Spatial Coordination Games for Large-Scale Visualization

Andre Ribeiro and Eiko Yoneki

University of Cambridge, The Computer Lab,
William Gates Building, 15 JJ Thomson Ave, Cambridge, UK CB30FD
{Andre.Ribeiro, Eiko.Yoneki}@cl.cam.ac.uk

Abstract. Dimensionality reduction ('visualization') is a central problem in statistics. Several of the most popular solutions grew out of interaction metaphors (springs, boids, neurons, etc.) We show that the problem can be framed as a game of coordination and solved with standard game-theoretic concepts. Nodes that are close in a (high-dimensional) graph must coordinate in a (low-dimensional) screen position. We derive a game solution, a GPU-parallel implementation and report visualization experiments in several datasets. The solution is a very practical application of game-theory in an important problem, with fast and low-stress embeddings.

Keywords: Dimensionality reduction, visualization, game-theory, belief revision, spatial coordination

1 Introduction

Most of the current practical visualization solutions make use of interaction metaphors (springs, boids, self-organizing neurons, etc.) among data-points. In this paper we give agents more strategic intelligence and consider whether a multi-agent perspective can bring in new connections and solutions to the problem.

Namely, the problem is to take a graph or high-dimensional distance matrix between data points as input and calculate a lower dimensional embedding of these points as output. General data practitioners, from diverse areas of knowledge, often use, for example, force-directed graph visualization, which is the main element in most popular graph visualization tools. Scientific practitioners and statisticians, on the other hand, tend to prefer less scalable and less exploratory solutions that have, instead, stronger guarantees. We depart from Multi-Dimensional Scaling (MDS), a classical statistical dimensionality reduction technique, and reach a solution that is as practical as force-directed systems, while maintaining (or improving) the quality of statistical solutions.

We look at the problem as a game where players have to decide which screen position to occupy. Given that they want to be far away on the screen to far away players on the graph (or distance matrix), their chosen position will depend on their expectations about what others will do. To solve the problem we thus study games, which we call Spatial Coordination Games, where player payoffs vary with mutual distances in a player-to-player basis. Players keep probability distributions over each others positions and update them with each individual movement.

2 Spatial Coordination Games

More precisely, we consider a game $G = [N, M, a_{ij}]$ with N players over a finite set of M positions and a pairwise distance-based payoff function, $a_{ij}(m, k)$ - with m and k as players i and j's respective positions, $i, j \in [N]$ [1], $m, k \in [M]$, $a : [N]^2 \times [M]^2 \to \mathbb{R}$ and $M, N \in \mathbb{N}^+$.

When $M = 1$, the Spatial Coordination Game is reduced to the consensus problem [24] (the solution is a single consensual position and players have incentive to conglomerate). The problem has applications in agent (spatial) sensing, formation, rendezvous, alignment, evacuation, flocking, coordinated decision making, etc.

When a_{ij} is uniform across players, $a_{ij}(m, .) = {}^1/_N$, the game is reduced to a congestion game [21] (all players have the exact same amount of influence on individual decisions, making payoffs proportional to the number of players using a resource and giving them incentive to disperse). The problem has applications in network routing (especially over wireless networks) and analysis.

We are interested in the general case when payoffs a_{ij} are not constant, but player-specific, and the game is played over a large set of positions and players, $M \gg 1$ and $N \gg 1$. This is the case of visualization - where players choose where to go based on where other individual players are, and not, for example, an absolute count of players (as in general congestion or the El-Farol Bar problem [20]). In this game a pure strategy profile specifies a player's position given all others' possible movements. A mixed profile assigns a probability (or belief) to each possible position, which players can review after observing others' beliefs.

Consider that player i has a probability distribution over positions m, $p_i(m)$, as mixed strategy. The player starts with a prior distribution and calculates its strategy based on the expected actions of all others $p_1(m), p_2(m), ..., p_N(m)$ a posteriori.

Player i can calculate the (expected) utility of choosing a position m as the probability of player j choosing a position k and the resultant (joint) utility $a_{ij}(m, k)$ in that case, $m, k \in [M]$. The expected utility reflects proximities across all possible *likely* placements. In a best-reply fashion, it is rational for the player to update its distribution at any point in the game proportional to its expected payoff,

$$p_i(m) \propto \sum_{j,k} p_j(k) a_{ij}(m, k) \tag{1}$$

[1] $[c]$ denotes a set with $c \in \mathbb{N}^+$ elements

which is iterated, with some prior distribution $p_i^0(m)$. The mutual, synchronous updates then create a coupled dynamical system that can be analyzed with game-theory.

The model has three parts: the players' utility functions, their belief revision strategy and their communicative strategy. We first formulate the payoff function $a_{ij}(m,k)$, then how players update their beliefs (i.e., the inherited dynamics of the game) and then how players can intervene (manipulate their private beliefs) to change the outcome of the game. We finally move on to implementation notes and experiments.

2.1 Stress Payoffs

The payoff for a player i varies with its distance to each other player j individually, and is derived from an input (weighted) graph or distance matrix D_{ij} - which we simply call 'graph' for short.

Specifically, the payoff $a_{i,j}(m,k)$ is the embedding's normalized stress [18], the difference between the (low-dimensional) spatial distance $d(m,k)$ and the (high-dimensional) distance $D(i,j)$ across placements.

$$a_{ij}(m,k) = \frac{\sum_{ij}[d_{ij}(m,k) - D(i,j)]^2}{\sum_{ij} d_{ij}(m,k)^2} \tag{2}$$

The measure explicitly indicates the difference between the distances in the input (high-dimensional) and the output (low-dimensional) positions. The values lie between zero and one (assuming a normalized input); the smaller, the better an embedding represents the high-dimensional data. This quantity can be calculated once and independently of the next equations (and thus can be precalculated and stored in look-up tables during the game). The squared euclidean distance is used for $d(m,k)$ in all experiments.

2.2 Update Strategy

A central issue is how exactly players change their distributions (redistribute their probabilistic mass) after observing others' beliefs, while keeping their own beliefs altogether stochastic. If the payoffs are normalized $a_{ij} \in [0,1]$, symmetric and do not change, then

$$p_i(m) = \sum_{j,k} \frac{1}{\sum_l p_j(k) a_{ij}(l,k)} p_j(k) a_{ij}(m,k) \tag{3}$$

with $l \in [M]$.

A player using Eq.(3) change its beliefs according to expected payoffs. For the type of problem studied here (where payoffs are coordinative), Eq.(3) using expected payoffs has advantages. This is because in these problems it's advantageous for players to readjust not based on their immediate gains but also considering what others might actually do, which seemingly allows them to explore further opportunities for coordination.

We want to derive next a differential equation $\dot{p}_i(m)$ which gives players iterative updating rules and allow us to better understand the asymptotic game behavior. The solution can be derived (with regularity assumptions) by enforcing that $\sum_i p_i(m) = 1$ and thus that $\sum_i \dot{p}_i(m) = 0$.

The denominator in Eq.(3) unfortunately makes updates impractical in large games. We can simplify the relationship between players strategies by assuming parametric forms to the belief distributions. Let players' beliefs have an exponential form with constant rate of change (derivative) $a_{ij}(m, k)$.

$$\sum_{j,k} \frac{1}{\sum_l p_j(k) e^{a_{ij}(l,k)}} p_j(k) e^{a_{ij}(m,k)} \tag{4}$$

This, together with the second-order regularity assumption, leads to a simpler form $\dot{p}_i(m) = p_i(m)(v_i(m, k) - \alpha(m, k))$, where $v_i(m, k)$ is the right-hand-side likelihood of Eq.(1) and $\alpha(m, k)$ is a renormalizing constant. Under these conditions, the normalizing constant $\alpha(m, k)$ exists and can be derived with a few simple logarithmic operations, leading to the final form of the belief updating strategy[2] and the multi-population replicator dynamics [30]:

$$\dot{p}_i(m) = p_i(m)[\sum_{i \neq j} \sum_k a_{ij}(m, k) p_j(k) - \sum_{i \neq j} \sum_{l,k} a_{ij}(l, k) p_i(l) p_j(k)], \tag{5}$$

with $i, j \in [N]$ and $l, m, k \in [M]$. The interpretation, however, is slightly different from typical multi-population replicator dynamics. We view the dynamics as a model of learning where the population m frequency correspond to an individual player's probability of playing m at time t. Thus $p_i(m)$ is its mixed strategy at t. And the payoff received from the rest of the players, α, is a renormalizing constant.

For analysis, we are interested in the situation in which *every* player chooses a pure strategy - i.e., for every player i, $p_i(m)$ is concentrated on a single position. That is, while having the choice from the set of mixed strategies, players choose a particular position with little uncertainty. These are "corners" of the M-dimensional probability vectors $p(m)$ simplex. Corners are trivially equilibrium points. Other equilibria, that are not corners, are sometimes named interior equilibria. A further attractiveness of the replicator dynamic is that it is typically well behaved. All asymptotically stable attractors must include corners, and if compact, these are the game Nash equilibria [13]. That is, a trajectory either converges to a corner, or eternally moves around in the interior of the simplex. However, when asymptotically stable corners do exist, their basin of attraction typically cover most of the simplex. Although we do not require equilibrium to reach the final game solution (see Implementation Section) and currently focus on demonstrating the practically of the game, we observe this empirically (the game typically converges to a corner).

[2] See [12] [1] for a similar derivation and further details on the relationship between the exponential family and the replicator dynamics

2.3 Communication Strategy

We introduce the notion of a player's probabilistic intervention in this game. An intervention [11] is an experimental change on a player's model (e.g., a player clamping of the variable $\mathbf{m}$ to a value k). Causal interventions are often notated as $P(X|do(y))$ [25], we use $p_j(m|do(i,k))$ to denote player j's distribution when i's distribution is clamped to a value k. Because an intervention affects the game outcome simply by changing others' mental models, it's natural to think of it as a model for communication ('I will be at position x').

If players are distributed on the graph and will favor positions according to (5), it is then the order of interventions that remains to be optimized. We define the player's risk in the game in terms of its uncertainty across positions. We then propose a minmax criterion that chooses the intervention with minimum risk across players at each step. In the resulting game, a competitive game is progressively modified through 'communication'(a cycle of intervention followed by equilibrium calculation, repeated to a desired uncertainty level). This way, an embedding is defined incrementally.

We imagine players incur a risk when they have to settle for a given position (i.e., it's beneficiary to keep their individual options open, or their own position uncertainty high). We then take the individual risk to be

$$R(i) = -log_2 p_i(m) \tag{6}$$

The intervention $do(i,k)$'s collective risk is the maximal risk across all players, $argmax_{(i,k)} R(j|do(i,k))$. Players then choose the intervention with minimum collective risk at each iteration,

$$do(i,k)^* = minargmax_{(i,k)} R(j|do(i,k)) \tag{7}$$

The criterion has a cooperative interpretation, in which all players incur the risk of the worst player in a given intervention. Below we show how to calculate Eq. (5) and (7) in a parallel GPU [28] implementation. We also explore the spatial nature of the problem to devise probability distributions with increasing resolution (i.e., a multi-level scheme that alleviates the solution's computational requirements). Finally, we show how to use the model for visualization and report results.

3 Related work

'Classic' (or Torgerson) metric MDS is often done by transforming distances into similarities and performing PCA (e.g., singular-value-decomposition) on those. PCA is sometimes taken as the simplest possible MDS algorithm. In particular for real world data, which is typically nonlinear, nonlinear techniques may offer a definite advantage. In the spatial domain, spatial data with local correlations are especially problematic for PCA, and it is similar to simply taking a Fourier transform and filtering out low-frequency components [23]. Linear methods (like classical MDS) are generally not good at modeling curved manifolds, often only preserving distances between widely separated points and loosing local structure [27]. Currently, Kruskal's stress [18] is the most common measure of goodness-of-fit for a non-metric MDS embedding, and serves as objective function to minimize in majorization approaches ('smacof') [9], which perform well in a wider range of domains.

Spring layout algorithms are probably the most practical and popular algorithms for drawing general graphs, as proposed by Eades [6]. Since, his method has been revisited and improved in a variety of ways [16]. In general, force-directed algorithms can produce good results for small graphs, but do not scale well. Large graphs often result in the energy function been trapped in local minima. Additionally, force-directed algorithms lack predictability, two different runs with the same input may lead to disparate results. This inconsistency can be a serious problem in visualization. Annealing [15] has show to be a uniquely effective way to globally optimize both force-directed [4] and stress majorization schemes [2]. The player-guided, progressive reduction of uncertainty studied here provides an alternative to annealing schedules. In the experimental section, we compare the performance of these methods to the suggested.

Self-organizing maps (SOMs) [17] also use the metaphor of a competitive process ('game') between agents, inspired by neural behavior. A neural network is trained to produce a low-dimensional (typically 2D), discretized representation of the training samples input space. For each training point, a neuron is selected, and weights in its neighborhood are moved in the same direction ('similar items tend to excite adjacent neurons'). More neurons go to regions with high training sample concentration, and fewer where the samples are scarce. SOMs have been shown to have advantages [31] over more conventional feature extraction methods such as PCA. The neurons' exact location in the grid/graph, however, constrain their interaction and capture completely their mutual model. If players are given the latitude to move around freely, they require foresight (and a model) on others' behavior to coordinate. In Boids [26, 22], players calculate individual positions using simple following and flocking behaviors. Players are homogeneous (have no preferential proximity among them) and need not to reason about each other actions.

In a polymatrix games [14], there is a utility matrix for every pair of players (i, j), each a separate component of player i's utility. Polymatrix games always have at least one mixed Nash equilibrium. Erdem and Pelillo [7] solved a generic polymatrix game using evolutionary game-theory (i.e., the replicator dynamics) to estimate a classification decision over partially-observed values in a set of prior graph-structured exemplars (i.e., transduction) with interesting results.

Congestion games were first proposed by Rosenthal [21]. In them, the payoff of each player depends on the resources it chooses and the number of players choosing the same resource. In general, players cannot communicate and have no uncertainty over each other's actions (only an observed and determined congestion on the chosen resource). More recently, the notion of uncertainty over a resource's congestion was explored [10]. Uncertainties are, however, static (i.e., cannot change or be strategically manipulated) - the inverse assumption of the current approach.

4 Implementation

4.1 Parallel Implementation

We implement a parallel GPU [28] version of the game where each thread-row correspond to players' probabilities $p_i(m)$ under a different intervention. Each individual intervention $y = do(i,k)$ is described jointly by a player i and a (clamped) position k, and we have C interventions.

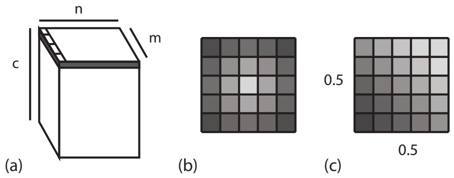

Fig. 1. (a) Player/threads layout, (b) radial prior, (c) multi-level prior.

The resulting thread layout for the game is a 3-dimensional $C \times N \times M$ matrix (Fig.1a). Each row (i,m) correspond to an intervention in i and each column to a player j. A tread-cell contains the likelihood that player j (column) will go to a given position m when the intervention (i,m) is applied (i.e., that an arbitrary player will go to an arbitrary position). The value of $d(m,k)$ in Eq.(2) is fixed for each cell. We use the Euclidian distance between the centers of a 2-dimensional $\sqrt{M} \times \sqrt{M}$ cells regular grid (Fig.1b). The value for a cell $p(i,k,m)$ can then be calculated from this constant, $D(i,j)$, and $N-1$ vectors $p(i,k,.)$, where $D(i,j)$ is taken from the input distance-matrix and $p(i,k,.)$ from neighboring cells. Since this corresponds to $N \times M$ parallel vector multiplications, common GPU vector-vector optimizations can be used [3]. The individual values can then be made stochastic again by a depth (i.e., across m values) and row prefix-sums and a parallel division. We let the system run for T cycles, with values $p^t(i,j,m)$. We do not require, however, for players to reach equilibrium to stop the game, although we have observed that for a value $t > 10$ they often do. We look at T, instead, as a time constraint on the solution.

For visualization, interventions are exhaustively enumerated and an optimal intervention is selected according to the risk criterion, Eq.(7). This correspond to a row selection, based on a calculated property f over rows, $argminf[p^t(i,j,m)]$. This row selection can be implemented trivially with parallel arithmetic operations and prefix-sums.

4.2 Priors and Multi-level Games

For realtime performance, we employ a second set of optimizations. Instead of playing the game over all M positions, we first play the game over a small grid $M' \ll M$. Then play the game again in each individual grid position, limited to players there but with further M' positions. We repeat this D_{max} times, generating an increasingly finer grid. The probabilistic priors serve to connect the levels, allowing players to take a summarized version of the previous (higher-level) game into consideration.

In practice, it's hard however to specify the prior distribution parametrically. We imagine then that the grid is a metal plate with discrete heat sources (each located in the middle position of two cell's border and with temperatures varying $[0, 1]$) [5]. With the sources positions and the heat equation, it's easy to calculate the temperatures at plate positions (i.e., the new level prior probabilities) by interpolating temperatures in parallel (Sanders and Kandrot, 2011).

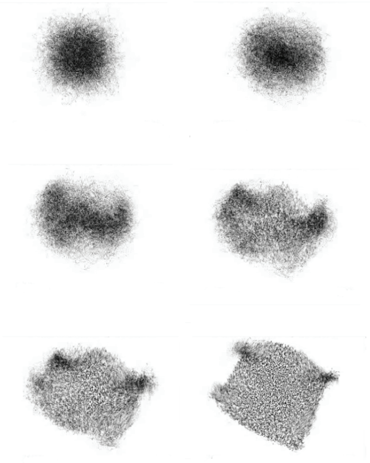

Fig. 2. 4.5M players in a synthetic dataset (subsequent interventions).

Let c^d denote a game's congestion - the number of players who intervened to position m in a game at level d, and with $c^{D_{max}} = N$. For the first-level prior, we place a single heat source at the grid center with temperature 0.5. For subsequent priors, we place sources at the middle positions of each adjacent square side with temperatures

$$1 - \frac{c^d(m)}{c^{d-1}(m)},$$

the normalized congestion on the prior level. Fig.1b shows a first-level prior and Fig.1c a prior with two previous neighbours (each with equal congestion) in a grid with $5x5 = 25$ positions. By breaking large games into independent smaller ones, we are able consider the exhaustive set of interventions C' in a level, for each position m ($C' = C^d(m) \times M'$).

4.3 Rendering

For rendering, players have no single, determinate position in this game (only a mixture of positions). We render player i's screen position $x_i \in \mathbb{R}^2$ (vector) as the average position across all M positions, weighted by the mixed strategy profile:

$$x_i = \sum_m p_i(m) x_c(m) \tag{8}$$

where $x_c(m)$ is the vector from the grid center to the position m in the regular, squared grid. Each intervention (and subsequent competitive play) changes incrementally others' mixtures, and thus positions.

Each level has a different set of vectors $x_c(m)$ from the level cell center to all lower level positions. The player's final position is then the sum of vectors across all levels. The metaphor of single-body attraction and repulsion between nodes, for example, is then replaced by the interventions on players' probability distributions. And players' velocity and acceleration are 'replaced' by their uncertainty. With a radial prior (Fig.1b), players start at the screen center and spread out. Typically, the first interventions are more 'catastrophic' and very noticeable, with latter ones barely. And equilibrium states are apparent (i.e., players halt movement).

5 Experiments

A difficulty with visualization is that there are no consensual benchmark measures (and often no comparison measures at all are given). We start with a few synthetic data. We then test the game ('coord-game') in several machine learning datasets and compare to standard-MDS and the deterministic annealing approach of [2] which do report stress measures. We also report results in a Facebook dataset with three networks. To reveal the structure of the output embedding more explicitly, we additionally reproduce the MNIST characters visualization of [29]. We finally briefly discuss running times. Experiments ran on an Intel Core 2 3 GHz CPU with 4 GB of memory and an nVidia 8800GTX graphics card with 512MB of texture memory. Timings do not include file loading time.

We started by generating a batch of synthetic datasets, consisting of 4.5 million players distributed in a 2D grid embedded in 7 dimensions. We also tested the effects of adding noise to this grid (5% noise in a third dimension). Fig.3a shows one run, comparing the performance (expected stress through number

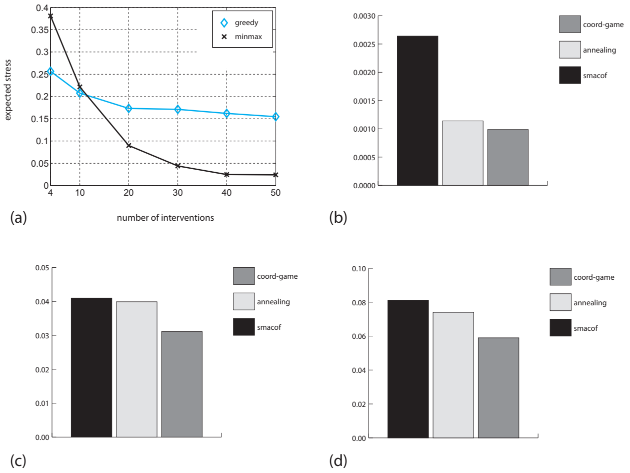

Fig. 3. (a) stress trough time, (b) iris dataset stress, (c) yeast dataset stress, (d) metagenomics dataset stress.

of interventions) of the min-max criterion, Eq.(7) compared to a greedy criteria, $min_{(i,k,j)} R(j|do(i,k))$. By avoiding riskier placements (i.e., ones that would constraint unfavorably individual, future others) the criterion leads to an overall layout that is closer to the high-dimensional, more quickly. The min-max criterion seems very resistant to local minima, when compared to annealing. This is reflected on the stress results reported below.

The Iris dataset (available on the UCI ML Repository) has 150 points in 4 dimensions (4 attributes over 3 classes of flowers). It's one of the most famous datasets in both machine learning and statistics [8]. Its dimensionality is speculated to be marginally greater than the embedding dimension (with two of the classes linearly separable), both the global structure and the local proximity of the data may be important but neither can be reconstructed without some distortion (not being perfectly separable). Some cluster structure can be distinguished.

Fig.4 shows the output map as a scatterplot (all experiments and figures ran with radial prior distribution in Fig.1c, $M = 5 \times 5 = 25$, $D_{max} = 3$ and $T = 10$). There is class information with each datapoint, but it is only used to label players in the figure (and no way influences their positions). Symbols (asterisk, cross and circle) clarifies how well the map preserves the similarities within each class. Qualitatively, the spatial embedding clearly separates the symbol-coded groups.

The yeast dataset has approximately three time more data points (1,484 points) than Iris, each 8-dimensional. The metagenomics data has twenty times more points (30,000 points). Fig.3 lists the obtained normalized stress ('coord-game'), together with those obtained with classical Scaling by Majorizing a Con-

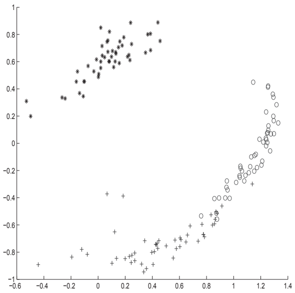

Fig. 4. Iris Dataset scatterplot (crosses, circles, asterisk coding from ground truth).

vex Function [9] ('smacof') and the more recent Deterministic Annealing with Iterative Majorization [2] ('annealing'). Since the later two algorithms have randomizations, these are average performances over 50 trial runs. For Iris, the stress obtained with coord-games is over twice as low as smacof and outperforms deterministic annealing, with a final normalized stress of 0.00111. This suggests that coord-game generates low-dimensional embeddings that are more accurate representations of these high-dimensional data-sets. Error bars across trials in [2] indicate variation across the trials for smacof and annealing which is not observed for coord-game.

Next, we run the algorithm in a set of 3 collegiate facebook social graphs with 1005, 10078 and 13455 nodes. We only consider users with at least 10 friends, and all information but the plain friendship graph is ignored. The gain in performance is more dramatic, Fig.5. Force-directed and MDS algorithms tend to look like a ball of yarn - a dense mess with no visually discernible structure - for networks with over 1000 nodes. We can however see structure in placements for these graphs using coord-game, Fig.6. This is described more precisely by the stress measures in Fig.5. MDS Algorithms [9] are $O(N^2)$ and obtaining comparison measures for networks larger than 10000 nodes is difficult without further optimization schemes.

The MNIST database of 28 × 28 (scaled and centered) handwritten digits (training set) has 60,000 examples. Although it's used mostly on classification tasks, it's interesting to take advantage of the visual difference between digits to make the mapped relationships clearer [29]. We downsampled and Gaussian smoothed digits to 16 × 16 bitmaps. We then selected 900 of the images, the 100 first examples for each digit in the original distribution [19]. Fig.5a contains the overall resulting embedding. Fig.5b,c show two bordering regions in detail (between 0-6 and 8-9-4 clusters). The separation between the digit classes is very clear. The few digits close to the wrong cluster are distorted, almost unrecognizable characters (Fig.5c highlights, with rectangles, two examples, both from the '9' training subset).

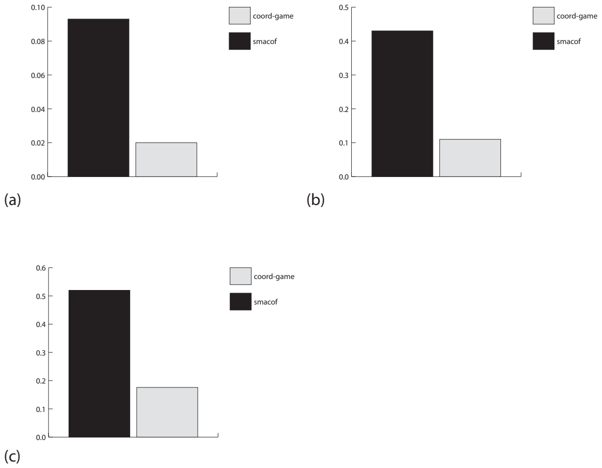

Fig. 5. (a) facebook friends network-1 stress, (c) network-2 stress, (d) network-3 stress.

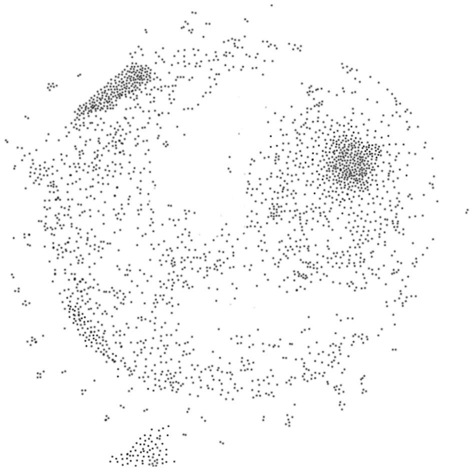

Fig. 6. Facebook friends network scatterplot with 10078 nodes.

The coord-game system is an order of magnitude faster than others with comparable performance (see, for example, [29]). It takes approximately 0.8 seconds, 1.9 and 8.3 seconds for the final placement of the Iris, Yeast and Metagenomics datasets. The visualization is iterative, and, perhaps more relevant, is the time per intervention, which is of 0.61ms/intervention with $M = 25$ and $T = 10$. The effective parameters T and M (resp., the equilibrium time constrain and square grid size) for these two systems offer useful speed-quality tradeoffs.

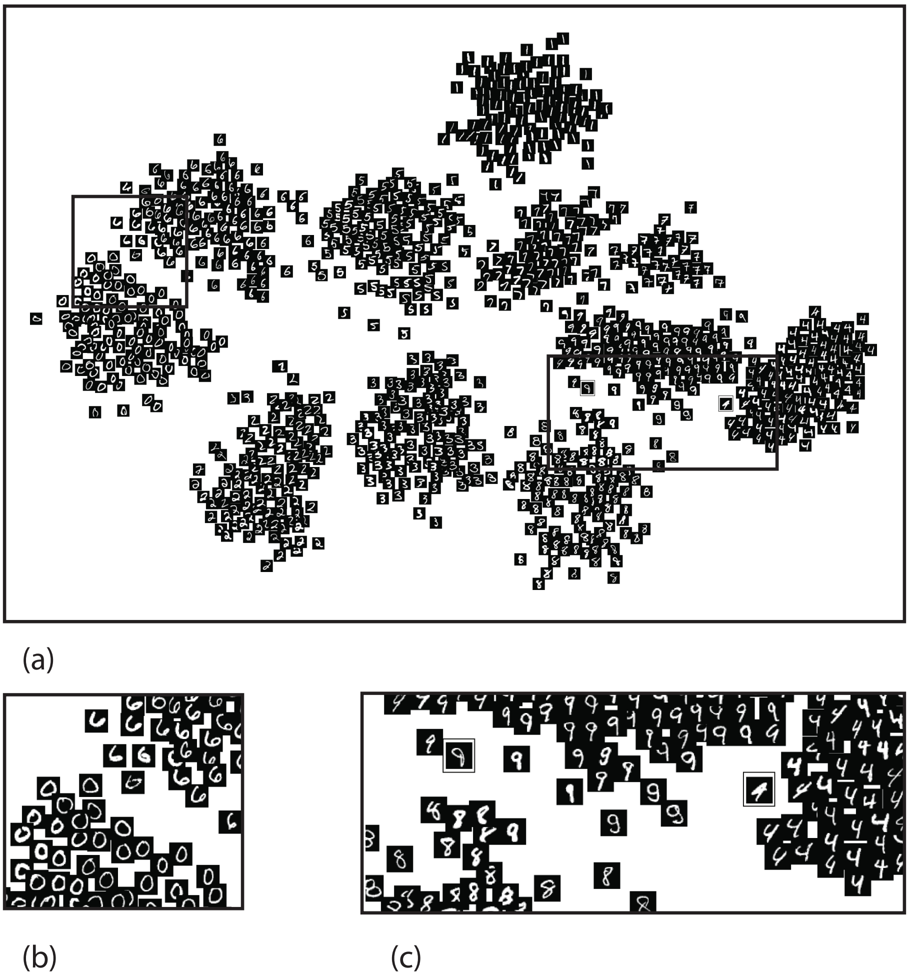

Fig. 7. MNIST dataset.

6 Conclusion

At the heart of the article is a game-theoretic model of many-players coordination using graphs (with models for players belief revision and communication). A common critic of game-theory is that it is mostly 'toy mathemathics'. We demonstrate that the perspective can be practical for visualization. Dimensionality reduction is closely related to a range of important topics such as compression and discriminant analysis. We have also applied the model to large-scale spatial coordination problems with shops check-in (Foursquare) and human travel (Flickr) data with surprising results. The work thus opens up new opportunities for game-theory in both Machine Learning and in new applications.

References

1. E. Akin. Exponential families and game dynamics. *Canadian Journal of Mathematics*, XXXIV(2), 1982.
2. S. Bae, J. Qiu, and G. Fox. Multidimensional scaling by deterministic annealing with iterative majorization algorithm. *Proceedings of IEEE eScience 2010 Conference*, 2010.
3. N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on cuda. *NVIDIA Technical Report NVR-2008-004*, 2008.
4. R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15, 1996.
5. P. Doyle and L. Snell. Random walks and electric networks. *Carus mathematical monographs, Mathematical Association of America*, 22, 1984.
6. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42, 1984.
7. A. Erdem and M. Pelillo. Graph transduction as a noncooperative game. *Neural Computation*, 24, 2012.
8. R. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics, 7, Part II*, 1936.
9. E. Gansner, Y. Koren, and S. North. Graph drawing by stress minimization. *Symposium on Graph Drawing*, 2004.
10. C. Georgiou, T. Pavlides, and A. Philippou. Network uncertainty in selfish routing. *IPDPS*, 25, 2006.
11. J. Halpern and J. Pearl. Causes and explanations: A structural-model approach. part i: Causes. *UAI*, 2001.
12. M. Harper. Information geometry and evolutionary game theory. *CoRR*, 2009.
13. J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics.* Cambridge University Press, 1998.
14. J. Howson. Equilibria of polymatrix games. *Management Science*, 18, 1972.
15. S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220, 1983.
16. S. Kobourov. *Force-Directed Drawing Algorithms.* 2012.
17. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 1982.
18. J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29, 1964.
19. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 1998.

20. H. Lus, C. Onur Aydn, S. Keten, H. Ismail Unsal, and A. Rana Atlgan. El farol revisited. *Physica A: Statistical Mechanics and its Applications*, 346, 2005.
21. I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13, 1996.
22. A. Moere. Time-varying data visualization using information flocking boids. *Information Visualization (InfoVis)*, 10-12, 2004.
23. J. Novembre and M. Stephens. Interpreting principal component analyses of spatial population genetic variation. *Nature Genetics*, 40, 2008.
24. R. Olfati-Saber, A. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95, 2007.
25. J. Pearl. *Causality.* Cambridge University Press, 2000.
26. C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics, SIGGRAPH*, 25, 1987.
27. S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.
28. J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming.* Addison-Wesley, 2011.
29. L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9, 2008.
30. J. Weibull. *Evolutionary game theory.* MIT Press, 1995.
31. L. Yonggang, R. Weisberg, and M. Christopher. Performance evaluation of the self-organizing map for feature extraction. *Journal of Geophysical Research*, 111, 2006.