Distributed Adaptive Control: Beyond Single-Instant, Discrete Variables

David H. Wolpert[1] and Stefan Bieniawski[2]

[1] NASA Ames Research Center, USA, dhw@ptolemy.arc.nasa.gov
[2] Dept. of Aeronautics, Stanford University, USA, stefanb@stanford.edu

Summary. In extensive form noncooperative game theory, at each instant t, each agent i sets its state x_i independently of the other agents, by sampling an associated distribution, $q_i(x_i)$. The coupling between the agents arises in the joint evolution of those distributions. Distributed control problems can be cast the same way. In those problems the system designer sets aspects of the joint evolution of the distributions to try to optimize the goal for the overall system. Now information theory tells us what the separate q_i of the agents are most likely to be if the system were to have a particular expected value of the objective function $G(x_1, x_2, ...)$. So one can view the job of the system designer as speeding an iterative process. Each step of that process starts with a specified value of $E(G)$, and the convergence of the q_i to the most likely set of distributions consistent with that value. After this the target value for $E_q(G)$ is lowered, and then the process repeats. Previous work has elaborated many schemes for implementing this process when the underlying variables x_i all have a finite number of possible values and G does not extend to multiple instants in time. That work also is based on a fixed mapping from agents to control devices, so that the the statistical independence of the agents' moves means independence of the device states. This paper also extends that work to relax all of these restrictions. This extends the applicability of that work to include continuous spaces and Reinforcement Learning. This paper also elaborates how some of that earlier work can be viewed as a first-principles justification of evolution-based search algorithms.

1 Introduction

This paper considers the problem of adaptive distributed control [19, 28, 24]. There are several equivalent ways to mathematically represent such problems. In this paper the representation of extensive form noncooperative game theory is adopted[14, 4, 25, 3, 13]. In that representation, at each instant t each control agent i sets its state x_i^t independently of the other agents, by sampling an associated distribution, $q_i^t(x_i^t)$. In this view the coupling between the agents does not arise directly, via statistical dependencies of the agents' states at the same time t. Rather it arises indirectly, through the stochastic joint evolution of their distributions $\{q_i^t\}$ across time.

More formally, let time be discrete, where at the beginning of each t all control agents simultaneously and independently set their states ("make their moves") by sampling their associated distributions. After they do so any remaining portions of the system (i.e., any stochastic part not being directly set by the control agents) responds to that joint move. Indicate the state of the entire system at time t as y^t. (y^t includes the joint move of the agents, x^t, as well as the state at t of all stochastic elements not directly set by the agents.) So the joint distribution of the moves of the agents at any moment t is given by the product distribution $q^t(x^t) = \prod_i q_i^t(x_i^t)$, and the state of the entire system, given joint move x^t, is governed by $P(y^t \mid x^t)$.

Now in general the observations by agent i of aspects of the system's state at times previous to t will determine q_i^t. In turn, those observations are determined by the previous states of the system. So q_i^t is statistically dependent on the previous states of the entire system, $y^{\{t'<t\}}$. Accordingly, the system can be viewed as a multi-stage noncooperative game among the agents and Nature. Each agent plays mixed strategies $\{q_i^t\}$ at moment t, and Nature's move space at that time consists of those components of the vector y^t not contained in x^t [14, 4, 25, 3, 13]. The interdependence of the agents across time can be viewed as arising through information sets and the like, as usual in game theory.

For pedagogical simplicity, consider the problem of inducing an optimal state y rather than the problem of inducing an optimal sequence of states. What the designer of the system can specify are the laws that govern how the joint mixed strategy q^t gets updated from one stage of the game (i.e., one t) to the next. The goal is to specify such laws that will quickly lead to a good value of an overall objective function of the state of the system, $F(y)$.[3] Note that the agents work in the space of x's; all aspects of the system not directly set by the agents, and in particular all noise processes, are implicitly contained in the distribution $P(y \mid x)$. Tautologically then, in distributed control the goal is to induce a joint strategy $q(x)$ with a good associated value of $E_q(F) = \int dxq(x)E(F \mid x)$. Defining the **world utility** $G(x) \triangleq \int dyF(y)P(y \mid x)$, we can re-express $E_q(F)$ purely in terms of x, as $\int dxq(x)G(x) = E_q(G)$.[4] Once such a q is found, one can sample it to get a final x, and be assured that, on average, the associated F value is low. In other words, such sampling is likely to give us a good value of our objective.

Previous work has has elaborated several iterative schemes for updating product distributions q to monotonically lower $E_q(G)$ [29, 30, 31, 32, 23, 33, 1]. In all of these schemes, each q in the sequence is defined indirectly, as the minimizer of a different G-parameterized Lagrangian, $\mathscr{L}(q)$. Implementing such a sequence of Lagrangian-minimizing q's results in the optimal control policy for the distributed system, i.e., in the q minimizing $E_q(G)$. However while one cannot directly solve for the q minimizing $E_q(G)$ in a distributed manner, as elaborated below one can solve for the q minimizing each $\mathscr{L}(q)$ in a distributed manner. In this way one can find the optimal distributed control policy using a purely distributed algorithm.

Many of these schemes are based on a steepest descent algorithm for each step of minimizing a Lagrangian $\mathscr{L}(q)$. Because the descent is over Euclidean vectors q,

[3] Here we follow the convention that lower F is better. In addition, for simplicity we only consider objectives that depend on the state of the system at a single instant; it is straightforward to relax this restriction.

[4] For simplicity, here we indicate integrals of any sort, including point sums for countable x, with the $\int$ symbol.

these algorithms can be applied whether the x_i are categorical/symbolic, continuous, time-extended, or a mixture of the three. So in particular, they provide a principled way to do "gradient descent over categorical variables".

Not all previously considered algorithms for how to perform the Lagrangian-minimizing step are based on steepest descent. However they do have certain other characteristics in common. One is that the underlying variables x_i all have a finite number of possible values. Another is G does not extend to multiple instants in time. This paper shows how to relax these restrictions, simply by redefining the spaces involved. This allows the previously considered algorithms to be used for continuous spaces, and also implement Reinforcement Learning (RL) [26, 17, 11, 15]. A final shared characteristic is that all of the previously considered algorithms for minimizing the Lagrangians employ a fixed mapping from the moves of agents to the setting of control devices, so that the statistical independence of the agents' moves means independence of the device states. This paper also shows how that restriction can be relaxed, so that independent agents can result in coupled control devices.

The general mathematical framework for casting control and optimization problems in terms of minimizing Lagangians of probability distributions is known as "Probability Collectives". The precise version where the probability distributions are product distributions is known as "Product Distribution" (PD) theory [29]. It has many deep connections to other fields, including bounded rational game theory and statistical physics [30]. As such it serves as a mathematical bridge connecting these disciplines. Some initial experimental results concerning the use of PD theory for distributed optimization and distributed control can be found in [2, 20, 1, 6, 8]. [22, 2, 20, 1, 6, 8].

The next section reviews the salient aspects of PD theory. The section does not consider any of the schemes for the Lagrangian-minimizing step of adaptive distributed control in great detail; the interested reader is directed to the literature. However it is shown in that section how those schemes provide a first-principles justification of certain types of evolution-based search algorithms.

The following section presents two ways to cast PD theory for uncountably infinite x as PD theory for countable X. This allows us to apply all the standard finite-X algorithms even for uncountable X. Experimental tests validating one of those ways of recasting PD theory are presented in [7]. The following section shows how to recast single-instant PD theory to apply to the RL domain, in which y is time-extended. That section considers both episodic and discounted sum RL. The final section considers varying the mapping from the moves of agents to the setting of control devices. Experimental tests validating the usefulness of such variations are presented in [1].

2 Review of PD theory

Say the designer stipulates a particular desired value of $E(G)$, γ. For simplicity, consider the case where the designer makes no other claims concerning the system besides γ and the fact that the joint strategy is a product distribution. Then information theory tells us that the *a priori* most likely q consistent with that information is the one that maximizes entropy subject to that information [10, 21, 16].[5] In other words, of all distributions that agree with the designer's information, that distribution is the "easiest" one to induce by random search.

Given this, one can view the job of the designer of a distributed control system as an iterative equilibration process. In the first stage of each iteration the designer works to speed evolution of the joint strategy to the q with maximal entropy subject to a particular value of γ. Once we have found such a solution we can replace the constraint — replace the target value of $E(G)$ — with a more difficult one, and then repeat the process, with another evolution of q [29].

To formalize this, define the **maxent** Lagrangian by

$$\begin{aligned}\mathscr{L}(q) \triangleq \mathscr{L}_\gamma(q) &\triangleq \beta(E_q(G) - \gamma) - S \\ &= \beta(\int dxq(x)G(x) - \gamma) - S(q),\end{aligned} \tag{1}$$

where $S(q)$ is the Shannon entropy of q, $-\int dxq(x)\ln\frac{q(x)}{\mu(x)}$, and for simplicity we here take the prior μ to be uniform.[6]. Given γ, the associated most likely joint strategy is the q that minimizes $\mathscr{L}(q)$ over all those (q, β) such that the Lagrange parameter β is at a critical point of $\mathscr{L}_\gamma$, i.e., such that $\frac{\partial L}{\partial \beta} = 0$.

Solving, we find that the q_i are related to each other via a set of coupled Boltzmann equations (one for each agent i),

$$q_i^\beta(x_i) \propto e^{-\beta E_{q_{(i)}^\beta}(G|x_i)} \tag{2}$$

where the overall proportionality constant for each i is set by normalization, the subscript $q_{(i)}^\beta$ on the expectation value indicates that it is evaluated according to the distribution $\prod_{j\neq i} q_j$, and β is set to enforce the condition $E_{q^\beta}(G) = \gamma$. Following Nash, we can use Brouwer's fixed point theorem to establish that for any fixed β, there must exist at least one solution to this set of simultaneous equations.

In light of the foregoing, one natural choice for an algorithm that lowers $E_q(G)$ is the repeated iteration of the following step: Start with the q^β matching a current γ value, then lower γ slightly, and end by modifying the old q^β to find the one that matches the new γ. A difficulty with this iterative step is the need to solve for β as a function of γ. However we can use a trick to circumvent this need. Typically if we evaluate $E(G)$ at the solutions q^β, we find that it is a declining function of β. So in following the iterative procedure of equilibrating and then lowering γ we will raise β. Accordingly, we can avoid the repeated matching of β to each successive constraint $E(G) = \gamma$, and simply monotonically increase β instead. This allows us to avoid ever explicitly specifying the values of γ [32].

An alternative interpretation of this iterative scheme is based on prior knowledge of the value of the entropy rather of the expected G. Given this alternative prior

[5] In light of how limited the information is here, the algorithms presented below are best-suited to "off the shelf" uses; incorporating more prior knowledge allows the algorithms to be tailored more tightly to a particular application.

[6] Throughout this paper the terms in any Lagrangian that restrict distributions to the unit simplices are implicit. The other constraint needed for a Euclidean vector to be a valid probability distribution is that none of its components are negative. This will not need to be explicitly enforced in the Lagrangian here.

knowledge, we can recast the designer's goal as finding the q that is consistent with that knowledge that has minimal $E(G)$. This again leads to Eq.'s 1 and 2. Now raising β is cast as lowering the (never-specified) prior knowledge of the entropy value rather than the (never-specified) prior knowledge of $E(G)$.

Simulated annealing is an example of this approach, where rather than work directly with q, one works with random samples of it formed via the Metropolis random walk algorithm [18, 12, 9, 27]. There is no *a priori* reason to use such an inefficient means of manipulating q however. In [32] for example one works with q directly instead. This results in an algorithm that is not simply "probabilistic" in the sense that the updating of its variables is stochastic (as in simulated annealing). Rather the very entity being updated is a probability distribution.

Another advantage of casting the problem directly in terms of the maxent Lagrangian is that one can even avoid the need to explicitly stipulate an annealing schedule. In the usual way, first order methods can be used to find the saddle point of the Lagrangian, e.g., by performing steepest ascent of $\mathscr{L}$ in the Lagrange parameter β while performing a descent in q [7].

In many situations one should use a modification of the maxent Lagrangian. Whenever one has extra prior knowledge about the problem domain, that should be used to modify the use of entropy as (in statistics terminology) a regularizer. This leads to Bayesian formulations [32]. Similarly, if one has constraints $\{f_i(x) = 0\}$, the Lagrangian has to be modified to accout for them. The most naive way of doing this is to simply cast the constraints as Lagrange penalty terms $\{E(f_i) = 0\}$ and add those terms to the Lagrangian, in the usual way [32, 8] [8].

2.1 How to find minima of the Lagrangian

Consider the situation where each x_i can take on a finite number of possible values, $|X_i|$, and we are interested in the unconstrained maxent Lagrangian. Say we are iteratively evolving q to minimize $\mathscr{L}$ for some fixed β, and are currently at some point q in $\mathcal{Q}$, the space of product distributions (i.e., in the Cartesian product of unit simplices). Using Lemma 1 of [32], we can evaluate the direction from q within $\mathcal{Q}$ that, to first order, will result in the largest drop in the value of $\mathscr{L}(q)$:

$$\frac{\partial^R \mathscr{L}(q)}{\partial^R q_i(x_i = j)} = u_i(j) - \sum_{x'_i} u_i(x'_i)/|X_i|, \tag{3}$$

where $u_i(j) \triangleq \beta E(G \mid x_i = j) + \ln[q_i(j)]$, and the symbol ∂^R indicates that we do not mean the indicated partial derivative, formally speaking, but rather the indicated component of the 1st-order descent vector [9].

[7] Formally, since the maxent Lagrangian is not convex, we have no guarantee that the duality gap is zero, and therefore no guarantee about saddlepoints. Nonetheless, just as in other domains, first order methods here seem to work well in practice.

[8] Note though that since the gradient of entropy is infinite at the border of the unit simplex, we are guaranteed that no component of q will ever exactly equal 0, which typically means that the constraints $\{f_i(x) = 0\}$ will never be satisfied with probability exactly 1.

[9] Formally speaking, the partial derivative is given by $u_i(j)$. Intuitively, the reason for subtracting $\sum_{x'_i} u_i(x'_i)/|X_i|$ is to keep the distribution in the set of all possible probability distributions over x, $\mathcal{P}$.

Eq. 3 specifies the change that each agent should make to its distribution to have them jointly implement a step in steepest descent of the maxent Lagrangian. These updates are completely distributed, in the sense that each agent's update at time t is independent of any other agents' update at that time. Typically at any t each agent i knows $q_i(t)$ exactly, and therefore knows $\ln[q_i(j)]$. However often it will not know G and/or the $q_{(i)}$. In such cases it will not be able to evaluate the $E(G \mid x_i = j)$ terms in Eq. 3 in closed form.

One way to circumvent this problem is to have those expectation values be simultaneously estimated by all agents by repeated Monte Carlo sampling of q to produce a set of $(x, G(x))$ pairs. Those pairs can then be used by each agent i to estimate the values $E(G \mid x_i = j)$, and therefore how it should update its distribution. In the simplest version of this, an update to q only occurs once every $\mathscr{M}$ time-steps. In this scheme only the samples $(x, G(x))$ formed within a block of $\mathscr{M}$ successive time-steps are used at the end of that block by the agents to update their distributions (according to Eq. 3).

There are numerous other schemes besides gradient descent for finding minima of the Lagrangian. One of these is a second order version of steepest descent, constrained to operate over $\mathcal{Q}$. This scheme, called "Nearest Newton" [32], starts by calculating the point $p \in \mathcal{P}$ that one should step to from the current distribution q, if one were to use Newton's method to descend the Lagrangian. Now in general that p is not a product distribution. So we instead find q', the $q \in \mathcal{Q}$ that is closest (as measured by Kullbach-Leibler distance) to that point p; the step actually taken is to q'.

This step from the current q turns out to be indentical to the gradient descent step, just with an extra multiplicative factor of $q_i(x_i = j)$ multiplying each associated component of that gradient descent step:

$$\Delta(q_i(x_i = j)) \;\propto\; q_i(j)u_i(j) \;-\; \sum_{x'_i} \frac{q_i(x'_i)u_i(x'_i)}{|X_i|}, \tag{4}$$

where u_i is as defined just below Eq. 3, and the proportionality constant is the step size.

In the continuum time limit, this step rule reduces to the replicator equation of evolutionary game theory, only with an entropic term added in [31]. (Intuitively, that entropic term ensures the evolution explores sufficiently.) This connection can be viewed as a first-principles justification for (particular versions of) evolution-based search algorithms, e.g., genetic algorithms. To be precise, say we have a biological population of many "genes", each specifying a value x, and an associated "fitness function" $G(x)$. Have the frequency of each gene in the population be updated via the replicator dynamics, as usual in evolutionary game theory. We can justify this evolution-based search algorithm as the $\beta \to \infty$ limit of Nearest Newton for the case of a single agent with moves x. By allowing $\beta < \infty$, we can extend those evolution-based search algorithms in a principled manner.

A final example of a Lagrangian descent scheme, which is analogous to block relaxation, is "Brouwer updating" [31]. In that kind of updating one or more agents simultaneously jump to their optimal distribution, as given by Eq. 2 (with β rather than γ specified, as discussed above). It turns out that if the expectations defining the Brouwer updating are "exponentially aged" to reflect nonstationarity, then in the continuum time limit Brouwer updating becomes identical to Nearest Newton.

The aging constant in Brouwer updating turns out to be identical to the step size in Nearest Newton.

All of the update schemes can be used so long as each agent i knows or can estimate q_i together with $E_{q_{(i)}}(G \mid x_i) = E(G \mid x_i)$ for all of its moves x_i. No other quantities are involved.

3 Semicoordinate transformations

3.1 Motivation

Consider a multi-stage game like chess, with the stages (i.e., the instants at which one of the players makes a move) delineated by t. In game theoretic terms, the "strategy" of a player is the board-configuration $\rightarrow$ response rule it adopts before play starts [14, 4, 25, 3, 13]. More generally, in a multi-stage game like chess the strategy of player i, x_i, is the set of t-indexed maps taking what that player has observed in the stages $t' < t$ into its move at stage t. Formally, this set of maps is called player i's **normal form** strategy.

The joint strategy of the two players in chess sets their joint move-sequence, though in general the reverse need not be true. In addition, one can always find a joint strategy to result in any particular joint move-sequence. Now typically at any stage there is overlap in what the players have observed over the preceding stages. This means that even if the players' strategies are statistically independent (being separately set before play started), their move sequences are statistically coupled. In such a situation, by parameterizing the space Z of joint-move-sequences z with joint-strategies x, we shift our focus from the coupled distribution $P(z)$ to the decoupled product distribution, $q(x)$. This is the advantage of casting multi-stage games in terms of normal form strategies.

More generally, given two spaces with values $\{x\}$ and $\{z\}$, any onto mapping $\zeta : x \rightarrow z$, not necessarily invertible, is called a **semicoordinate system**. The identity mapping $z \rightarrow z$ is a trivial example of a semicoordinate system. Another example is the mapping from joint-strategies in a multi-stage game to joint move-sequences. In other words, changing the representation space of a multi-stage game from move-sequences z to strategies x is a semicoordinate transformation of that game. Yet another example is where $Z = \mathbb{B}^K$ for some K, while $X = \{1, 2, \ldots, b\}$ where $b \geq 2^K$. In this last example X has greater cardinality than Z, yet its elements are 1-dimensional, whereas Z's elements are K-dimensional.

Intuitively, a semi-coordinate transformation is a reparameterization of how a game — a mapping from joint moves to associated payoffs — is represented. So we can perform a semicoordinate transformation even in a single-stage game. Say we restrict attention to distributions over X that are product distributions. Then changing $\zeta(.)$ from the identity map to some other function means that the players' moves are no longer independent. After the transformation their move choices — the components of z — are statistically coupled, even though we are considering a product distribution.

Formally, this is expressed via the standard rule for transforming probabilities,

$$P_Z(z \in Z) = \int dx P_X(x) \delta(z - \zeta(x)) \triangleq \zeta(P_X), \tag{5}$$

where P_X and P_Z are the distributions across X and Z, respectively. To see what this rule means geometrically, let $\mathcal{P}$ be the space of all distributions (product or otherwise) over Z. Recall that $\mathcal{Q}$ is the space of all product distributions over X, and let $\zeta(\mathcal{Q})$ be the image of $\mathcal{Q}$ in $\mathcal{P}$. Then by changing $\zeta(.)$, we change that image; different choices of $\zeta(.)$ will result in different manifolds $\zeta(\mathcal{Q})$.

As an example, say we have two players, with two possible moves each. So z consists of the possible joint moves, labeled $(1,1), (1,2), (2,1)$ and $(2,2)$. Have $X = Z$, and choose $\zeta(1,1) = (1,1)$, $\zeta(1,2) = (2,2)$, $\zeta(2,1) = (2,1)$, and $\zeta(2,2) = (1,2)$. Say that q is given by $q_1(x_1 = 1) = q_2(x_2 = 1) = 2/3$. Then the distribution over joint-moves z is $P_Z(1,1) = P_X(1,1) = 4/9$, $P_Z(2,1) = P_Z(2,2) = 2/9$, $P_Z(1,2) = 1/9$. So $P_Z(z) \neq P_Z(z_1)P_Z(z_2)$; the moves of the players are statistically coupled, even though their strategies x_i are independent.

Such coupling of the players' moves can be viewed as a manifestation of sets of potential binding contracts. To illustrate this return to our two player example. Each possible value of a component x_i determines a pair of possible joint moves. For example, setting $x_1 = 1$ means the possible joint moves are $(1,1)$ and $(2,2)$. Accordingly such a value of x_i can be viewed as a set of proffered binding contracts. The value of the other components of x determines which contract is accepted; it is the intersection of the proffered contracts offered by all the components of x that determines what single contract is selected. Continuing with our example, given that $x_1 = 1$, whether the joint-move is $(1,1)$ or $(2,2)$ (the two options offered by x_1) is determined by the value of x_2.

To relate semicoordinates to distributed control we have to fix some notation. To maintain consistency with the discussion of maxent Lagrangians, we will have product distributions $q(x \in X) \in \mathcal{Q}_{\mathcal{X}}$. Also as before, to allow stochasticity, we write the ultimate space of interest as y with associated cost function $F(y)$. This means that x sets z which stochastically sets y:

$$E_q(F) = \int dy\ P(y)F(y) = \int dz\ P(z)G(z) = \int dx\ q(x)G(x) \tag{6}$$

where

$$\begin{aligned} G(z) &\triangleq \int dy\ F(y)P(y \mid z) \qquad (7)\\ &= \int dx\ G(x)\delta(\zeta(x) - z); \\ G(x) &= \int dy\ F(y)P(y \mid x) \qquad (8)\\ &= \int dy\ F(y)P(y \mid \zeta(x)). \end{aligned}$$

Finally, note that so long as $|\mathcal{X}| \geq |\mathcal{Z}|$, any distribution $P_{\mathcal{Z}}(z)$ is the image under ζ of at least one distribution $P_{\mathcal{X}}(x)$. In other words, any distribution over $\mathcal{Z}$ can be converted, in at least one way, into a distribution over $\mathcal{X}$. This holds even if neither $P_{\mathcal{X}}$ nor $P_{\mathcal{Z}}$ is a product distribution, and is due to the fact that ζ is surjective.[10]

[10] If need be, for each $z \in \mathcal{Z}$, we can set $P_{\mathcal{X}}(x) = P_{\mathcal{Z}}(\zeta(x))$ for exactly one of the $x \in \zeta^{-1}(z)$, and set $P_{\mathcal{X}}(x) = 0$ for all the other x in z's pre-image. It's the surjectivity of ζ that guarantees that that pre-image is non-empty.

3.2 Representational properties

Binding contracts are a central component of cooperative game theory. In this sense, semicoordinate transformations can be viewed as a way to convert noncooperative game theory into a form of cooperative game theory. Indeed, any cooperative mixed strategy can be cast as a non-cooperative game mixed strategy followed by an appropriate semicoordinate transformation. Formally, any P_Z, no matter what the coupling among its components, can be expressed as $\zeta(P_X)$ for some product distribution P_X and associated $\zeta(.)$ [11]

Less trivially, given any model class of distributions $\{P_Z\}$, there is an X and associated $\zeta(.)$ such that $\{P_Z\}$ is identical to $\zeta(\mathcal{Q}_X)$. Formally this is expressed in a result concerning Bayes nets. For simplicity, restrict attention to finite Z. Order the components of Z from 1 to N. For each index $i \in \{1, 2, \ldots, N\}$, have the **parent function** $\mathcal{P}(i, z)$ return a subset of the components of z with index greater than i, where the choice of what components depends only on i, and not on z. So for example, with $N > 5$, we could have $\mathcal{P}(1, z) = (z_2, z_5) \forall z$. Another possibility is that $\mathcal{P}(1, z)$ is the empty set, independent of z.

Let $A(\mathcal{P})$ be the set of all probability distributions P_Z that obey the conditional dependencies implied by $\mathcal{P}$: $\forall\ P_Z \in A(\mathcal{P}), z \in Z$,

$$P_Z(z) = \prod_{i=1}^{N} P_Z(z_i \mid \mathcal{P}(i, z)). \tag{9}$$

(By definition, if $\mathcal{P}(i, z))$ is empty, $P_Z(z_i \mid \mathcal{P}(i, z))$ is just the i'th marginal of P_Z, $P_Z(z_i)$.) Note that any distribution P_Z is a member of $A(\mathcal{P})$ for some $\mathcal{P}$ — in the worst case, just choose the exhaustive parent function $\mathcal{P}(i, z) = \{z_j : j > i\}$.

For any choice of $\mathcal{P}$ there is an associated set of distributions $\zeta(\mathcal{Q}_X)$ that equals $A(\mathcal{P})$ exactly:

Theorem 1: Define the components of X using multiple indices: For all $i \in \{1, 2, \ldots, N\}$ and possible associated values (as one varies over $z \in Z$) of the vector $\mathcal{P}(i, z)$, there is a separate component of x, $x_{i;\mathcal{P}(i,z)}$. Have this component's allowed values be the those of z_i. Define $\zeta(.)$ recursively, starting at $i = N$ and working to lower i, by the following rule: $\forall\ i \in \{1, 2, \ldots, N\}$,

$$z_i \ = \ [\zeta(x)]_i \ = \ x_{i;\mathcal{P}(i,z)}.$$

Then $A(\mathcal{P}) = \zeta(\mathcal{Q}_X)$.

Proof: First note that by definition of parent functions, due to the fact that we're iteratively working down from higher i's to lower ones, $\zeta(x)$ is properly defined. Next plug that definition into Eq. 5. For any particular x and associated $z = \zeta(x)$, those components of x that do not "match" z by having their second index equal $\mathcal{P}(i, z)$ get integrated out. After this the integral reduces to

$$P_Z(z) = \prod_{i=1}^{N} P_X([x_{i;\mathcal{P}(i,z)}] = z_i),$$

[11] In the worst case, one can simply choose X to have a single component, with $\zeta(.)$ a bijection between that component and the vector z — trivially, any distribution over such an X is a "product distribution".

i.e., is exactly of the form stipulated in Eq. 9. Accordingly, for any fixed x and associated $z = \zeta(x)$, ranging over the set of all values between 0 and 1 for each of the distributions $P_X([x_{i;\mathcal{P}(i,z)} = z_i)$ will result in ranging over all values for the distribution $P_Z(z)$ that are of the form stipulated in Eq. 9. This must be true for all x. Accordingly, $\zeta(\mathcal{Q}_X) \subseteq A(\mathcal{P})$. The proof that $A(\mathcal{P}) \subseteq \zeta(\mathcal{Q}_X)$ goes similarly: For any given P_Z and z, simply set $P_X([x_{i;\mathcal{P}(i,z)}] = z_i)$ for all the independent components $x_{i;\mathcal{P}(i,z)}$ of x and evaluate the integral in Eq. 5. **QED.**

Intuitively, each component of x in Thm. 1 is the conditional distribution $P_Z(z_i \mid \mathcal{P}(i,z))$ for some particular instance of the vector $\mathcal{P}(i,z))$. Thm. 1 means that in principle we never need consider coupled distributions. It suffices to restrict attention to product distributions, so long as we use an appropriate semicoordinate system. In particular, mixture models over Z can be represented this way.

The Kullback Leibler (KL) distance from the distribution $a(y)$ to $b(y)$ is defined as

$$KL(a||b) \triangleq -\int dy \; a(y)\ln[\frac{b(y)}{a(y}].$$

It is a common way of measuring how far apart two distributions are. It is straightforward to see that the product distribution $a(y) \triangleq \prod_i a_i(y_i)$ with minimal KL distance to b is given by $a_i(y_i) = \int dy_{-i} \; b(y_i, y_{-i})$, where y_{-i} is all components of y other than i. We can use semi-coordinate transformations to establish a generalization of this that is related to Thm. 1:

Corollary 1: Of the Bayes nets $q(z)$ that have parent function $\mathcal{P}$, the one with minimal KL to $p(z)$ is given by

$$q_{\mathcal{Z}}(z_i \mid \mathcal{P}(i,z)) = p(z_i \mid \mathcal{P}(i,z)) \;\; \forall i, z.$$

Proof: Perform a semi-coordinate transformation like that discussed in Thm. 1 on both q and p. So $q_{\mathcal{X}}$ is a product distribution. Use our freedom in setting $p_{\mathcal{X}}$ to ensure that $\forall z, \forall x \in \zeta^{-1}(z)$, $p_{\mathcal{X}}(x)/q_{\mathcal{X}}(x)$ has the same value. (Note that in general, this means that if we change $q_{\mathcal{X}}$, we must also change $p_{\mathcal{X}}$.) Due to the rule for transforming probabilities, that value of the ratio must be $p_{\mathcal{Z}}(\zeta(x))/q_{\mathcal{Z}}(\zeta(x))$. Now write

$$\begin{aligned}
-\int dz \; q_{\mathcal{Z}}(z)\ln[\frac{p_{\mathcal{Z}}(z)}{q_{\mathcal{Z}}(z)}] &= -\int dz \int dx \; \delta(z - \zeta(x))q_{\mathcal{X}}(x) \ln[\frac{p_{\mathcal{Z}}(z)}{q_{\mathcal{Z}}(z)}] \\
&= -\int dx \int dz \; \delta(z - \zeta(x))q_{\mathcal{X}}(x) \ln[\frac{p_{\mathcal{Z}}(z)}{q_{\mathcal{Z}}(z)}] \\
&= -\int dx \; q_{\mathcal{X}}(x) \ln[\frac{p_{\mathcal{Z}}(\zeta(x))}{q_{\mathcal{Z}}(\zeta(x))}] \\
&= -\int dx \; q_{\mathcal{X}}(x) \ln[\frac{p_{\mathcal{X}}(x)}{q_{\mathcal{X}}(x)}]
\end{aligned}$$

where the last step used our stipulation on the ratio of $p_{\mathcal{X}}$ and $q_{\mathcal{X}}$. In other words, the KL distance from q to p is the same in both $\mathcal{Z}$ and $\mathcal{X}$.
This means that the $q_{\mathcal{Z}}$ that minimizes ($\mathcal{Z}$-space) KL distance to $p_{\mathcal{Z}}$ is the image under ζ of the $q_{\mathcal{X}}$ that minimizes ($\mathcal{X}$-space) KL distance to $p_{\mathcal{X}}$. However since $q_{\mathcal{X}}$

is a product distribution, the $q_{\mathcal{X}}$ that minimizes ($\mathcal{X}$-space) KL distance to $p_{\mathcal{X}}$ is just given by the product of the associated ($\mathcal{X}$-space) marginals of $p_{\mathcal{X}}$. Under our semi-coordinate transformation, those marginals of $p_{\mathcal{X}}$ get mapped to the associated Bayes net conditional distributions in $\mathcal{Z}$. **QED**

Coroll. 1 can be extended in a straightforward way to other graphical models besides Bayes nets. For example, it can be extended to models having "hidden variables", like mixture models. So long as the appropriate semi-coordinate transformation changes the graphical model $q_{\mathcal{X}}$ into a product distribution, minimal KL distance arises when each component of the product $q_{\mathcal{X}}$ equals the associated marginal of $p_{\mathcal{X}}$. Translating this into $\mathcal{Z}$ space, , each term in the optimal graphical model $q_{\mathcal{Z}}$ is just associated conditional of $p_{\mathcal{Z}}$).

3.3 Maxent Lagrangians over X rather than Z

While the distribution over X uniquely sets the distribution over Z, the reverse is not true. However so long as our Lagrangian directly concerns the distribution over X rather than the distribution over Z, by minimizing that Lagrangian we set a distribution over Z. In this way we can minimize a Lagrangian involving product distributions, even though the associated distribution in the ultimate space of interest is not a product distribution.

The Lagrangian we choose over X should depend on our prior information, as usual. If we want that Lagrangian to include an expected value over Z (e.g., of a cost function), we can directly incorporate that expectation value into the Lagrangian over X, since expected values in X and Z are identical: $\int dz P_Z(z)A(z) = \int dx P_X(x)A(\zeta(x))$ for any function $A(z)$. (Indeed, this is the standard justification of the rule for transforming probabilities, Eq. 5.)

However other functionals of probability distributions can differ between the two spaces. This is especially common when $\zeta(.)$ is not invertible, so X is larger than Z. In particular, while the expected cost term is the same in the X and Z maxent Lagrangians, this is not true of the two entropy terms in general; typically the entropy of a $q \in \mathcal{Q}$ will differ from that of its image, $\zeta(q) \in \zeta(\mathcal{Q})$ in such a case.

More concretely, the fully formal definition of entropy includes a prior probability μ: $S_X \triangleq \int dx p(x) \ln(\frac{p(x)}{\mu(x)})$, and similarly for S_Z. So long as $\mu(x)$ and $\mu(z)$ are related by the normal laws for probability transformations, as are $p(x)$ and $p(z)$, then *if the cardinalities of X and Z are the same, $S_Z = S_X$* [12]. When the cardinalities of the spaces differ though (e.g., when X and Z are both finite but with differing numbers of elements), this need no longer be the case. The following result bounds how much the entropies can differ in such a situation:

Theorem 2: For all $z \in Z$, take $\mu(x)$ to be uniform over all x such that $\zeta(x) = z$. Then for any distribution $p(x)$ and its image $p(z)$,

---

[12] For example, if $X = Z = \mathbb{R}^m$, then $\ln[\frac{p(\zeta(x))}{\mu(\zeta(x))}] = \ln[\frac{p(x)J_\zeta(x)}{\mu(x)J_\zeta(x)}] = \ln[\frac{p(x)}{\mu(x)}]$, where $J_\zeta(x)$ is the determinant of the Jacobian of $\zeta(.)$ evaluated at x. Accordingly, as far as transforming from X to Z is concerned, entropy is just a conventional expectation value, and therefore has the same value whichever of the two spaces it is evaluated in.

$$-\int dz\ p(z)\ \ln(K(z))\ \leq\ S_X - S_Z\ \leq\ 0,$$

where $K(z) \triangleq \int dx\delta(z - \zeta(x))$. (Note that for finite X and Z, $K(z) \geq 1$, and counts the number of x with the same image z.) If we ignore the μ terms in the definition of entropy, then instead we have

$$0\ \leq\ S_X - S_Z\ \leq\ -\int dz\ p(z)\ \ln(K(z)).$$

Proof: Write

$$\begin{aligned} S_X &= -\int dz \int dx\ \delta(z - \zeta(x))\ p(x)\ \ln[\frac{p(x)}{\mu(x)}] \\ &= -\int dz \int dx\ \delta(z - \zeta(x))\ p(x)\ \times \\ &\qquad\qquad (\ln[\frac{p(x)}{d(z)\mu(x)} + \ln[d(z)]) \\ &= -\int dz\ p(z)\ln[d(z)]\ - \\ &\quad \int dz \int dx\ \delta(z - \zeta(x))\ p(x)\ \ln[\frac{p(x)}{d(z)\mu(x)}] \end{aligned}$$

where $d_z \triangleq \int dx\ \delta(z - \zeta(x))\ \frac{p(x)}{\mu(x)}$. Define μ^z to be the common value of all $\mu(x)$ such that $\zeta(x) = z$. So $\mu(z) = \mu^z K(z)$ and $p(z) = \mu^z d(z)$. Accordingly, expand our expression as

$$\begin{aligned} S_X &= -\int dz\ p(z)\ \ln[\frac{p(z)}{\mu(z)}]\ -\ \int dz\ p(z)K(z)\ - \\ &\quad \int dz \int dx\ \delta(z - \zeta(x))\ p(x)\ \ln[\frac{p(x)}{d(z)\mu(x)}] \\ &= S_Z - \int dz\ p(z)K(z)\ + \\ &\quad \int dz\ p(z)\ (-\int dx\ \delta(z - \zeta(x))\ \frac{p(x)}{p(z)}\ \ln[\frac{p(x)}{p(z)}]). \end{aligned}$$

The x-integral of the right-hand side of the last equation is just the entropy of normalized the distribution $\frac{p(x)}{p(z)}$ defined over those x such that $\zeta(x) = z$. Its maximum and minimum are $\ln[K(z)]$ and 0, respectively. This proves the first claim. The second claim, where we "ignore the μ terms", is proven similarly. **QED.**

In such cases where the cardinalities of X and Z differ, we have to be careful about which space we use to formulate our Lagrangian. If we use the transformation $\zeta(.)$ as a tool to allow us to analyze bargaining games with binding contracts, then the direct space of interest is actually the x's (that is the place in which the players make their bargaining moves). In such cases it makes sense to apply all the analysis of the preceding sections exactly as it is written, concerning Lagrangians and distributions over x rather than z (so long as we redefine cost functions to implicitly pre-apply the mapping $\zeta(.)$ to their arguments). However if we instead use $\zeta(.)$ simply as a way of establishing statistical dependencies among the moves of the players, it may make sense to include the entropy correction factor in our x-space Lagrangian.

An important special case is where the following three conditions are met: Each point z is the image under $\zeta(.)$ of the same number of points in x-space, n; $\mu(x)$ is uniform (and therefore so is $\mu(z)$); and the Lagrangian in x-space, $\mathscr{L}_x$, is a sum of expected costs and the entropy. In this situation, consider a z-space Lagrangian, $\mathscr{L}_z$, whose functional dependence on P_z, the distribution over z's, is identical to the dependence of $\mathscr{L}_x$ on P_x, except that the entropy term is divided by n [13]. Now the minimizer $P^*(x)$ of $\mathscr{L}_x$ is a Boltzmann distribution in values of the cost function(s). Accordingly, for any z, $P^*(x)$ is uniform across all n points $x \in \zeta^{-1}(z)$ (all such x have the same cost value(s)). This in turn means that $S(\zeta(P_x)) = nS(P_z)$. So our two Lagrangians give the same solution, i.e., the "correction factor" for the entropy term is just multiplication by n.

3.4 Exploiting semicoordinate transformations

This subsection illustrates some way to exploit semicoordinate transformations to facilitate descent of the Lagrangian. To illustrate the generality of the arguments, situations where one has to to use Monte Carlo estimates of conditional expectation values to descend the shared Lagrangian (rather than evaluate them closed-form) will be considered.

Say we are currently at a local minimum $q \in \mathcal{Q}$ of $\mathscr{L}$. Usually we can break out of that minimum by raising β and then resuming the updating; typically changing β changes $\mathscr{L}$ so that the Lagrange gaps are nonzero. So if we want to anneal β anyway (e.g., to find a minimum of the shared cost function G), it makes sense to do so to break out of any local minima.

There are many other ways to break out of local minima without changing the Lagrangian (as we would if we changed β, for example) [32]. Here we show how to use semicoordinate transformations to do this. As explicated below, they also provide a general way to lower the value of the Lagrangian, whether or not one has local minimum problems.

Say our original semicoordinate system is $\zeta^1(.)$. Switch to a different semicoordinate system $\zeta^2(.)$ for Z and consider product distributions over the associated space X^2. Geometrically, the semicoordinate transformation means we change to a new submanifold $\zeta^2(\mathcal{Q}) \subset \mathcal{P}$ without changing the underlying mapping from $p(z)$ to $\mathscr{L}_Z(p)$. In practice, when the cardinalities of X^1 and X^2 are the same, often we keep the product distribution unchanged when we make the transformation. This means that the entropy term in the Lagrangian will be unchanged by the transformation, while expected G — the ultimate object of interest — will change.

As a simple example, say ζ^2 is identical to ζ^1 except that it joins two components of x into an aggregate semicoordinate. Since after that change we can have statistical dependencies between those two components, the product distributions over X^2, $\zeta^2(\mathcal{Q}_{X^2})$, map to a superset of $\zeta^1(\mathcal{Q}_{X^1})$. Typically the local minima of that superset do not coincide with local minima of $\zeta^1(\mathcal{Q}_{X^1})$. So this change to X^2 will indeed break out of the local minimum, in general.

[13] For example, if $\mathscr{L}_x(P_x) = \beta E_{P_x}(G(\zeta(.))) - S(P_x)$, then $\mathscr{L}_z(P_z) = \beta E_{P_z}(G(.)) - S(P_z)/n$, where P_x and P_z are related as in Eq. 5.

More care is needed when working with more complicated semicoordinate transformations. Say before the transformation we are at a point $p^* \in \zeta^1(\mathcal{Q}_{X^1})$. Then in general p^* will not be in the new manifold $\zeta^2(\mathcal{Q}_{X^2})$, i.e., p^* will not correspond to a product distribution in our new semicoordinate system. (This reflects the fact that semicoordinate transformations couple the players.) Accordingly, we must change from p^* to a new distribution when we change the semicoordinate system.

To illustrate this, say that the semicoordinate transformation is bijective. Formally, this means that $X^2 = X^1 \triangleq X$ and $\zeta^2(x) = \zeta^1(\xi(x))$ for a bijective $\xi(.)$. Have $\xi(.)$, the mapping from X^2 to X^1, be the identity map for all but a few of the M total components of X, indicated as indices $1 \rightarrow n$. Intuitively, for any fixed $x^2_{n+1\rightarrow M} = x_{n+1\rightarrow M}$, the effect of the semicoordinate transformation to $\zeta^2(.)$ from $\zeta^1(.)$ is merely to "shuffle" the associated mapping taking semicoordinates $1 \rightarrow n$ to Z, as specified by $\xi(.)$. Moreover, since $\xi(.)$ is a bijection, the maxent Lagrangians over X^1 and X^2 are identical: $\mathscr{L}_{X^1}(\xi(p^{X^2})) = \mathscr{L}_{X^2}((p^{X^2}))$.

Now say we set $q^{X^2}_{n+1\rightarrow M} = q^{X}_{n+1\rightarrow M}$. This means we can estimate the expectations of G conditioned on possible $x^2_{1\rightarrow n}$ from the Monte Carlo samples conditioned on $\xi(x^2_{1\rightarrow n})$. In particular, for any $\xi(.)$ we can estimate $E(G)$ as $\int dx^2_{1\rightarrow n} p^{X^2}(x^2_{1\rightarrow n}) E(G \mid \xi(x^2_{1,\ldots,n}))$ in the usual way. Now entropy is the sum of the entropy of semicoordinates $n+1 \rightarrow M$ plus that of semicoordinates $1 \rightarrow n$. So for any choice of $\xi(.)$ and $q^{X^2}_{1\rightarrow n}$, we can approximate $\mathscr{L}_X = \mathscr{L}_{X^2}$ as (our associated estimate of) $E(G)$ minus the entropy of $p^{X^2}_{1\rightarrow n}$, minus a constant unaffected by choice of $\xi(.)$.

So for finite and small enough cardinality of the subspace $|X_{1\rightarrow n}|$, we can use our estimates $E(G \mid \xi(x^2_{1\rightarrow n}))$ to search for the "shuffling" $\xi(.)$ and distribution $q^{X^2}_{1\rightarrow n}$ that minimizes $\mathscr{L}^X$ [14]. In particular, say we have descended $\mathscr{L}_X$ to a distribution $q^{X^1}(x) = q^*(x)$. Then we can set $q^{X^2} = q^*$, and consider a set of of "shuffling $\xi(.)$". Each such $\xi(.)$ will result in a different distribution $q^{X^1}(x) = q^{X^2}(\xi^{-1}(x)) = q^*(\xi^{-1}(x))$. While those distributions will have the same entropy, typically they will have different (estimates of) $E(G)$ and accordingly different local minima of the Lagrangian.

Accordingly, searching across the $\xi(.)$ can be used to break out of a local minimum. However since $E(G)$ changes under such transformations even if we are not at a local minimum, we can instead search across $\xi(.)$ as a new way (in addition to those discussed above) for lowering the value of the Lagrangian. Indeed, there is always a bijective semicoordinate transformation that reduces the Lagrangian: simply choose $\xi(.)$ to rearrange the $G(x)$ so that $G(x) < G(x') \Leftrightarrow q(x) < q(x')$. In addition one can search for that $\xi(.)$ in a distributed fashion, where one after the other each agent i rearranges its semicoordinate to shrink $E(G)$. Furthermore to search over semicoordinate systems we don't need to take any additional samples of G. (The existing samples can be used to estimate the $E(G)$ for each new system.) So the search can be done off-line.

To determine the semicoordinate transformation we can consider other factors besides the change in the value of the Lagrangian that immediately arises under the transformation. We can also estimate the amount that subsequent evolution under the new semicoordinate system will decrease the Lagrangian. We can estimate that

[14] penalizing by the bias^2 plus variance expression if we intend to do more Monte Carlo — see [29].

subsequent drop in a number of ways: the sum of the Lagrangian gaps of all the agents, gradient of the Lagrangian in the new semicoordinate system, etc.

3.5 Distributions over semicoordinate systems

The straightforward way to implement these kinds of schemes for finding a good semicoordinate systems is via exhaustive search, hill-climbing, simulated annealing, or the like. Potentially it would be very useful to instead find a new semicoordinate system using search techniques designed for continuous spaces. When there are a finite number of semicoordinate systems (i.e., finite X and Z) this would amount to using search techniques for continuous space to optimize a function of a variable having a finite number of values. However we now know how to do that: use PD theory. In the current context, this means placing a product probability distribution over a set of variables parameterizing the semicoordinate system, and then evolving the probability distribution.

More concretely, write

$$\begin{aligned}\mathscr{L}(q) &= \beta \sum_{\theta} \sum_{x} P(\theta) \prod_{i=1}^{N} q_i(x_i) G(\zeta(x,\theta)) + S(q) \qquad (10)\\ &= \beta \sum_{\theta} \sum_{x} \prod_{i=1}^{N} q_i(x_i) P(\theta) G(\zeta(x,\theta)) + S(q)\end{aligned}$$

where θ is a parameter on the semicoordinate system. We can rewrite this using an additional semicoordinate transformation, as

$$\mathscr{L}(q^*) = \beta \sum_{x^*} \prod_{i=1}^{N+1} q_i^*(x_i^*) G(\zeta(x^*)) + S(q^*) \qquad (11)$$

where $x_i^* = x_i$ for all i up to N, and $x_{N+1}^* = \theta$. (As usual, depending on what space we cast our Lagrangian in, the entropy can either have the argument of the entropy term starred — as here — or not.)

Intuitively, this approach amounts to introducing a new coordinate/agent, whose "job" is to set the semicoordinate system governing the mapping from the other agents to a z value. This provides an alternative to periodically (e.g., at a local minimum) picking a set of alternative semicoordinate systems and estimating which gives the biggest drop in the overall Lagrangian. We can instead use Nearest Newton, Brouwer updating, or what have you, to continuously search for the optimal coordinate system as we also search for the optimal x. The tradeoff, of course, is that by introducing an extra coordinate/agent, we raise the noise level all the original semicoordinates experience. (This raises the issue of what best parameterization of $\zeta(.)$ to use, an issue not addressed here.)

4 PD theory for uncountable Z

In almost all computational algorithms for finding minima, and in particular in the algorithms considered above, we can only modify a finite set of real numbers from one step to the next. When Z is finite, we accomodate this by having the real numbers be the values of the components of the q_i. But how can we use a computational algorithm to find a minimum of the maxent Lagrangian when Z is uncountable?

One obvious idea is to have the real numbers our algorithm works with parameterize p differently from how they do with product distributions. For example, rather than product distributions, we could use distributions that are mixture models. In that case the real numbers are the parameters of the mixture model; our algorithm would minimize the value of the Lagrangian over the values of the parameters of the mixture model.

An alternative set of approaches still use product distributions, with all of its advantages, but employs a special type of semicoordinate system for Z. For pedagogical simplicity, say that Z is the reals between 0 and 1. So ξ must be a semicoordinate system for the reals, i.e., each $x \in \xi$ must map to a single $z \in \zeta$. Now we want to have those of the q_i that we're modifying be probability distributions, not probability density functions (pdf's), so that our computational algorithm can work with them. Accordingly, in our minimization of the Lagrangian we do not directly modify coordinates that can take on an uncountable number of values (generically indicated with superscript 2), but only coordinates that take on a finite number of values (generically indicated with superscript 1).

We illustrate this for the minimization schemes considered in the preceding sections. For generality, we consider the case where Monte Carlo sampling must be used to estimate the values of $E(G \mid x^1)$ arising in those schemes. Accordingly, we need two things. The first is a way to sample q to get an z, which then provides a G value. The second is a way to estimate the quantities $E(G \mid x^1)$ based upon such empirical data. Given those, all the usual algorithms for searching q^1 to minimize the Lagrangian hold; intuitively, we treat the q^2 like stochastic processes that reside in Z but not X, and therefore not directly controllable by us.

4.1 Reimann semicoordinates

In the **Reimann** semicoordinate system, x^1 can take values $0, 1, ..., B-1$, and x^2 is the reals between 0 and 1. Then with $\alpha_i \triangleq i/B$, we have

$$\begin{aligned} z &= \alpha_{x^1} + x^2/B \\ &= \alpha_{x^1} + x^2(\alpha_{x^1+1} - \alpha_{x^1}). \end{aligned} \tag{12}$$

We then fix $q^2(x^2)$ to be uniform. So all our minimization scheme can modify are the B values of $q^1(x^1)$.

To sample q, we simply sample q^1 to get a value of x^1 and q^2 to get a value of x^2. Plugging those two values into Eq. 13 gives us a value of z. We then evaluate the associated value of the world utility; this provides a single step in our Monte Carlo sampling process.

Next we need a way to use a set of such Monte Carlo sample points to estimate $E(G \mid x^1)$ for all x^1. We could do this with simple histogram averaging, using Laplace's law of succession to deal with bins (x^1 values) that aren't in the data. Typically though with continuous Z we expect $F(z)$ to be smooth. In such cases, it makes sense to allow data from more than one bin to be combined to estimate $E(G \mid x^1)$ for each x^1, by using a regression scheme.

For example, we could use the weighted average regression

$$\hat{F}(z) = \frac{\sum_i F_i e^{-(z-z_i)^2/2\sigma^2}}{\sum_i e^{-(z-z_i)^2/2\sigma^2}}, \tag{13}$$

where σ is a free parameter, z_i is the i'th value of z out of our Monte Carlo samples, and F_i is the associated i'th value of F. Given such a fit, we would then estimate

$$\begin{aligned} E(G \mid x^1) &= \int dx^2 q^2(x^2) F(\zeta(x^1, x^2)) \\ &\approx \int dx^2 q^2(x^2) \hat{F}(\zeta(x^1, x^2)). \end{aligned} \tag{14}$$

This integral can then be evaluated numerically.

Typically in practice one would use a trapezoidal semicoordinate system, rather than the rectangular illustrated here. Doing that introduces linear terms in the integrals, but those can still be evaluated as before.

4.2 Lebesgue semicoordinates

The **Lebesgue** semicoordinate system generalizes the Reimann system, by parameterizing it. It does this by defining a set of increasing values $\{\alpha_0, \alpha_2, ..., \alpha_B\}$ that all lie between 0 and 1 such that $\alpha_0 = 0$ and $\alpha_B = 1$. We then write

$$z = \alpha_{x^1} + x^2(\alpha_{x^1+1} - \alpha_{x^1}). \tag{15}$$

Sampling with this scheme is done in the obvious way. The expected value of G if q^2 is uniform (i.e., all x^2 are equally probable) is

$$\begin{aligned} E(G) &= \sum_{x^1} q_1(x^1) \int dx^2 q^2(x^2) F[\alpha_{x^1} + x^2(\alpha_{x^1+1} - \alpha_{x^1})] \\ &= \sum_{x^1} q_1(x^1) \int_{\alpha_{x^1}}^{\alpha_{x^1+1}} dz \, \frac{F(z)}{\alpha_{x^1+1} - \alpha_{x^1}} \end{aligned} \tag{16}$$

and similarly for $E(G \mid x^1)$. When the α_i are evenly spaced, the Lebesgue system just reduces to the Reimann system, of course.

Note that for a given value of x^1, we have probability mass 1 in the bin following α_{x^1}. So $q^1(x^1)$ sets the cumulative probability mass in that bin. Changing the parameters α_i will change what portion of the real line we assign to that mass — but it won't change the mass.

This may directly affect the Lagragian we use, depending on whether it's the X-space Lagrangian or the Z-space one. In the Reimann semicoordinate system, $S_X \propto S_Z$, and both Lagrangians are the same (just with a rescaled Lagrange parameter). However in the Lebesgue system, if the α_i are not evenly spaced, those two entropies are not proportional to one another. Accordingly, in that scenario, one has to make a reasoned decision of which maxent Lagrangian to use.

The $\{\alpha_i\}$ are a finite set of real numbers, just like q^1. Accordingly, we can incorporate them along with q^1 into the argument of the maxent Lagrangian, and search for the Lagrangian-minimizing set $\{\alpha_i\}$ and q^1 together [15]. In fact, one can

[15] Compare this to the scheme discussed previously for searching directly over semicoordinate transformations, where here the search is over probability distributions defined on the set of possible semicoordinate transformations.

even have q^1 fixed, along with q^2, and only vary the $\{\alpha_i\}$. The difference between such a search over the $\{\alpha_i\}$ when q^1 is fixed, and a search over q^1 when the $\{\alpha_i\}$ are fixed, is directly analogous to the difference between Reimann and Lebesgue integration, in how the underlying distribution $P(z)$ is being represented.

Whether or not q^1 is also varied, one must be careful in how one does the search for each α_i. Unlike for each $\{q_i\}$, each α_i does not arise as a multilinear product, and therefore appears more than once in the Lagrangian. For example, any particular α_{x^1} term arises in Eq. 16 twice as a limit of an integral, and twice in an integrand. All four instances must be accounted for in differentiating the $E(G)$ term in the Lagrangian with respect to that α_{x^1} term.

4.3 Decimal Reimann semicoordinates

In the standard Reimann semicoordinate system, we use only one agent to decide which bin x^1 falls into. To have good precision in making that decision, there must be many such bins. This often means that there are few Monte Carlo samples in most bins. This is why we need to employ a regression scheme (with its attendant implicit smoothness assumptions) to estimate $E(G \mid x_i)$.

An alternative is to break x^1 into a set of many agents, through a hierarchical decimal-type representation. For example, say x^1 can take on 2^K values. Then under a binary representation, we would specify the bin by

$$x^1 = \sum_{i=1}^{K} x_i^1 2^{-i} \tag{17}$$

where x_i^1 is the bit specifying agent i's value. With this change updating the Lagrangian is done by K agents, with each agent i estimating $E(G \mid x_i^1)$ for two values of x_i^1, rather than by a single agent estimating $E(G \mid x^1)$ for all 2^K values of x^1.

With this system, each agent performs its estimations by looking at those Monte Carlo samples where z fell within one particular subregion covering half of [0.0, 1.0]. So long as the samples weren't generated from too peaked a distribution (e.g., early in the search process), there will typically be many such samples, no matter what bit i and associated bit value x_i^1 we are considering. Accordingly, we do not need to perform a regression to estimate $E(G \mid x_i^1)$ to run our Lagrangian minimization algorithms [16]. When q is peaked, some of bin counts from the Monte Carlo data may be small. We can use regression as above, if desired, for such impoverished bins. Alternatively, we can employ a Lebesgue-type scheme to update the bin borders, to ensure that all x_i^1 occur often in the Monte Carlo data.

5 PD theory for Reinforcement Learning

In this section we show how to use semicoordinate transformations and PD theory for a single RL algorithm playing against nature in a time-extended game with delayed payoffs. The underlying idea is to "fracture" the single learner across multiple timesteps into a set of separate agents, one for each timestep. This gives us a

[16] As usual, we could have the entropy term in the Lagrangian be based on either X space or Z space.

distributed system. Constraints are then used to couple those agents. It is straightforward to extend this approach to the case of a multi-agent system playing against nature in a time-extended game.

5.1 Episodic RL

First consider episodic RL, in which reward comes only at the end of an **episode** of T timesteps. The learner chooses actions in response to observations of the state of the system via a **policy**. It does this across a set of several episodes, modifying its policy as it goes to try to maximize reward. The goal is to have it find the optimal such policy as quickly as possible.

To make this concrete, use superscripts to indicate timestep in an episode. So $z = (z^1, z^2, z^3, \ldots z^T) = \zeta(x)$. If we assume the dynamics is Markovian, $P(z) = P(z^1)P(z^2 \mid z^1)P(z^3 \mid z^2)\ldots P(z^T \mid z^{T-1})$. Typically the objective function G depends solely on z^T. For the conventional RL scenario, each z^t can be expressed as (s^t, a^t), where s^t is the state of the system at t, and a^t is the action taken then.

As an example, say the learner only takes into account the previous state when deciding its current action, and that it observes the state of the system (at the previous instant) with zero error. As a matter of notation, we must decide where in the sequence $\ldots \rightarrow$ *state* $\rightarrow$ *action* $\rightarrow$ *state*$\ldots$ to put the dividing lines between successive moments in time. Choose to put the dividing lines between a state and the succeeding action. So the state at t is conditionally independent of the action at $t-1$, given the state at $t-1$ and the action at t. Accordingly we can write

$$P(z^t \mid z^{t-1}) = P(s^t, a^t \mid s^{t-1}, a^{t-1}) = P(a^t \mid s^{t-1})P(s^t \mid s^{t-1}, a^t). \tag{18}$$

$P(z)$ is not a product distribution over the components of z. So in the usual way we must use a semicoordinate transformation $\zeta(.)$ to represent each of the conditional distributions (see Thm. 1). This will allow us to use a product distribution to represent the system. The result is an X far larger than Z, and we write $P(z)$ with abbreviated notation as

$$\begin{aligned} P(s^1, a^1, \ldots, s^T, a^T) &= P(a^1)P(s^1)\prod_{t>1} P(a^t \mid s^{t-1})P(s^t \mid s^{t-1}, a^t) \\ &= q_{A^1}(a^1)q_{S^1}(s^1)\prod_{t>1} q_{t,s^{t-1}}(a^t)q_{t,s^{t-1},a^t}(s^t). \end{aligned} \tag{19}$$

where the subscripts A^1 and S^1 indicate the components of q associated with the time-1 action and state, respectively. (Note we implicitly assume that the action at time 1 is decided independently of the state then.) In RL we typically can only control the $q_{t,s^{t-1}}$ distributions. While the other q_i go into the Lagrangian, they are fixed by nature and not directly varied by us.

Recall that the usual reason we restrict ourselves to (low-dimensional) product distributions in PC is that their updating is tractable, whereas updating of (high-dimensional) fully coupled distributions would not be. In contrast, here the restriction to product distributions is not a restriction that is imposed to make updating tractable. Rather it is simply a reflection of the Markovian character of the problem.

It may be desired that the policy of the learner be constant across each episode (i.e., stationary). To impose that we can add penalty terms $\lambda_i[q_{t,s}(a) -$

$q_{t+1,s}(a)]\ \forall t, s, a$ to the X-space Lagrangian to enforce time-translation invariance in the usual way. [17],[18]. Time-translation invariance of the $P(s^t \mid s^{t-1}, a^{t-1})$ does not explicitly need to be addressed. Indeed, it need not even hold.

As usual, the $E(G)$ term in our Lagrangian has the same value whether we evaluate it as

$$\int dx\ q(x)G(x) \triangleq \int dx\ q(x)G(\zeta(x)) \tag{20}$$

or as

$$\int dz\ P(z)G(z) \triangleq \int dz\ [\int dx\ q(x)\delta(\zeta(x) - z)]G(z). \tag{21}$$

Also as usual, in contrast to the $E(G)$ term the entropy term in the Lagrangian can have a different value, depending on whether we consider the X-space entropy, $\int dx\ q(x)\ln[q(x)]$ or the Z-space entropy, $\int dz\ P(z)\ln[P(z)]$. For simplicity, here we will consider the X-space Lagrangian.

Up to an overall additive constant, the X-space Lagrangian is

$$\begin{aligned}\mathscr{L}(\{q_{t,s^{t-1}}\}) = \beta \sum_{\mathbf{a},\mathbf{s}} q_{A^1}(a^1) q_{S^1}(s^1) \prod_{t>1} q_{t,s^{t-1}}(a^t) q_{t,s^{t-1},a^t}(s^t) G(\mathbf{s}) \\ - S(q_{A^1}) - \sum_{t>1,s^t} S(q_{t,s^t}) + \sum_{t>1,s,a} \lambda_{t,s,a}[q_{t,s}(a) - q_{t-1,s}(a)]\end{aligned} \tag{22}$$

where $\mathbf{s}$ and $\mathbf{a}$ indicate the trajectories of all s^t and all a^t, respectively, and the entropy function $S(.)$ should not be confused with the subscript s^1 on q (which indicates the component of q referring to the time-1 value of the state variable).

We can then use any of the standard techniques for finding the critical point of this Lagrangian. So for example say we use Nearest Newton. Then at the end of each episode, for each $t > 1, s, a$, we should increase $q_{t,s}(a)$ by

$$\alpha \left[q_{t,s}(a) \Big(\beta E(G \mid s^{t-1} = s, a^t = a) + \ln(q_{t,s}(a)) + \lambda_{t,s,a} - \lambda_{t+1,s,a} \Big) - \text{const} \right], \tag{23}$$

where as usual α is the step size and const is the normalization constant (see Eq. 4). Monte Carlo can be used to estimate the expectation values. Note that in addition to q, the Lagrange parameters must be periodically updated, for example in a first order ascent method [5].

Similarly, a parallel Brouwer update would set

$$q_{t,s}(a) \propto e^{-\beta E(G|s^{t-1}=s,a^t=a) + \lambda_{t,s,a} - \lambda_{t+1,s,a}} \tag{24}$$

If we use Monte Carlo with geometric data-aging to estimate the expectation value, then this update rule can be viewed as a version of Q-learning [26] modified by inclusion of the Lagrange parameters, by indexing of (state, action) pairs by time, and by the use of probability distributions rather than Q-functions as the real-valued function of state-action pairs. Note though that here such a use of real-valued

[17] Note that unlike constraints over X, those over $\mathcal{Q}$ are not generically true only to some high probability, but rather can typically hold with probability 1.

[18] If such constancy is a hard and fast requirement, rather than just desirable, then the simplest aproach is simply to have a single agent with a distribution $q_s(a)$ that sets $q_{t,s}(a)\ \forall t$.

functions of state-action pairs is derived, from the nature of the Markovian process. In contrast, as conventionally considered in RL, the choice of working with a real-valued function of state-action pairs (as opposed to a real-valued function of states, like the value function for example) is essentially arbitrary.

The Lagrange parameters $\lambda_{t,s,a}$ can provide useful information about the sensitivities of performance to the stationarity requirement for different moments in time, and for different state-action pairs. In addition we can also stop their growth short, so that they do not force stationarity exactly. Doing this allows the system to have some non-stationarity, if doing so helps performance.

We can enforce the time-translation invariance without the $\lambda_{t,s,a}$ Lagrange parameters. This is done by rewriting the Lagrangian so that separately for every s and a, the variables $\{q_{t,s}(a) : t = 1, \ldots T\}$ are explicitly set equal to one another. This means using a single variable $q_s(a)$ rather than the time-indexed set $q_{t,s}(a)$.

In this alternative our formula for the Lagrangian and update rules have to be changed to reflect the fact that a single $q_s(a)$ can contribute more than once to a particular trajectory (s, a). This potential multiplicity modifies the Monte Carlo estimation procedure as well. In particular, it means that a single Monte Carlo episode may contribute more than one datum to the estimation of a term $E(G \mid s, a)$.

Consider updating $q_s(a)$ for one particular $s - a$ pair under this alternative approach. To do so first define $(\mathbf{a}, \mathbf{s})$ as the entire history, and

$$q(\mathbf{a}^{t>1}, \mathbf{s}^{t>1}) \triangleq \prod_{t>1} q_{s^{t-1}}(a^t) q_{t,s^{t-1},a^t}(s^t). \tag{25}$$

Note that $q(\mathbf{a}^{t>1}, \mathbf{s}^{t>1})$ does not involve the variable $q_{A^1}(.)$.

Using this shorthand we can write our alternative X-space Lagrangian as

$$\begin{aligned}\mathscr{L}(\{q_s\}) = \beta \sum_{\mathbf{a},\mathbf{s}} [q_{A^1}(a^1) q_{S^1}(s^1) \; \times \\ \sum_{n \geq 0} I((\mathbf{a}^{t>1}, \mathbf{s}^{t>1}) \text{ goes through } (s, a) \text{ exactly } n \text{ times}) \; q(\mathbf{a}^{t>1}, \mathbf{s}^{t>1}) G(\mathbf{s})] \\ - S(q_{A^1}) \; - \; T \sum_s S(q_s)\end{aligned} \tag{26}$$

where the indicator function $I(.)$ equals 1 if its argument is true, 1 otherwise. Differentiating gives

$$\begin{aligned}\frac{\partial \mathscr{L}(\{q_s\})}{\partial q_s(a)} = \beta \sum_{\mathbf{a},\mathbf{s}} [q_{A^1}(a^1) q_{S^1}(s^1) \; \times \\ \frac{\sum_{n \geq 0} I((\mathbf{a}^{t>1}, \mathbf{s}^{t>1}) \text{ goes through } (s, a) \text{ exactly } n \text{ times}) \; n q(\mathbf{a}^{t>1}, \mathbf{s}^{t>1}) G(\mathbf{s})}{q_s(a)}] \\ + T \; + \; T \ln(q_s(a)).\end{aligned} \tag{27}$$

A similar expression gives the partial derivative with respect to q_{A^1}.

The randomly (according to q) generated Monte Carlo sample trajectories can be used to estimate this. Note that here we have expressed the partial derivative without conditional expectations, in contrast to how we express it for conventional Monte Carlo estimation for Probability Collectives. (The reason for doing it this alternative way is that it simplifies how to relate the $n > 1$ terms in the sum to empirical quantities arising in the Monte Carlo.) Accordingly, to evaluate the Monte Carlo estimates we divide by the total number of Monte Carlo samples, not by the total number of such samples that went through some particular state-action pair.

In practice, to deal with state-action pairs that are never visited in any of the Monte Carlo samples, one should use shrink-wrapping (rather than, for example, forcing samples).

5.2 Discounted sum RL

It is worth giving a brief overview of how the foregoing gets modified when we instead have a single "episode" of infinite time, with rewards received at every t, and the goal of the learner at any instant being to optimize the discounted sum of future rewards.

Let the matrix $\mathbf{P}$ be the conditional distribution of state z_t given state z_{t-1}, and γ a real-valued discounting factor between 0 and 1. Write the single-instant reward function as a vector $\mathbf{R}$ whose components give the value for the various z_t. Then if $\mathbf{P_0}$ is the current distribution of (single-instant) states, z_0, the world utility is

$$([\sum_{t=1}^{\infty}(\gamma\mathbf{P})^t]\mathbf{P_0})\cdot\mathbf{R}$$

The sum is just a geometric series, and equals $\frac{\gamma\mathbf{P}}{\mathbf{1}-\gamma\mathbf{P}}$, where $\mathbf{1}$ is the identity matrix, and it doesn't matter if the matrix inversion giving the denominator term is right-multiplied or left-multiplied by the numerator term.

We're interested in the partial derivative of this with respect to one of the entries of $\mathbf{P}$ (those entries are given by the various $q_{i,j}$). What we know though (from our historical data) is a (non-IID) set of samples of $(\gamma\mathbf{P})^t\mathbf{P}\cdot\mathbf{R}$ for various values of t and various (delta-function) $\mathbf{P}$. So it is not as trivial to use historical data to estimate the gradient of the Lagrangian as in the canonical optimization case. More elaborate techniques from machine learning and statistics need to be brought to bear.

Acknowledgements

We would like to thank Stephane Airiau, Chiu Fan Lee, Chris Henze, George Judge, Ilan Kroo and Bill Macready for helpful discussion.

References

1. Airiau, S., and D. H. Wolpert, "Product distribution theory and semi-coordinate transformations", Submitted to AAMAS 04.
2. Antoine, N., S. Bieniawski, I. Kroo, and D. H. Wolpert, "Fleet assignment using collective intelligence", *Proceedings of 42nd Aerospace Sciences Meeting*, (2004), AIAA-2004-0622.
3. Aumann, R.J., and S. Hart, *Handbook of Game Theory with Economic Applications*, North-Holland Press (1992).
4. Basar, T., and G.J. Olsder, *Dynamic Noncooperative Game Theory*, Siam Philadelphia, PA (1999), Second Edition.

5. Bertsekas, D.P., *Constrained Optimization and Lagrange Multiplier Methods*, Athena Scientific, Belmont, MA (1996).
6. Bieniawski, S., and D. H. Wolpert, "Adaptive, distributed control of constrained multi-agent systems", *Proceedings of AAMAS 04*, (2004).
7. Bieniawski, S., and D. H. Wolpert, "Using product distributions for distributed optimization", *Proceedings of ICCS 04*, (2004).
8. Bieniawski, S., D. H. Wolpert, and I. Kroo, "Discrete, continuous, and constrained optimization using collectives", *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York*, (2004), in press.
9. Catoni, O., "Solving scheduling problems by simulated annealing", *SIAM Journal on Control and Optimization* **36**, 5 (1998), 1539–1575.
10. Cover, T., and J. Thomas, *Elements of Information Theory*, Wiley-Interscience New York (1991).
11. Crites, R. H., and A. G. Barto, "Improving elevator performance using reinforcement learning", *Advances in Neural Information Processing Systems - 8* (D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo eds.), MIT Press (1996), 1017–1023.
12. Diekmann, R., R. Luling, and J. Simon, "Problem independent distributed simulated annealing and its applications", *Applied Simulated Annealing.* Springer (1993), pp. 17–44.
13. Fudenberg, D., and D. K. Levine, *The Theory of Learning in Games*, MIT Press Cambridge, MA (1998).
14. Fudenberg, D., and J. Tirole, *Game Theory*, MIT Press Cambridge, MA (1991).
15. Hu, J., and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm", *Proceedings of the Fifteenth International Conference on Machine Learning*, (June 1998), 242–250.
16. Jaynes, E. T., and G. Larry Bretthorst, *Probability Theory : The Logic of Science*, Cambridge University Press (2003).
17. Kaelbing, L. P., M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey", *Journal of Artificial Intelligence Research* **4** (1996), 237–285.
18. Kirkpatrick, S., C. D. Jr Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", *Science* **220** (May 1983), 671–680.
19. Laughlin, D.L., M. Morari, and R.D. Braatz, "Robust performance of cross-directional control systems for web processes", *Automatica* **29** (1993), 1394–1410.
20. Lee, C. Fan, and D. H. Wolpert, "Product distribution theory for control of multi-agent systems", *Proceedings of AAMAS 04*, (2004).
21. Mackay, D., *Information theory, inference, and learning algorithms*, Cambridge University Press (2003).
22. Macready, W., S. Bieniawski, and D.H. Wolpert, "Adaptive multi-agent systems for constrained optimization", Technical report IC-04-123 (2004).
23. Macready, W. G., and D. H. Wolpert, "Distributed constrained optimization", preprint cond-mat/XXX (June 2004).
24. Mesbai, M., and F.Y. Hadaegh, "Graphs, matrix inequalities, and switching for the formation flying control of multiple spacecraft", *Proceedings of the American Control Conference, San Diego, CA*, (1999), 4148–4152.
25. Osborne, M., and A. Rubenstein, *A Course in Game Theory*, MIT Press Cambridge, MA (1994).

26. Sutton, R. S., and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press Cambridge, MA (1998).
27. Vidal, R. V. V. ed., *Applied Simulated Annealing (Lecture Notes in Economics and Mathematical Systems)*, Springer (1993).
28. Wolfe, J., D.F. Chichka, and J.L. Speyer, "Decentralized controllers for unmanned aerial vehicle formation flight", *American Institute of Aeronautics and Astronautics* **96** (1996), 3933.
29. Wolpert, D. H., "Factoring a canonical ensemble", preprint cond-mat/0307630.
30. Wolpert, D. H., "Bounded rational games, information theory, and statistical physics", *Complex Engineering Systems* (D. Braha and Y. Bar-Yam eds.), (2004).
31. Wolpert, D. H., "What information theory says about best response, binding contracts, and collective intelligence", *Proceedings of WEHIA04* (A. N. et al ed.), Springer Verlag (2004).
32. Wolpert, D. H., and S. Bieniawski, "Distributed control by lagrangian steepest descent", *Proceedings of CDC 04*, (2004).
33. Wolpert, D. H., and C. F. Lee, "Adaptive metropolis hastings sampling using product distributions", *Proceedings of ICCS 04*, (2004).