: $\gamma_\eta \equiv \zeta_{\eta,0;\mathrm{private-utility}} = g_\eta$. Enforcing this equality, our problem becomes determining what $\{\gamma_\eta\}$ the agents will best be able to maximize while also causing dynamics that is factored with respect to the $\{\gamma_\eta\}$.

As mentioned above, regardless of the system dynamics, having $\gamma_\eta = G \; \forall \eta$ means the

system is factored. It is also true that regardless of the dynamics, $\gamma_\eta = WLU_{C_\eta^{eff}}\ \forall \eta$ is a factored system (for $g_\eta = \gamma_\eta$) (proof in []). Which of these two choices of $\{\gamma_\eta\}$ should we use?

To answer this, note that since each agent is operating in a large system, it may experience difficulty discerning the effects of its actions on G when G sensitively depends on all the myriad components of the system. Therefore each η may have difficulty learning how to achieve high γ_η when $\gamma_\eta = G$. (In particular, in the routing problem, having $\gamma_\eta = G$ means providing each router with the full throughput of the entire network. This is usually infeasible in practice. Even if it weren't though, using these private utilities would mean that the routers face a very difficult task in trying to discern the effect of their actions on the overall throughput, and therefore would likely be unable to learn their best routing strategies.) This problem can be obviated using effect set WLU as the private utility, since the subtraction of the clamped term removes much of the "noise" of the activity of other agents, leaving only the underlying "signal" of how the agent in question affects the utility. (This reasoning is formalized as the concept of "learnability" in [22]. Accordingly, one would expect that using WLU's should result in better performance, than having $\gamma_\eta = G\ \forall \eta$. In practice, we will often only be able to estimate the "primary", most prominent portion of the effect set. However assuming that the associated WLU is close enough to being factored, we would expect the advantage in learnability with such a WLU to still result in better performance than would using $\gamma_\eta = G\ \forall \eta$. (See [24, 22].)

3.2 Model Description

To apply the COIN formalism to a network routing model, we must formally identify the components of that model as deterministically evolving elements of vectors $\underline{\zeta}$. In the model used in this paper, at any time step t, all traffic at a router r is a real-valued number together with an ultimate destination tag, d. At each time step, each router sums all traffic received by r at that time step is used to determine r's "instantaneous load". That load is given by $z_r(t) = \sum_{r'} x_{r,d,r'}(t)$ where r' is a first stop on a path from router r to d, and $x_{r,d,r'}(t)$ is the total traffic at time t going from r towards d with a first stop at r'. After computing its instantaneous load at time t, the router then sends its load to downstream routers and the cycle repeats itself, until all traffic reaches its destinations. (So for example, since no packets are dropped in our model, $\sum_{r'} x_{r',d,r}(t-1) = \sum_{r''} x_{r,d,r''}(t)$.)

In a real network, the cost of traversing a router does not change dramatically from one packet to the next as these costs are mainly determined by the state of the queues at the router. To simulate this effect, we use cumulative values of the load at a router rather than instanteneous load to determine the cost of traversing that router. More precisely, we use its "windowed load", which is a running average of that router's load value over a window of the previous W timesteps, Z_r, where $Z_r(t) = \frac{1}{W} \sum_{\tau=t-W+1}^{t} z_r(\tau)$.

Using such a window ensures that the costs across nodes change over a time frame that is larger than the individual routing decisions. The windowed load is the argument of the *load-to-cost* function, $V(\cdot)$, which provides the *cost* accrued at this timestep by each packet traversing this router at this timestep. That is, at time t, the cost for each packet to traverse router r is given by $V(Z_r(t))$. (Note that in our model, the costs are accrued at the routers, not the links.) Different routers have different $V(\cdot)$, to reflect the fact that real networks have differences in router software and hardware (response time, queue length, processing speed etc). Note that with this notation, world utility is given by $G = \sum_t \sum_r \; z_r(t) \; V_r(Z_r(t))$.

At time step t, ISPA has access to all the windowed loads at time step $t-1$ (i.e., it has access to $Z_{r'}(t-1) \; \forall r'$), and assumes that those values will remain the same at time t. Note that for large window sizes, this assumption should be arbitrarily accurate. Based on those values, in ISPA, each router sends packets along the path that it surmises will minimize the costs accumulated by its packets. In this model, all of the dynamics is encapsulated in the values of $x_{r,d,r'}(t) \; \forall r, d, r', t$. Accordingly, $\underline{\zeta}$ must specify those values. For routing based on ML agents, other variables must also be included in $\underline{\zeta}$, to capture the internal parameters used by those agents to make their routing decisions.

3.3 COIN Routing

Based on the COIN formalism presented in Section 3.1 and the model described above, we now present the COIN-based routing algorithms.

First, we must define $\underline{\zeta}$ precisely. We identify a node as a router-destination pair. The state of node $\eta = r - d$ at time t is the Euclidean vector of the values $x_{r,d,r'}(t) \; \forall r'$, together with an internal parameters router r might use to route traffic destined for d. To compute the WLU for a node we must estimate the associated primary effect set. In the results presented here, the effect set of a node is estimated to be all nodes that share the same destination. Using this estimate for the effect set, the WLU for a node η is given by the difference between the total cost accrued by all nodes in the network and the cost accrued by nodes when all nodes sharing η's destination are "erased." More precisely, any node η that has a destination d will have the following effect set WLU, g_η:

$$
\begin{aligned}
g_\eta(\underline{\zeta}) = \; & G(\underline{\zeta}) - G(\mathrm{CL}_{C^{eff}_\eta}(\underline{\zeta})) \\
= \; & \sum_{t,r'} z_{r'}(t) \; V_{r'}(Z_{r'}(t)) \; - \sum_{t,r',d' \neq d} z_{r',d'}(t) \; V_{r'}(\sum_{d' \neq d} Z_{r',d'}(t)) \\
= \; & \sum_{t,r'} \left(\sum_{d'} z_{r',d'}(t) \; V_{r'}(Z_{r'}(t)) \; - \sum_{d' \neq d} z_{r',d'}(t) \; V_{r'}(\sum_{d' \neq d} Z_{r',d'}(t)) \right) \\
= \; & \sum_{t,r'} \left(z_{r',d}(t) V_{r'}(Z_{r'}(t)) + \sum_{d' \neq d} z_{r',d'}(t) [V_{r'}(\sum_{d'} Z_{r',d'}(t)) - V_{r'}(\sum_{d' \neq d} Z_{r',d'}(t))] \right) \quad (7)
\end{aligned}
$$

where $Z_{r,d}(t)$ is the window load at router r at time t that has ultimate destination d, so

that $\sum_d Z_{r,d}(t) = Z_r(t)$, and similarly for $z_{r,d}(t)$. Notice that each term in the expression in Equation 7 can be computed at each router r' separately, from information available to that router. Subsequently those terms can be propagated through the network by to η, much the same way as routing tables updates are propagated.

The starting point of the COIN-based routing algorithm is the collection of the data that is to be used in the process of making routing decisions. In our experiments that data was collected during the initial running of an SPA. In this stage, the routing decisions are made using the SPA, but the resulting actions are "scored" using the WLU given in Equation 7. [6] This training set then consists of loads on outgoing links as inputs and resulting WLU values as outputs for each destination. After sufficient data is collected using the ISPA, the system switches to using the COIN algorithm. At this stage, the memory based COIN (MB-COIN) routes packets along the path that in its estimate would provide the best WLU.

More precisely, the router uses a single nearest neighbor algorithm — which simply assigns the inputs to the class to which its "nearest" neighbor in input space belongs — as its learner.[7] This process consists of the learner finding the state of the network (loads on outgoing links) closest to that which would result from each potential routing decision. Then the learner assigns the WLU associated with those states as estimates for the WLU that would result from said routing decisions. Finally, the router sends the packets along the path which has the highest estimated WLU value.

Note that the routers are trying to estimate how the various routing decisions (loads at individual time steps) will affect their WLU values (based on loads summed over a window). In the limit of a large window, this becomes a differential operation, and as such can be reduced to derivative balancing over personal utility functions. In these experiments we used routing decisions based on such estimates to help the MB-COIN. These values provide an actual WLU value (analogous to the ideal SPA) and can be used to "steer" the learner in the right direction. Because they are based on the actual WLU derivatives, we will call them Full Knowledge COIN (FK-COIN) values. The steering parameter determines how often the routing decision is based on the MB-COIN as opposed to the FK-COIN.

4 SIMULATION RESULTS

Based on the model and routing algorithms discussed above, we have performed simulations to compare the performance of ISPA and MB-COIN across a variety of networks,

[6] We are using the SPA as the starting point, since it most likely will be the algorithm running prior to the COIN algorithm. Alternately one can collect data by allowing the routers to make random decision, or decisions that "sweep" the possible set of actions.

[7] This is a very simple learning algorithm, and we use it here to show the feasibility of a COIN based routing algorithm. The performance of the MB-COIN can be improved if more sophisticated learning algorithms (e.g., Q-learning [20, 21]) are used.

varying in size from five to eighteen nodes. In all cases traffic was inserted into the network deterministically at the sources. The results we report are averaged over 20 runs, but we do not report error bars as they are all lower than 0.05.

In particular in Sections 4.1 - 4.4 we analyze traffic patterns over four networks where ISPA suffers from the Braess' paradox, while the routing scheme highlighted in Section 3.3 does not. Then in Section 4.5 we discuss the effect of the "steering" parameter which determines the amount of information to which the the COIN algorithm has access, on the performance of COIN routing.[8]

4.1 Bootes Network

The first network we will investigate is shown in Figure 3. It is by many accounts a trivial network, and in Net A, in particular, the sources do not have any choices to make. At each time step, each source sends its load (given in the Table 1) to the destination. The interesting phenomena occur when source S_1 is given a choice in its routing by the addition of the new router. Efficient usage of this routing option can lead better throughput for the network, while poor usage can have deleterious effects.

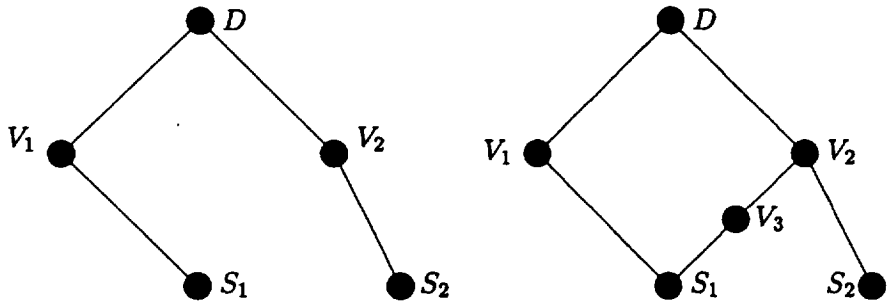


Figure 3: Bootes Network

Tables 1-2 show the results for both ISPA and MB-COIN on two variations of the network shown in Figure 3. In both tables, the loads show the packets injected to the network at S_1 and S_2 respectively, at each time step.

The MB-COIN results are identical to the ISPA results in the absence of the additional link. The Braess' paradox is apparent in ISPA where the addition of the new link degrades the performance of the ISPA in all cases. The MB-COIN on the other hand shows no ill-effect from the addition of the new link. In particular, with both sets of cost functions, the MB-COIN either uses the additional link efficiently or chooses to ignore it, thereby avoiding the Braess' paradox. Adding the new link causes a degradation of the performance by as much as 30 % (load = 2) for the ISPA, whereas for the same load MB-COIN performance improves by 7 %.

[8] In Sections 4.1 - 4.4, the steer parameter is set at 0.5.

This network is particularly interesting because of its simplicity, and demonstrates that the Braess' paradox is not restricted to complex situations. Here a simple two source, one destination network suffers from this phenomenon when a link that allows decision making is added.

Table 1: Average Per Packet Cost for BOOTES2 Network with (Net B) and without (Net A) added link for $V_1 = 10 + log(1+x)$; $V_2 = 4x^2$; $V_3 = log(1+x)$.

Load	Net	ISPA	MB-COIN
1,1	A	6.35	6.35
	B	8.35	5.93
2,1	A	8.07	8.07
	B	10.40	7.88
2,2	A	9.55	9.55
	B	10.88	9.71
4,2	A	10.41	10.41
	B	11.55	10.41

Table 2: Average Per Packet Cost for BOOTES4 Network with (Net B) and without (Net A) added link for $V_1 = 50 + log(1+x)$; $V_2 = 10x$; $V_3 = log(1+x)$.

Load	Net	ISPA	MB-COIN
1,1	A	30.35	30.35
	B	20.35	20.35
2,2	A	35.55	35.55
	B	40.55	34.99
4,2	A	41.07	41.07
	B	50.47	44.13
6,3	A	44.63	44.63
	B	51.40	44.63

4.2 Hex Network

In this section we revisit the network first discussed in Section 2.1 (shown in Figure 2). In Table 3 we give full results for the cost functions discussed in that section. We then use load-to-cost functions which are qualitatively similar to those used in Section 2.1, but incorporate non-linearities that better represent router characteristics. That load-to-cost function and associated results are reported in Table 4.

This network demonstrates that while the addition of a new link may be beneficial in low traffic cases, it leads to bottlenecks in higher traffic regimes. For ISPA although the per packet cost for loads of 1 and 2 drop drastically when the new link is added, the

Table 3: Average Per Packet Cost for HEX3 Network with (Net B) and without (Net A) added link for $V_1 = 50 + x$; $V_2 = 10x$; $V_3 = 10 + x$.

Load	Net	ISPA	MB-COIN
1	A	55.50	55.56
	B	31.00	31.00
2	A	61.00	61.10
	B	52.00	51.69
3	A	66.50	66.65
	B	73.00	64.45
4	A	72.00	72.25
	B	87.37	73.41

Table 4: Average Per Packet Cost for HEX2 Network with (Net B) and without (Net A) added link for $V_1 = 50 + log(1 + x)$; $V_2 = 10x$; $V_3 = log(1 + x)$.

Load	Net	ISPA	MB-COIN
1	A	55.41	55.44
	B	20.69	20.69
2	A	60.69	60.80
	B	41.10	41.10
3	A	65.92	66.10
	B	61.39	59.19
4	A	71.10	71.41
	B	81.61	69.88

per packet cost increases for higher loads. The MB-COIN on the other uses the new link efficiently. Notice that the MB-COIN's performance is slightly lower than that of the ISPA in the absence of the additional link. This is caused by the MB-COIN having to use a learner to estimate the WLU values for potential actions whereas the ISPA simply has direct access to all the information it needs (costs at each link).

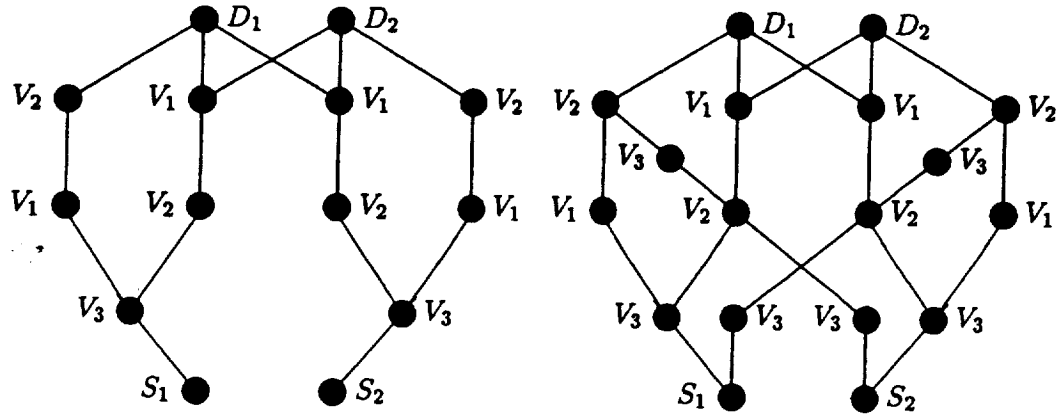
4.3 Butterfly Network

The next network we investigate is shown in Figure 4. It is an extension to the simple network discussed in Section 4.1. We now have doubled the size of the network and have three sources that have to route their packets to two destinations. Initially the two halves of the network have minimal contact, but with the addition of the extra link two sources share a common router on their potential shortest path.

Table 5 presents two sets of results: first we present results for uniform traffic through all three sources, and then results for asymmetric traffic. For the first case, the Braess'

4.4 Ray Network

the final network we investigate is a larger network where the number of routers with decision making potential is significantly higher than in the previous networks. Figure 5 shows the initial network (Net A) and the "augmented" network (Net B), where new links have been added. The original network has relatively few choices for the routers, as packets are directed toward their destinations along "conduits." New links are added to provide more choices (crossing patterns) in case certain conduits experience large costs.



Net A Net B

Figure 5: Ray network with $V_1 = 10 + log(1 + x)$; $V_2 = 4x^2$; $V_3 = log(1 + x)$

Table 6: Average Per Packet Cost for RAY1 Network with (Net B) and without (Net A) added link for $V_1 = 50 + log(1 + x)$; $V_2 = 10x$; $V_3 = 10 + log(1 + x)$. The "Load" column shows the packet insertion rates at sources S_1 and S_2 respectively.

Load	Net	ISPA	MB-COIN
2,2	A	143.6	143.7
	B	124.4	126.9
3,3	A	154.6	154.9
	B	165.5	151.0
4,4	A	165.4	166.0
	B	197.7	165.6
6,6	A	186.7	187.4
	B	205.1	191.6

Table 6 shows the simulation results for these networks. At low load levels both the ISPA and the MB-COIN use the new links although the MB-COIN performs slightly worse. This is mainly caused by the difficulty encountered by the simple learner (single

nearest neighbor algorithm) in quickly learning the traffic patterns in this large network. The discrepancies though, are minor and can be readily corrected by using more powerful learners. Note however, that the MB-COIN still avoids the Braess' paradox in this network in all cases but the highest traffic regime. However, even there, the effect is significantly milder than that encountered by the ISPA.

4.5 Steering the MB-COIN

The final aspect of the COIN based routing we investigate is the impact of the switching parameter discussed in Section 3.3. This parameter both controls the amount of exploration the algorithm performs and determines the "faithfulness" of the MB-COIN to the the derivative based full knowledge COIN (FK-COIN). In Figure 6 we also provide results based solely on the FK-COIN algorithm, which provides a bound on the performance of the MB-COIN.

For HEX (Figure 6(a)), the worst setting for MB COIN, which corresponds to no steering, is comparable to ISPA, while with high steering (0.5) the results are similar to that of the FK-COIN. This figure dramatically demonstrates the strength of this algorithm: as the learner has more information to work with, it bridges the gap between a suboptimal algorithm susceptible to Braess' paradox to one which efficiently avoids it.

For RAY (Figure 6(b)), the tradeoff is more critical. Without steering, the MB-COIN performs poorly in this network. This is not surprising in that because there are many choices that affect the outcome, the simple memory based learner needs proper "seeding" to be able to perform well. Note that with the addition of steering the MB-COIN quickly outperforms the ISPA.

Finally, for both Butterfly and Bootes networks (Figures 6(c) - 6(d)) the MB-COIN needs very little steering to perform well. Although for Butterfly, the performance of MB-COIN improves slightly with more information, it is significantly better than the ISPA across the board.

5 CONCLUSION

Effective routing in a network is a fundamental problem in many fields, including data the data communications and transportation. Shortest path algorithms provide an elegant solution to this problem, but under certain circumstances suffer from less than desirable effects. On such effect is Braess' paradox where increased capacity results in lower overall throughput.

Collective Intelligence is a novel way of addressing distributed control problems. In a COIN, the central issue is in determining the personal objectives of the components such that their greedy pursuit of those goals leads to a globally desirable solution. We have summarized COIN theory and derived a routing algorithm based on that theory. In our

nearest neighbor algorithm) in quickly learning the traffic patterns in this large network. The discrepancies though, are minor and can be readily corrected by using more powerful learners. Note however, that the MB-COIN still avoids the Braess' paradox in this network in all cases but the highest traffic regime. However, even there, the effect is significantly milder than that encountered by the ISPA.

4.5 Steering the MB-COIN

The final aspect of the COIN based routing we investigate is the impact of the switching parameter discussed in Section 3.3. This parameter both controls the amount of exploration the algorithm performs and determines the "faithfulness" of the MB-COIN to the the derivative based full knowledge COIN (FK-COIN). In Figure 6 we also provide results based solely on the FK-COIN algorithm, which provides a bound on the performance of the MB-COIN.

For HEX (Figure 6(a)), the worst setting for MB COIN, which corresponds to no steering, is comparable to ISPA, while with high steering (0.5) the results are similar to that of the FK-COIN. This figure dramatically demonstrates the strength of this algorithm: as the learner has more information to work with, it bridges the gap between a suboptimal algorithm susceptible to Braess' paradox to one which efficiently avoids it.

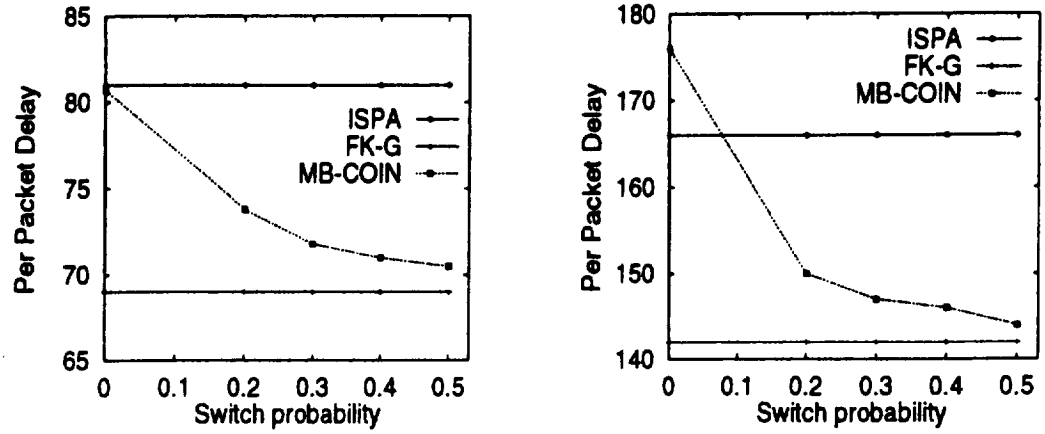
For RAY (Figure 6(b)), the tradeoff is more critical. Without steering, the MB-COIN performs poorly in this network. This is not surprising in that because there are many choices that affect the outcome, the simple memory based learner needs proper "seeding" to be able to perform well. Note that with the addition of steering the MB-COIN quickly outperforms the ISPA.

Finally, for both Butterfly and Bootes networks (Figures 6(c) - 6(d)) the MB-COIN needs very little steering to perform well. Although for Butterfly, the performance of MB-COIN improves slightly with more information, it is significantly better than the ISPA across the board.

5 CONCLUSION

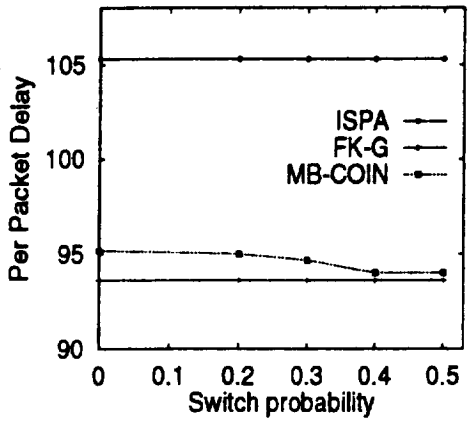
Effective routing in a network is a fundamental problem in many fields, including data the data communications and transportation. Shortest path algorithms provide an elegant solution to this problem, but under certain circumstances suffer from less than desirable effects. On such effect is Braess' paradox where increased capacity results in lower overall throughput.

Collective Intelligence is a novel way of addressing distributed control problems. In a COIN, the central issue is in determining the personal objectives of the components such that their greedy pursuit of those goals leads to a globally desirable solution. We have summarized COIN theory and derived a routing algorithm based on that theory. In our

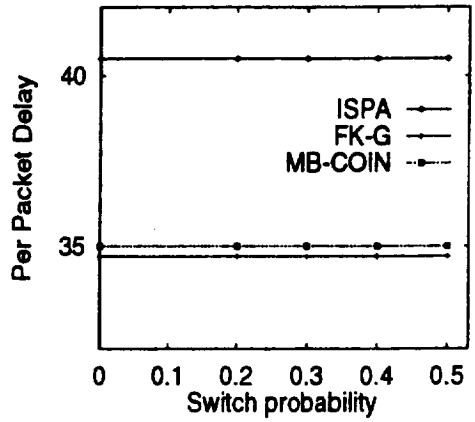


(a) Hex4

(b) Ray4



(c) Butterfly4



(d) Bootes4

Figure 6: Impact of switching.

simulations, the ISPA provided average costs as much as 32 % higher than the COIN-based algorithm. Furthermore the COIN-based algorithm avoided the Braess' paradox that seriously deteriorated the performance of the ISPA.

In the work presented here, the COIN-based algorithm had to overcome severe limitations. Firstly, the selection of the primary effect set was almost arbitrary, and therefore rendered the learners tasks more difficult than it needed to be. Secondly, the learners themselves were particularly simple-minded, and therefore were not able effectively maximize the gains. That a COIN-based router with such serious limitations outperformed an ideal shortest path algorithm demonstrates the strength of the proposed method. We are currently investigating novel utilities that are more "learnable" for the routers as well as expand the simulations to larger networks using a commercial event driven simulator.

6 Acknowledgements

We would like to thank Joe Sill for helpful discussion.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, New Jersey, 1993.

[2] T. Bass. Road to ruin. *Discover*, pages 56–61, May 1992.

[3] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[4] J. Boyan and M. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems - 6*, pages 671–678. Morgan Kaufmann, 1994.

[5] S. P. M. Choi and D. Y. Yeung. Predictive Q-routing: A memory based reinforcement learning approach to adaptive traffic control. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems - 8*, pages 945–951. MIT Press, 1996.

[6] J. E. Cohen and C. Jeffries. Congestion resulting from increased capacity in single-server queueing networks. *IEEE/ACM Transactions on Networking*, 5(2):305–310, 1997.

[7] J. E. Cohen and F. P. Kelly. A paradox of congestion in a queuing network. *Journal of Applied Probability*, 27:730–734, 1990.

[8] N. Deo and C. Pang. Shortest path algorithms: Taxonomy and annotation. *Networks*, 14:275–323, 1984.

[9] E. Dijkstra. A note on two problems in connection with graphs. *Numeriche Mathematics*, 1(269-171), 1959.

[10] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, MA, 1991.

[11] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[12] M. Heusse, D. Snyers, S. Guerin, and P. Kuntz. Adaptive agent-drivent routing and load balancing in communication networks. *Advances in Complex Systems*, 1:237–254, 1998.

[13] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.

[14] Y. A. Korilis, A. A. Lazar, and A. Orda. Capacity allocation under nooncooperative routing. *IEEE Transactions on Automatic Control*, 1998.

[15] Y. A. Korilis, A. A. Lazar, and A. Orda. Architechting noncooperative networks. *IEEE Journal on Selected Areas in Communications*, 13(8), 1999.

[16] Y. A. Korilis, A. A. Lazar, and A. Orda. Avoiding the Braess paradox in noocooperative networks. *Journal of Applied Probability*, 1999. (to appear).

[17] P. Marbach, O. Mihatsch, M. Schulte, and J. Tsisiklis. Reinforcement learning for call admission control and routing in integrated service networks. In *Advances in Neural Information Processing Systems - 10*, pages 922–928. MIT Press, 1998.

[18] A. Orda, R. Rom, and M. Sidi. Minimum delay routing in stochastic networks. *IEEE/ACM Transactions on Networking*, 1(2):187–198, 1993.

[19] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence*, pages 832–838, 1997.

[20] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[21] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

[22] D. H. Wolpert and K. Tumer. An Introduction to Collective Intelligence. In J. M. Bradshaw, editor, *Handbook of Agent technology*. AAAI Press/MIT Press, 1999. to appear.

[23] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems - 11*. MIT Press, 1999.

[24] D. H. Wolpert, K. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In *Proceedings of the Third International Conference of Autonomous Agents*, pages 77–83, 1999.