

Sequentially updated Probability Collectives

Michalis Smyrnakis and David S. Leslie

Abstract—Multi-agent coordination problems can be cast as distributed optimization tasks. Probability Collectives (PCs) are techniques that deal with such problems in discrete and continuous spaces [15]. In this paper we are going to propose a new variation of PCs, Sequentially updated Probability Collectives. Our objective is to show how standard techniques from the statistics literature, Sequential Monte Carlo methods and kernel regression, can be used as building blocks within PCs instead of the ad hoc approaches taken previously to produce samples and estimate values in continuous action spaces.

We test our algorithm in three different simulation scenarios with continuous action spaces. Two classical distributed optimization functions, the three and six dimensional Hartman functions [6] and a vehicle target assignment type game [1]. The results for the Hartman functions were close to the global optimum, and the agents managed to coordinate to the optimal solution of the target assignment game.

I. INTRODUCTION

Multi-agent systems solve applications in which agents should act independently to achieve a common goal. This kind of problem can be solved using distributed optimization algorithms. In this paper we will consider cases where the agents should coordinate to accomplish a task in continuous action space without discretising it.

Probability collectives [15] are optimization techniques that can be used for such problems. The goal of these algorithms is to find a product distribution that optimizes the expected utility of the multi-agent system. This product distribution consists of a probability distribution over each agent's action space. The PC framework iteratively updates these distributions, with each agent 'responding' to the distributions of the others.

The solution that we obtain from PC algorithms requires integration over the distributions of the other agents. In continuous actions, with general distributions over these spaces, it is impossible to even write down the distribution explicitly, and therefore impossible to share the distributions or perform integrations. A solution to this problem is to approximate this integral using sampling methods. Additionally when the action spaces are continuous we have to extrapolate the results of the sampling methods to cover the whole range of the actions.

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralized Data and Information Systems) project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership (EP/C548051/1).

M. Smyrnakis is with Department of Mathematics, University of Bristol, University Walk, Bristol, BS8 1TW, U.K. m.smyrnakis@bris.ac.uk

D. Leslie is with the Department of Mathematics, University of Bristol, University Walk, Bristol, BS8 1TW, U.K. david.leslie@bris.ac.uk

Our aim is to introduce standard statistical techniques into the main framework of PCs, replacing the ad-hoc techniques that have been used till now for these two building blocks of PC algorithms. In particular we will use Sequential Monte Carlo [8] to produce samples and kernel regression [9], [12] to perform the extrapolations.

The remainder of this report is organized as follows. The next section contains a description of probability collectives. Sections III and IV present a brief description of the statistical methods we are going to incorporate in the PCs framework. Section V describes the proposed variation of probability collectives. The results of the Hartman's functions simulations are in Section VI and those of the vehicle target assignment game are presented in Section VII. Finally, in the last section we are present some conclusions and future work.

II. PROBABILITY COLLECTIVES

Probability collectives are a suite of decentralized algorithms which are used to solve distributed optimization problems. Most optimization methods search for an $x \in \mathbb{R}^N$ that optimizes a reward function $G(x)$. In contrast PCs search for a product distribution $q(x) = q_1(x_1) \times \dots \times q_i(x_i) \times \dots \times q_N(x_N)$ that optimizes the expected value of $G(x)$. More precisely, for a minimization problem, PC algorithms search for a distribution $q \in \mathcal{Q}$, where \mathcal{Q} is the set of all the product distributions, that minimizes $E_q(G(x))$ [15].

In [15] entropic barrier functions, S , and Lagrange multipliers, λ_i , were introduced to encode the constraint that each q_i must be a probability distribution that puts mass in all areas of action space. This results in the new objective function:

$$L(q, T) = E_q(G) - T \sum_i S(q_i) + \sum_i \lambda_i \left(\int q_i(x_i) dx_i - 1 \right) \quad (1)$$

where $S(q_i) = - \int_{x_i} q_i(x_i) \log q_i(x_i) dx_i$ is the Shannon entropy and T is a temperature parameter. Following Wolpert we will call expression (1) the Maxent Lagrangian.

PCs search for the critical points of the Maxent Lagrangian. We can find these critical points if we set the derivative of expression (1) equal to zero. Brouwer's fixed point theorem ensures that solutions exist. The form of these solutions up to a multiplicative constant is the following:

$$q_i(x_i) \propto e^{-E_{q_{-i}}(G|x_i)/T} \quad (2)$$

where $E_{q_{-i}}(G|x_i)$ is the expected value of the reward function under the probability distribution $q_1 \times \dots \times q_{i-1} \times q_{i+1} \times \dots \times$

q_N conditional on the value x_i . More formally

$$E_{q_{-i}}(G|x_i) = \int G(x)q_{-i}(x_{-i})dx_{-i} \quad (3)$$

where $x = (x_i, x_{-i})$. We can express the solution as

$$b_i(q_{-i})(x_i) = k \cdot \exp(-E_{q_{-i}}(G|x_i)/T) \quad (4)$$

where k is a constant.

Because there is no easy analytical solution of equation (2) we can search for solutions by iteratively setting:

$$q_i^{t+1} = b_i(q_{-i}^t) \quad \forall \quad i = 1, \dots, N \quad (5)$$

Following Wolpert we will call this the parallel Brouwer updating method. This is because of its link to the Brouwer fixed point theorem.

The simultaneous update of all the distributions q_i has as an effect that it is possible to observe thrashing [15]. This describes the situation where each agent updates his distribution according to the previous values of the other agents' distributions, but since all the agents are changing their distributions there is no guarantee that the new product distribution q^{t+1} will not increase the Maxent Lagrangian. It is worth noting that a similar phenomenon is well known in game theory and indeed it is possible to consider PCs as a type of learning in games [15], [13], [11].

Several techniques have been proposed to avoid thrashing. These include serial updates where only some of the q_i^t s are updated between time t and $t+1$, and moving q_i^t part way towards $b_i(q_{-i}^{t-1})$. This article does not address techniques to avoid thrashing, and throughout we set

$$q_i^{t+1} = (1 - \alpha)q_i^t + \alpha b_i(q_{-i}^t). \quad (6)$$

In this report we are concerned with decentralized optimization when the action spaces are continuous, and exclude techniques that discretise to reduce the problem to the discrete case. In this context it is impossible to write and perform calculations directly with a density function q_i . This is because it is not feasible to compute q_i^t pointwise and, as a result of this, the agents cannot share their distributions to evaluate equation (4). Hence sampling techniques must be used. Two have currently been suggested in the PCs literature.

The first one is called delayed sampling PCs [14]. Under this scheme at time t agents sample from their distributions q_i^t using a full Markov Chain Monte Carlo (MCMC) sampler. Then the agents use this sample for the next L iterations of the algorithm to evaluate expectation (3). Then at time $t+L+1$ we sample again using MCMC with target distribution q_i^{t+L+1} . We continue this procedure until the termination of our algorithm. Since the samples $x_{i,1:N}^t$ are distributed according to q_i^t , but are used to evaluate $E_{q_{-i}^{t+L}}(G|x_i)$, for $l = 0, \dots, L$, "data-aging" techniques must be used to ensure validity of the estimate. One such approach is to apply rejection sampling to $x_{i,1:N}^t$, resulting in a sample $x_{i,1:M}^{t+l}$ from q_i^{t+l} , where $M \leq N$

The other variation of probability collectives is called immediate sampling [14]. Instead of using the same sample from distribution q_i^t for l time steps, we sample from the current distribution on each iteration using importance sampling. We produce a sample $x_{i,1:N}$ from a distribution $g(x)$ which is easy to sample from and is similar to q_i^t . Then we calculate the weights of the sample using the expression $w_{i,n}^t = \frac{q_i^t(x_{i,n})}{g_i(x_{i,n})}$. This sample can be used to evaluate the desired expectation, using the identity

$$\begin{aligned} & \int G(x_i, x_{-i}) [\prod_{j \neq i} q_j(x_j)] dx_{-i} \\ &= \int G(x_i, x_{-i}) [\prod_{j \neq i} \frac{q_j(x_j)}{g_j(x_j)} g_j(x_j)] dx_{-i} \\ &= E_{g_{-i}} [G(x) \prod_{j \neq i} \frac{q_j(x_j)}{g_j(x_j)}] \simeq \sum_{n=1}^N G(x_n) w_{-i,n} \end{aligned} \quad (7)$$

where $w_{-i,n} = \prod_{j \neq i} w_{j,m}$

Choosing g is difficult. For that reason we are going to introduce Sequential Monte Carlo (SMC) methods, which generalize importance sampling to the case of a sequence of distributions.

An additional problem in continuous action spaces is the estimation of $E(G|x_i)$ for values of x_i that are not present in the sample of q_i . The solution that has been proposed [15] is to use regression. When we are using regression we are able to extrapolate the estimation of $G(x)$ in the whole range of x . In particular in [2] a simple regression based on exponentially weighted averaging across samples is used. In this article we introduce the standard and well understood approach of kernel regression [9], [12] to the PCs technique.

III. SEQUENTIAL MONTE CARLO SAMPLERS

Consider a general case where there is a sequence of distributions π^{t-1} , where $t = 1, \dots, T$, and we want to evaluate $E_{\pi^t} f(x^t)$. When we cannot evaluate this integral analytically a common approach is to make an approximation based on sampling methods. Sequential Monte Carlo samplers [8] are used for this approximation in a sequential context.

SMC samplers can be viewed as a generalization of sequential importance sampling that also have some of the iterative approximation benefits of MCMC algorithms. At time t there is a weighted sample, $(x_{1:N}^{t-1}, w_{1:N}^{t-1})$, named "particles". The SMC algorithm uses these particles to generate a sample from π^t through re-weighting, re-sampling if it is necessary and moving. Under the assumption that distribution π^t is not very different from π^{t-1} , Figure 1 can describe an approximately optimal [8] SMC algorithm.

At time $t = 0$ we have an equally weighted set of particles $(x_{1:n}^0, w_{1:n}^0)$ from a distribution π^0 . Then for each iteration t , $t = 1, \dots, T$, we create a weighted sample from distribution π^t . If we follow the method of Section 3.3.2.3 of [8] then we can use the following formula to update the weights:

$$w_n^t = w_n^{t-1} \frac{\pi_n^t(x_n^{t-1})}{\pi_n^{t-1}(x_n^{t-1})}. \quad (8)$$

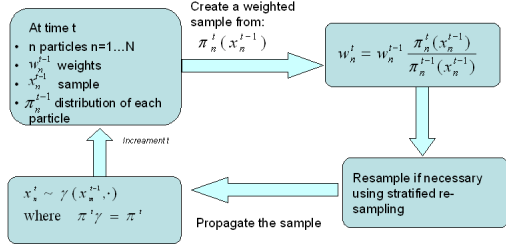


Fig. 1. Sequential Monte Carlo procedure

The current set of particles $\{x_{1:N}^{t-1}, w_{1:N}^t\}$ are now a weighted sample from distribution π^t .

The SMC approaches suffer from a degeneracy problem. After a number of iterations most particles will have small weights, so only few of them will influence the final result. For that reason a re-sampling procedure has to be introduced. In this report when the effective sample size $N_{effective} = \frac{1}{\sum (w_n^t)^2}$ is smaller than a threshold number N_{thres} , then we use stratified re-sampling [7] to avoid the degeneracy of the weights.

In the case we have to use re-sampling then our sample will consist of copies of particles with high weights from our initial sample. This narrows the number of unique samples and thus there is a need for greater divergence in the sample. For that reason we propagate the sample $x_{1:N}^{t-1}$ using a single move of an MCMC sampler with stationary distribution π^t . In this article we are going to use a random walk Metropolis-Hastings algorithm to propagate the samples. In this case we will propose a sample x_n^t from a symmetric distribution about x_n^{t-1} , then accept it with probability $c_n^t = \min\{1, \frac{\pi_n^t(x_n^t)}{\pi_n^t(x_n^{t-1})}\}$.

IV. KERNEL REGRESSION

Suppose we have a sample of pairs $(x_{1:N}, y_{1:N})$ and we wish to form an estimate of the function $f(x) = E(y|x)$. A standard non-parametric technique, based on kernel density estimation, is the Nadaraya-Watson estimator [9], [12]

$$\hat{f}(x) = \frac{\sum_{n=1}^N K_h(x - X_n) Y_n}{\sum_{n=1}^N K_h(x - X_n)} \quad (9)$$

where h is a bandwidth parameter of the regression and K_h is a kernel function. In this report we take:

$$K_h(u) = (2\pi h)^{-1/2} \exp\{-u^2/2h\}. \quad (10)$$

and automatically calculate the bandwidth h using

$$h_x = \frac{M(|x(k) - M(x(k))|)}{(\frac{4}{3}0.6745N)^{0.2}}. \quad (11)$$

where $M(x) = \text{median}(x)$ [4].

This is an estimation for the kernel regression when the sample is equally weighted. In the case that the sample is weighted with w_i denoting the weight of sample X_i , the formula slightly changes and has the following form [5]:

$$\hat{f}(x) = \frac{\sum_{n=1}^N K_h(x - X_n) w_n Y_n}{\sum_{n=1}^N K_h(x - X_n) w_n} \quad (12)$$

We used Nadaraya-Watson regression, instead of more recent approaches like local polynomial regression, because it is easy to implement, it has been widely studied and also it is proved to be consistent [9], [12], [5].

V. SEQUENTIALLY UPDATED PROBABILITY COLLECTIVES

In this section we describe how we can incorporate the SMC algorithm and kernel regression, as described in previous sections, into probability collectives. Our proposed algorithm is described in Figure 2

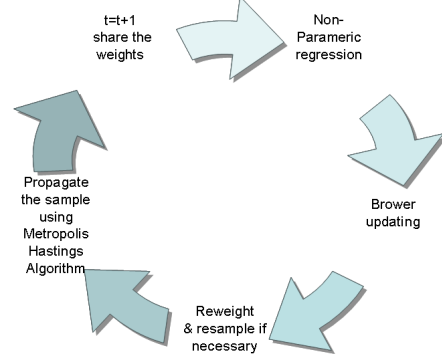


Fig. 2. PC algorithm with SMC

At time t each agent has a set of particles $(w_{i-1,1:N}^{t-1}, x_{i,1:N}^{t-1})$ and corresponding observations $G_{1:N}$ of the utility function. When the action space is continuous we need to estimate the value of $E_{q_{t-1}^i}(G(x)|x_i)$. Since the expectation is over x_{-i} , conditional on x_i , the correct weights to use in our kernel regression are w_{-i}^{t-1} . See equation (7). Thus formula (12) can be used to evaluate this expectation which results in the following expression:

$$E_{q_{t-1}^i}(G(x)|x_i) = \frac{\sum_{n=1}^N K_h(x_i - x_{i,n}^{t-1}) w_{-i,n}^{t-1} G(x_{i,n}^{t-1})}{\sum_{n=1}^N K_h(x_i - x_{i,n}^{t-1}) w_{-i,n}^{t-1}}. \quad (13)$$

Now it is possible to evaluate $b_i(q_{t-1}^i)(x)$ for any x and use the Brouwer updating method to give $q_t^i(x)$. To avoid thrashing we will use formula (6) for this update.

Next we create a weighted sample as described in Section III, using equation (8) to update the weights. The new weights of agent i will be :

$$w_{i,n}^t = w_{i,n}^{t-1} \frac{q_{i,n}^t(x_{i,n}^{t-1})}{q_{i,n}^{t-1}(x_{i,n}^{t-1})}. \quad (14)$$

The set of particles that it is available now, $(w_{i-1,1:N}^t, x_{i,1:N}^t)$ is a weighted sample from q_t^i . If it is necessary we re-sample to avoid degeneracy of the weights.

Finally we propagate the sample introducing random walk Metropolis-Hastings moves. We initially propagate the sample $x_{i,1:N}^{t-1}$ adding some random noise $\epsilon_{i,1:N}^t \sim N(0, \sigma^2)$, where $\sigma = 0.01$. The new sample will be $x_{i,1:N}^{new} = x_{i,1:N}^{t-1} + \epsilon_{i,1:N}^t$. We accept sample $x_{i,n}^{new}$, so $x_{i,n}^t = x_{i,n}^{new}$, with probability $\min\{1, \frac{q_{i,n}^t(x_{i,n}^{new})}{q_{i,n}^{t-1}(x_{i,n}^{t-1})}\}$. Otherwise $x_{i,n}^t = x_{i,n}^{t-1}$. Hence to accept or

Hartman3		Hartman6	
Value	Type	Value	Type
-1.0008	local	-1.0116	local
-2.6722	local	-1.5099	local
-3.0796	local	-3.2036	local
-3.7618	local	-3.2028	local
-3.8628	global	-3.3224	global

TABLE I
MINIMA OF HARTMAN'S FUNCTION

reject the sample we need to evaluate $q_i^t(x^{new})$ for arbitrary x_i^{new} . We can use the results of the kernel regression we have until time t to compute $q_i^t(x^{new})$ as follows:

$$q_i^t(x^{new}) = q_i^0(x^{new})(1-\alpha)^t + b_i(q_{-i}^1(x^{new}))(1-\alpha)^{t-1}\alpha + \dots + b_i(q_{-i}^t(x^{new}))\alpha. \quad (15)$$

VI. HARTMAN'S FUNCTIONS

We tested our algorithm using two functions that are used to examine the performance of distributed optimization algorithms, the three and six dimensional Hartman's functions [6]. The form of the Hartman's functions are:

$$H_3 = -\sum_{i=1}^4 a_i \exp\left(-\sum_{j=1}^3 A_{ij}(X_j - P_{ij})^2\right) \quad (16)$$

$$H_6 = -\sum_{i=1}^4 a_i \exp\left(-\sum_{j=1}^6 B_{ij}(X_j - R_{ij})^2\right) \quad (17)$$

for the three and six dimensions respectively, where the rows of P and R represent the locations of the functions' local minima, A and B are proportional to the value of the Hessian evaluated at the points of these minima and a is the depth of the minimum [6]. In addition to the global minimum there are four local minima with values as in Table I.

A. Results

We compared the results of the sequential PC algorithm with the basic PC algorithm that is presented in [10]. In particular we used two variations of this algorithm, a centralized one that uses information also about the correlation between the agents and one that is decentralized assuming that the agents are independent.

We tested the algorithms using two different stopping criteria, the first one was when the relative change in the utility we gain was smaller than 0.01, and the latter one was to stop the algorithm after a fixed number of iterations [3]. We run 100 trials of each problem and in each trial we used the same initial sample for all the algorithms.

The results we present in this section were obtained using temperature parameter $T = 0.1$ and inertia parameter $\alpha = 1$. We set the inertia parameter to this extreme value because we didn't observe thrashing and also the algorithm converged faster to the optimal solution. For the three dimensional case we used 50 particles, and the fixed stopping criterion was

set to 10 iterations of the algorithm. When we used higher number of particles the performance of all the algorithms was very similar, thus we preferred to use a small number of particles, i.e. 50 particles, so we were able to observe some differences between the algorithms. In six dimensions we increased the number of particles to 500 and the iteration number to 25. In both cases we used a small number of iterations because, as well as convergence of the algorithm to the global optimum, we are also interested in the speed of the convergence of the algorithm. This is because in numerous applications there are restrictions on the communications and the most preferable solution is the one where fewest messages are exchanged between the agents. As performance measurement we compute the best observed function value in each trial and the number of trials each algorithm failed to converge to the global optimum. The condition we set to choose if the algorithm had converged to the global optimum was the following: in a trial if the obtained utility was less than the mid-point of the global minimum and its nearest local minimum we assumed that the algorithm had converged to the global minimum, otherwise it had converged to the local one.

Table II illustrates the results for the three different algorithms. For the three dimensional case when we used the stopping criterion the SPC performed better than the other two variations of PC. When we allowed more iterations for each trial then $PC_{decentralized}$ performed better.

In the six dimensional case SPC performed better whether or not we used the stopping criterion. The average utility we obtained after 100 trials combined with the associated standard deviation, as they are in Table II, does not provide enough evidence to decide if the SPC has significantly better results than the other two PC variations, especially the decentralized one. This is because the distance between the global minimum and its nearest local one is small. But if we observe the times that each algorithm became trapped in the local minimum area then we can observe that SPC performs significantly better than the other two algorithms. Figure 3 depicts the number of the trials that SPC and $PC_{decentralize}$ were trapped in an area near to the -3.20 minimum and how many times manage to reach an area near to the global minimum -3.32 . In 48 % of the trials $PC_{decentralise}$ stopped

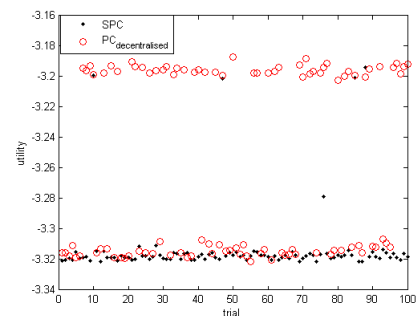


Fig. 3. Convergence of the SPC and $PC_{decentralized}$ algorithms.

in an area near to the local minimum, whereas the SPC algorithm was stuck in this local minimum only 4% of the time.

VII. VEHICLE TARGET ASSIGNMENT TYPE OF GAME

A. Problem definition

We have also tested our algorithm in a vehicle target assignment type game [1]. Sensors have been placed at the corners of a unit square aiming to observe the objects that are inside this area. The sensors can choose a direction $x \in [0, \frac{\pi}{2}]$ to observe. According to this direction there is a probability to observe the objects around them, which depends on the angle, θ , between the direction that the sensor has chosen to observe and the object. This probability depends also on the distance, r , that the object has from the sensor. A description of the simulations set up is depicted in Figure 4.

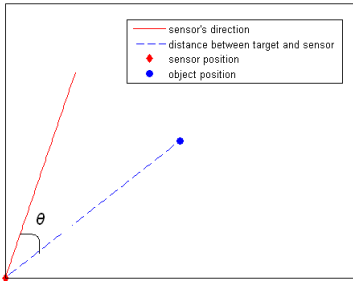


Fig. 4. Simulation scenario description. The solid line represents the direction x of the sensor and the dashed one is the distance r between the sensor and the object

For a sensor i the probability to observe an object j , p_{ij} , is given by the following formula:

$$p_{ij} = \frac{1}{r^2} \frac{\cos(4\theta_{ij}) + 1}{2} \mathbb{I}_{\{|\theta_{ij}| \leq \frac{\pi}{4}\}} \quad (18)$$

This formula denotes that a sensor has higher chance to observe objects that are close to the direction he has chosen. In the case that an object is further than $\frac{\pi}{4}$ radians to the left or to the right of the direction of a sensor, then this sensor cannot observe this object, and so $p_{ij} = 0$.

The probability a sensor i observes an object j is independent from the other sensors. Hence the probability that an object will be observed by any sensor is $1 - \prod_{i=1}^I (1 - p_{ij})$. Assuming the value of observing an object is V_j , we compute the utility it produces as the product between its value and the probability to be observed.

$$G_j = V_j (1 - \prod_{i=1}^I (1 - p_{ij})) \quad (19)$$

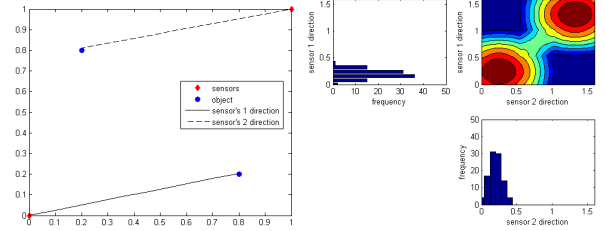
which leads to the following form of global utility:

$$G = \sum_j G_j. \quad (20)$$

B. Results

We have tested our algorithm in 2 different variations of the scenario that was described in the previous section. In each experiment we performed our simulations using 100 particles, temperature parameter $T = 0.1$ and inertia parameter $\alpha = 0.3$. In these experiments we observed thrashing and for that reason we had to use an inertia parameter smaller than one.

Our initial experiment was set to test if two sensors will be able to coordinate to maximize their utility. One of the



(a) Direction of the two sensors (b) Direction of the two sensors and the utility they gain

Fig. 5. Direction of the two sensors when they have to observe two opposed objects

sensors (sensor 1) was placed at the origin of the axis $(0,0)$, and the other one (sensor 2) in the opposing direction at position $(1,1)$ of the unit square as shown in Figure 5(a). Figure 5(b) illustrates the results of this experiment after 10 iterations of the algorithm. The y and x axis of the contour plot, in the right top corner of the figure, represents the actions of sensor 1 and sensor 2 respectively. The height of the contour plot describes the utility the agents will gain if they choose the joint action (x,y) . The two dark circles, at the lower left corner and the top right corner, represent the areas that the utility is maximized. In the bottom and the left of this figure we can see the histograms of the actions (in radians) the agents have chosen. We can observe the two sensors manage to coordinate and the mass of their particles create a joint action that is in one of the two areas that the utility is maximized. Furthermore from Figure 5(a) we can also observe that the sensors managed to coordinate and choose a different object to observe and so maximize their utility.

Our last experiment included one hundred sensors and one hundred objects in the unit square. In this experiment each object had a different random value between zero and one.

We allowed 200 iterations of this experiment, with temperature parameter $T = 1$. Figure 6 illustrates the utility that the sensors gained when they used SPC to coordinate (solid line) and the utility they gained when they were using $PC_{decentralised}$ (dot line). We can observe that SPC performs better than $PC_{decentralised}$ since the agents managed to coordinate to a solution that gives higher utility when they used SPC. Furthermore the SPC algorithm seems to converge to a solution since the payoffs are smooth but this is not the case for the $PC_{decentralised}$ algorithm.

		Hartman3			Hartman6		
		Average Utility	Evaluations	Failure %age	Average Utility	Evaluations	Failure %age
use	$PC_{centralized}$	-3.515 (0.0925)	350	34	-3.1180 (0.1615)	3500	70
stopping	$PC_{decentralized}$	-3.7933 (0.0284)	400	6	-3.2385 (0.0811)	3000	51
criterion	SPC	-3.8353 (0.0746)	400	10	-3.2673 (0.0696)	3500	20
fixed	$PC_{centralized}$	-3.8041 (0.0848)	500	31	-3.2221 (0.0962)	12500	49
iterations	$PC_{decentralized}$	-3.8581 (0.0193)	500	5	-3.2575 (0.0599)	12500	48
	SPC	-3.8424 (0.0385)	500	10	-3.3127 (0.0238)	12500	4

TABLE II
RESULTS FOR HARTMAN'S FUNCTION FOR DIFFERENT ALGORITHMS

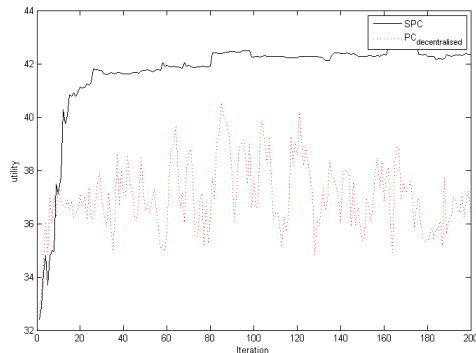


Fig. 6. Utility of SPC and $PC_{decentralised}$ for the case with 100 sensors and 100 objects.

VIII. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In conclusion, we have introduced a new variation of the probability collectives algorithm, using standard techniques from the statistics literature. In particular, kernel regression was used to smooth and extrapolate the utility for values of the sample we have not observed. In addition, we used SMC methods to propagate the sample we needed, using information that we had from previous time steps of the iterative procedure.

We tested our algorithm in the three dimensions Hartman's function using only 50 particles and SPC performed less well than the $PC_{decentralized}$. An explanation for this performance is the small number of particles that were used, since when we used bigger number of particles in the same example all the algorithms performed similarly. When we tested our algorithm in the more difficult example of the six dimensions SPC performed significantly better than the other two variations of PCs since it managed to converge to the area of global optimum the 96% of the trials when the best of the other two variations converged in there only in 52% of the trials.

Furthermore in the vehicle target assignment game we observed that the sensors manage to coordinate, and in particular in the scenario with the 100 sensors and 100 objects it converged to a solution with higher payoffs than $PC_{decentralised}$ which didn't manage to converge after 200 iterations

B. Future Works

Our ongoing work includes further comparisons of the proposed PCs algorithm with other kinds of functions, like badly scaled functions, that will allow us to have a more complete idea for the performance of the algorithm. Also we are going to test our algorithm in more complicated experiments. Finally, we are planning to develop an adaptive method to choose the temperature parameter T of our updating method.

REFERENCES

- [1] G. Arslan, J. Marden and J. Shamma. Autonomous Vehicle-Target Assignment: A Game Theoretical Formulation. *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, 2007, pp 584-596.
- [2] S. R. Bieniawski, I. M. Kroo and D. H. Wolpert. Discrete Continuous and Constrained Optimization Using Collectives. In *Proceedings of the 10th conference on American Institute of Aeronautics and Astronautics (AIAA)*, 2004.
- [3] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
- [4] A. W. Bowman and A. Azzalini. *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-Plus Illustrations* (Oxford Statistical Science Series). Oxford University Press, 1997.
- [5] Z. Cai. Weighted Nadaraya-Watson regression estimation. *Statistics & Probability Letters*, vol. 51, 2001, pp:307-318.
- [6] L. C. W. Dixon and G. P. Szego. In *The global Optimisation Problem: An Introduction. Towards Global Optimization*, Editors L.C.W. Dixon and G.P. Szego, vol. 2, 1978.
- [7] R. Douc, O. Capp and E. Moulines. Comparison of Resampling Schemes for Particle Filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, 2005 pp:64-69.
- [8] P. Del Moral, A. Doucet and A. Jasra. Sequential Monte Carlo Samplers. *Journal of the Royal Statistical Society: Series B*, vol. 68, 2006, pp 411-436.
- [9] E. Nadaraya. On Estimating Rgression. *Theory of Probability and its Applications*, vol 9, 1964, pp:141-142.
- [10] D. Rajnarayan, I. Kroo and D. H. Wolpert. Probability Collectives for Optimization of Computer Simulations. In *Proceedings of the AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2007.
- [11] I. Rezek, D. S. Leslie, S. Reece, S. J. Roberts, A. Rogers, R. K. Dash and N. R. Jennings. On Similarities between Inference in Game Theory and Machine Learning. *Journal of Artificial Intelligence Research*, vol. 33, 2008, pp 259-283.
- [12] G. Watson. Smooth regression analysis. *Sankhya Series A*, vol. 26, 1964, pp:359-372.
- [13] D. H. Wolpert. What Information Theory Says about Bounded Rational Best Response. In *The Complex Networks of Economic Interactions*, Editor A. Namatame, 2006, pp 293-306.
- [14] D. H. Wolpert and D. G. Rajnarayan. Parametric Learning and Monte Carlo Optimization. Available at <http://arxiv.org/abs/0704.1274>, 2007.
- [15] D. H. Wolpert, C. E. M. Strauss and D. G. Rajnarayan. Advances in Distributed Optimization Using Probability Collectives. *Advances in Complex Systems (ACS)*, vol. 9, 2006, pp 383-436.