Ranking with Hamiltonian Dynamics

W. Garrett Mitchener[a]

[a]*Department of Mathematics, College of Charleston, 66 George St., Charleston, SC 29424*

Abstract

In data science, a ranking or linear ordering problem is to place items into a linear order based on comparison data. In this article, we consider a Hamiltonian system, in which particles, representing the items to be ordered, exert forces on each other as determined by the comparison data. An ordering of the items can be derived from the relative positions of the particles in any state of the system. The Hamiltonian is designed so that states with low potential energy yield orderings that agree strongly with the comparison data. Although the dynamics are related to the Toda lattice, no usable ranking information seems to be available from that variation of the Hamiltonian. Instead, a Toda-like Hamiltonian plus a confinement potential yields better results. Several algorithms based on Hamiltonian trajectories, minimization of a ranking potential, and a Hamiltonian Markov chain are compared to the widely used RankBoost algorithm. They all perform relatively well on synthetic test data and on problems from a library of test cases, but none is clearly the best in all circumstances. The trajectory-based methods and Markov chain show interesting dynamics. Furthermore, since these methods attach particle positions to the items as well as an ordering, the distances between the particles indicate how rankable the items are.

Keywords: Hamiltonian dynamics, ranking, linear ordering

1. Introduction

In data science, a *ranking* or *linear ordering* problem is the task of putting items into a linear order, from least to greatest, that is somehow consistent with given comparison data. For example, given outcomes of games played by several sporting teams, it is of interest to rank them from weakest to strongest. The comparison data may be incomplete, in that not every team has played every other team. It may also be inconsistent, that is, the data may not obey any sort of transitivity law. For example, there may be a *cycle*, in which team A won against team B, and B won against C, but C won against A. The data may also include some measure of strength, such as a point spread, that indicates how much stronger the victor is compared to the loser.

Email address: mitchenerg@cofc.edu (W. Garrett Mitchener)

There are many reasons one might want to solve a ranking problem. For the purposes of this article, we adopt the perspective that the primary goal is to find a linear ordering that can be used to predict the outcomes of future comparisons among the same items. Since the data may contain noise, there is a danger that the selected ordering overfits it, that is, the ordering may be in strong agreement with the data, but is less suitable for prediction than some other ordering that agrees more weakly with the data. For example, suppose that in general, A defeats B with probability 60%, B defeats C with probability 60%, and A defeats C with probability 80%, so the best predictions come from the ordering $C < B < A$. However, the data from a tournament may consist of $\{C < B, C < A, A < B\}$ because by chance there was an upset in the game between A and C, yielding the suboptimal ordering $C < A < B$.

One way to formalize the ranking problem is to formulate the data as a matrix W, in which the entry $w_{jk} \geq 0$ represents how strongly the data suggests that item j should be ordered before k. Entries can be binary, that is, $w_{jk} = 1$ if team j lost to team k and 0 otherwise. Alternatively, w_{jk} could be the number of points by which team k won. An ordering of the items is represented by a permutation τ, interpreted as a function from item indices to their positions in the ordering. An ordering is *perfectly consistent with* W if

$$\forall\, j, k, l :\ w_{jk} > 0 \implies \tau(j) < \tau(k) \tag{1}$$

The matrix W can be interpreted as a weighted adjacency matrix for a directed graph, where $w_{jk} > 0$ when there is an edge from vertex j to vertex k. The matrix is said to be *consistent with transitivity* if

$$\forall\, j, k :\ w_{jk} > 0 \wedge w_{kl} > 0 \implies w_{jl} > 0 \tag{2}$$

in which case the corresponding digraph is acyclic, and the well known topological sorting algorithm provides linear orderings of the items that are perfectly consistent with W. If the items are renumbered from least to greatest according to one of these orderings, the columns and rows of W will be permuted such that all non-zero entries are in the upper right triangle. That observation motivates the following standard definition of the *tournament score* $\mathrm{TS}(\tau; W)$ as a measure of how well a linear ordering reflects the data. It takes a weight matrix W and a permutation τ, and produces the sum of weights in the upper triangle of the result of permuting the rows and columns of W with τ:

$$\mathrm{TS}(\tau; W) = \sum_{\tau(j) < \tau(k)} w_{jk} \tag{3}$$

One approach to ranking is to seek a τ that maximizes TS. Since there are $n!$ permutations τ to consider for n items, finding the absolute maximum quickly becomes computationally infeasible as n grows, so a variety of heuristics and integer programming techniques are used to find good τ's in a reasonable amount of time. In [1], for example, (3) is the objective function for a $\{0, 1\}$-integer program that represents a ranking problem. (Alternatively, in [2], an integer

program is used to find a permutation that minimizes the sum of ordering violations.)

For this article, a different approach is taken. Each item j is represented by a particle with some position q_j and momentum p_j. Given any position vector q, an ordering of the items can be obtained by sorting them from least to greatest, resulting in a permutation τ,

$$\forall\, j < k : \; q_{\tau(j)} \leq q_{\tau(k)} \tag{4}$$

A ranking potential energy function is defined based on W and used as part of a Hamiltonian. As the particles move, they experience forces such that if $w_{jk} > 0$, particle j is pushed to the left of particle k. In the language of statistical learning theory, the ranking potential is an empirical risk function, assembled as the sum of loss terms, one for each given pairwise comparison. In addition, an adjustable confinement potential imposes regularity by keeping the positions from growing uncontrollably.

Overall, the ranking potential is low when the particles are positioned in an order that is harmonious with the data W. That is, if there is strong evidence within the data that item j is weaker than item k, the corresponding particles should satisfy $q_j < q_k$ at most times. Thus, a reasonable linear ordering of the items is obtained by following a trajectory to a system state (q, p) with low potential energy, and sorting q to yield a permutation τ.

Alternatively, one could directly search for a configuration of the particles that minimizes the ranking potential, using gradient descent or other optimization technique. Such a direct search may settle at a local minimum near the initial condition that yields poor results, or it may find a minimum that overfits the data. It would then seem necessary to look at many initial conditions to improve the chances of finding a better local minimum. A possible advantage of Hamiltonian dynamics is that they can be chaotic, in which case trajectories explore the local minima of the ranking potential automatically.

We will explore several potentially useful algorithms based on minimizing R. In section 2, the ranking potential, confinement potential, and Hamiltonian are formulated, and several related ranking algorithms are formulated. Although the ranking problem itself is of interest, this article will focus on developing interesting dynamical systems related to ranking problems. To begin with, the ranking potential strongly resembles the potential of the well known Toda lattice. This connection is explored in section 3, but it turns out that the isospectral flow associated with the Toda lattice has properties that make it unsuitable for ranking. Cycles in comparison data are the essential obstacle to ordering items. In section 4, the Hamiltonian dynamics are worked out for the case of a cycle, and as expected, the origin is a nonlinear center. A few low-dimensional cases are analyzed in section 5, confirming that Hamiltonian dynamics yield intuitively correct results.

In section 6 Hamiltonian algorithms are compared to direct minimization of the ranking potential and to the widely used RankBoost algorithm on synthetic test problems. There are several variations of RankBoost [3, 4, 5] but

for simplicity, we will use the original formulation, as detailed in appendix Appendix A. The Hamiltonian methods perform reasonably well in comparison with RankBoost. Directly minimizing the ranking potential generally results in slightly worse performance, as it seems to be overfitting the data.

Hamiltonian dynamics have also been used to construct Markov chains for Monte Carlo methods [6, 7]. In section 7, such a Markov chain is used to generate sample position vectors q from a probability distribution that favors positions with low ranking potential. It turns out that, when tested on the synthetic data from section 6, the q's generated by the Markov chain do not achieve the low ranking potential that the other algorithms do, and the resulting orderings are generally inferior.

Further test cases on real-world data are given in section 9. The algorithms are compared on a standard corpus of ranking problems derived from economic data. RankBoost, direct minimization of the ranking potential, and the Hamiltonian Markov chain method yield higher tournament scores than the trajectory-based methods. It is possible that the trajectory-based methods do not perform as well on that kind of problem in general, but it may also be the case that other algorithms are overfitting the data to achieve higher TS. An example from college basketball reveals that somewhat different rankings arise naturally from different objective functions. A prediction problem from a season of professional American football illustrates how time-dependent team strengths and a high probability of upsets can easily lead to bad predictions.

Since each item to be ranked is associated with a particle in these dynamical systems, the positions of the particles can indicate how much confidence the algorithm has in the ordering. If two particles are well separated, it has high confidence in the order the associated items, but if they are close together, it is less certain. The particles should spread out over a relatively wide interval exactly when the data is highly rankable, that is, generally compatible with some underlying linear ordering [8]. The particles should be clustered when the data is generally disordered. RankBoost produces a function that maps items to real numbers, so it too provides a measure of confidence. In section 8, the various ranking algorithms are tested on synthetic data generated by three different methods. Statistics indicate that for all the algorithms under consideration, the intuitively more rankable data sets do result in greater particle spread.

2. Exponential potential and Hamiltonian dynamics

Given an $n \times n$ weight matrix W with entries $w_{jk} \geq 0$, and a vector of particle positions $q \in \mathbb{R}^n$, the *ranking potential* R is defined to be

$$R(q; W, \gamma_{\mathrm{r}}) = \sum_{j,k} w_{jk} \mathrm{e}^{\gamma_{\mathrm{r}}(q_j - q_k)} \tag{5}$$

where $\gamma_{\mathrm{r}} \geq 0$ is a scaling parameter. If $w_{jk} > 0$, then q_j must be less than q_k for the (j, k) term in the sum to be small. Thus, to achieve a low value of R, particles must be positioned so as to force as many of the non-zero terms

as possible to be small, with larger w_{jk} terms wielding greater influence. In the language of statistical learning theory, R is an empirical risk function and the terms of the sum are losses. We will consider several algorithms based on finding particle configurations for which R is low. The implementations are in Julia, and are available at the author's web site[1].

As an alternative to (5), one could use a risk function of the form

$$\sum_{j,k} \mathrm{e}^{w_{jk}(q_j - q_k)} \tag{6}$$

However, the connection to the Toda lattice explored in section 3 would not be possible. We will therefore focus on (5) in this article.

The most straightforward approach is to find the *direct minimum of R,* that is, use a numerical method to find q^{DMR} that is a high-quality local minimum of R. The permutation obtained by sorting the indices according to the components of q^{DMR} will be denoted τ^{DMR}. Good results are obtained from the `Optim.jl` package for the Julia programming language, using the `LBGFS()` method specification with all default options and leaving it up to the `optimize()` function to numerically approximate the gradient [9]. Experiments using the default Nelder-Mead method yielded distinctly inferior results.

It is possible that R has a tight lower bound, but no vector q achieves it. For example, assume a best case scenario in which W is consistent with transitivity and has at least one non-zero entry. Then there exists a permutation τ that is perfectly consistent with W. For any $s > 0$, the positions can be assigned values $q_j = s\tau(j)$, so that $w_{jk} > 0 \implies q_j < q_k$. Large enough values of s then make each non-zero term of R arbitrarily small, but never zero, by pushing the particles off to infinity at different rates. When computing q^{DMR} under such circumstances, the change in R resulting from adjustments to q eventually becomes exponentially small, at which point the algorithm halts gracefully and yields a reasonable permutation τ^{DMR}.

Using Hamiltonian dynamics, it is possible to prevent the particles from escaping to infinity by adding a *confinement potential.* There are many choices, but in keeping with the exponential structure of R, let us use

$$C(q; \gamma_{\mathrm{c}}) = \sum_j \cosh\left(\gamma_{\mathrm{c}} q_j\right) \tag{7}$$

where $\gamma_{\mathrm{c}} > 0$ is another scaling parameter. In the language of statistical learning theory, $C(q)$ is a regularization term.

Adding the usual kinetic energy term and weighting the potential energy terms by factors $\alpha_{\mathrm{r}} \geq 0$ and $\alpha_{\mathrm{c}} \geq 0$, the complete Hamiltonian is then

$$H = \frac{1}{2} \sum_j p_j^2 + \alpha_{\mathrm{r}} R(q) + \alpha_{\mathrm{c}} C(q) \tag{8}$$

[1] http://mitchenerg.people.cofc.edu

Without the ranking potential, the Hamiltonian is that of a set of uncoupled nonlinear oscillators. The comparison data provides coupling forces. Roughly, each particle oscillates about some point, but positive w_{jk} terms generally push q_j to the left and q_k to the right.

To choose a specific ordering from a trajectory $(q(t), p(t))$, there are two obvious things to try. The first is the *trajectory minimum ranking potential* method (TMR). From a trajectory, build a numerical representation of $R(q(t))$ over some finite time interval $t \in [0, T]$, and search for the time t^{TMR} at which $R(q(t))$ achieves its minimum value R^{TMR}. Calculating the minimum of R along a trajectory in this way is potentially a much less resource-intensive computation than a direct n-dimensional search for q^{DMR}. However, finding a suitable local minimum requires some care, as $R(q(t))$ oscillates. The Brent method is the default used by `optimize()` to find the minimum of a function of a single real number on an interval. When used alone, it frequently finds a local minimum that yields disappointing results. Much better results are obtained by dividing $[0, T]$ into subintervals of length $\frac{1}{2}$, using the Brent method on each of those, and choosing t^{TMR} as the time of the lowest of the resulting local minima. Let $q^{\mathrm{TMR}} = q(t^{\mathrm{TMR}})$, and choose as τ^{TMR} the permutation that sorts q^{TMR}.

The second is the *average integral* method (AI). From a trajectory, compute the average positions of the particles over the time interval $[0, T]$,

$$q^{\mathrm{AI}} = \frac{1}{T} \int_0^T q(s)\, \mathrm{d}s \tag{9}$$

Choose as τ^{AI} the permutation that sorts q^{AI}. Thus, if q_j spends most of the time far enough to the left of q_k, q_j^{AI} will be less than q_k^{AI}, so τ_j^{AI} will be less than τ_k^{AI}. This method has the advantage that no heuristics are required to compensate for the oscillations inherent in the trajectory.

3. Connection to the Toda lattice

When the items are clearly linearly ordered, ranking them is trivial, although the dynamics are still potentially interesting. The form of (5) suggests the following line of reasoning, motivated by the integrable systems literature.

Suppose $\alpha_{\mathrm{c}} = 0$ and $\alpha_{\mathrm{r}} = 1$ and consider comparison data in the form of a chain,

$$w_{jk} = \begin{cases} 1 & \text{if } k = j + 1 \text{ for } j, k \in \{1, \ldots, n\} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

then the Hamiltonian (8) is, up to a constant factor, the same as that of the traditional finite Toda lattice [10, 11],

$$H^{\mathrm{Toda}} = \frac{1}{4} \left(\frac{1}{2} \sum_j p_j^2 + \sum_j \mathrm{e}^{q_j - q_{j+1}} \right) \tag{11}$$

This is a well studied integrable system. Its long-term behavior is known: All momentum variables converge to constants, and the particle positions are asymptotic to linear functions of time. The particles scatter in the obvious order [12, 13].

3.1. Flaschka's formulation

With (8) in mind, it is tempting to try to generalize Toda dynamics to more interesting ranking problems. An obvious starting place is to change variables to Flaschka's formulation [14, 15], and generalize it to include weights. Using notation that will be necessary for further generalizations, Flaschka's change of variables is to define

$$\begin{aligned} b_j &= -\frac{1}{2}p_j \\ a_{jk} &= \begin{cases} \frac{1}{2}\mathrm{e}^{\frac{1}{2}(q_j - q_k)} & \text{if } k = j+1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

from which one forms the matrix B, which has the b_j's along the diagonal and zeros elsewhere, and the upper triangular matrix A with entries a_{jk}. Define the Lax pair

$$M = A - A^{\mathsf{T}} = \begin{pmatrix} 0 & a_{12} & 0 & \cdots \\ -a_{12} & 0 & a_{23} & \cdots \\ 0 & -a_{23} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{13}$$

and

$$L = A + A^{\mathsf{T}} + B = \begin{pmatrix} b_1 & a_{12} & 0 & \cdots \\ a_{12} & b_2 & a_{23} & \cdots \\ 0 & a_{23} & b_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{14}$$

The equations of motion change to isospectral flow

$$\dot{L} = [M, L] = ML - LM \tag{15}$$

under which the eigenvalues of L are conserved quantities, that is, they are constant with respect to time t. The Toda Hamiltonian (11) satisfies $H^{\mathrm{Toda}} = X_2$, defined as

$$X_2 = \frac{1}{2}\operatorname{tr} L^2 \tag{16}$$

which works out to be

$$X_2 = \frac{1}{2}\sum_j b_j^2 + \sum_{j,k} a_{jk}^2 \tag{17}$$

As it is a function of the eigenvalues of L, it is conserved. The a's converge to 0 as $t \to \infty$, and the b_j's converge to a sequence of the eigenvalues of L,

$b_1(\infty) > b_2(\infty) > \cdots$. The dynamics (15) effectively perform a continuous-time QR factorization of the initial condition matrix [16, 17, 18, 11]. Thus, the momentum variables converge to a sequence $q_1(\infty) < q_2(\infty) < \cdots$, again reflecting the fact that the particles scatter in the obvious order.

3.2. Generalization to data consistent with transitivity

A straightforward way to incorporate weighted comparisons is to abandon the particle variables q_j and p_j, and formulate dynamics directly in terms of vertex variables b_j and edge variables a_{jk}. Let us first consider the case where the data is consistent with transitivity and all comparison weights are strictly positive. Without loss of generality, we may topologically sort the items and renumber them so that the weight matrix satisfies

$$\begin{aligned} w_{jk} &> 0 \text{ if } j < k \\ w_{jk} &= 0 \text{ otherwise} \end{aligned} \tag{18}$$

Now define the following generalizations of A, B, L, and M,

$$\begin{aligned} A &= \begin{pmatrix} 0 & \sqrt{w_{12}}a_{12} & \sqrt{w_{13}}a_{13} & \cdots \\ 0 & 0 & \sqrt{w_{23}}a_{23} & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \\ B &= \begin{pmatrix} b_1 & 0 & \cdots \\ 0 & b_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \\ M &= A - A^{\mathsf{T}} \\ L &= A + A^{\mathsf{T}} + B \end{aligned} \tag{19}$$

Under isospectral flow (15), X_2 is still conserved, as it is a function of the eigenvalues of L, and it is invariant under renumbering of the items. It resembles (8), although it is no longer corresponds to a Hamiltonian as in (11) because the dynamics of a_{jk} can no longer be interpreted directly in terms of position variables q_j and q_k; setting $a_{jk} = \frac{1}{2}\exp\left(\frac{1}{2}(q_j - q_k)\right)$ imposes too many constraints on $q_1, \ldots, q_n$. Other information potentially related to ranking does seem to be available, in that the vertex variables b_j again converge to an ordered sequence $b_1(\infty) > b_2(\infty) > \cdots$ that, at first glance, may reflect something about how to order the items. Were it possible to interpret $b_j = -\frac{1}{2}p_j$, this asymptotic behavior would indicate that the momentum variables converge to an increasing sequence of constants, again resulting in positions that are asymptotically linear functions of time, and in the expected order.

3.3. General comparison data

To generalize to comparison data that is not consistent with transitivity, suppose more generally that

$$\begin{aligned} w_{jk} &> 0 \text{ if } j \neq k \\ w_{jk} &= 0 \text{ if } j = k \end{aligned} \tag{20}$$

Aiming to arrive at a conserved quantity of the form (11), define

$$\begin{aligned} A &= \begin{pmatrix} 0 & \sqrt{w_{12}}a_{12} - i\sqrt{w_{21}}a_{21} & \sqrt{w_{13}}a_{13} - i\sqrt{w_{31}}a_{31} & \cdots \\ 0 & 0 & \sqrt{w_{23}}a_{23} - i\sqrt{w_{32}}a_{32} & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \\ B &= \begin{pmatrix} b_1 & 0 & \cdots \\ 0 & b_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \\ M &= A - A^* \\ L &= A + A^* + B \end{aligned} \tag{21}$$

Assuming the a's and b's remain real, X_2 is still conserved and invariant under renumbering of the items. Thus, the formulation (21) does seem to be the correct generalization of the Toda lattice to a ranking problem. Now that L and M have complex entries, $L^* = L$ and $M^* = -M$, and both sides of (15) are self-adjoint. The diagonal elements of (15) have no imaginary component and yield a single real-valued differential equation for each b_j. The equations in the upper triangle of (15) suffice to determine the a's; those in the lower triangle are redundant. Each equation in the upper triangle must be split into its real and imaginary components. Two real-valued differential equations result, one for a_{jk} and one for a_{kj}. The equation for $\dot{a}_{jk}$ involves division by $\sqrt{w_{jk}}$, which means that all off-diagonal elements of w must be positive. Thus, the comparison data may need to be modified before the dynamics can be formulated, by replacing zeros with some tiny positive number, for example.

Unfortunately, as $t \to \infty$, the a's all converge to 0 and the b's all converge to constants $b_1(\infty) > b_2(\infty) > \cdots$. As $t \to -\infty$, the same convergence occurs, except that the limits of the b's are in the reverse order. This happens because the aforementioned convergence of B to a diagonal matrix of eigenvalues happens in this generalization of the Toda lattice as well [16]. Specifically, define the complex-valued variable u_{jk} to be the (j, k) element of L,

$$u_{jk} = \sqrt{w_{jk}}a_{jk} - i\sqrt{w_{kj}}a_{kj} \tag{22}$$

Under isospectral flow, as $t \to \pm\infty$, it is known that the u's converge to 0 and the b's converge to the eigenvalues of L in increasing or decreasing order.

Thus, it seems impossible to retrieve any useful ranking information from Toda dynamics despite the resemblance of (8) and (11). So, for the rest of this article, we put (11) aside and return to (8).

4. Cycle dynamics

Failure of transitivity is an essential challenge for ranking. When W is not consistent with transitivity, there is a cycle in the data, and it is unclear how to rank the items involved in the cycle. Let us consider the dynamics under (8) in the case of a cycle of n items, that is, $w_{jk} = 1$ when $k - j \equiv 1 \pmod{n}$, and is zero otherwise. The equations of motion are

$$\begin{aligned} \dot{q}_j &= q_j \\ \dot{p}_j &= -\alpha_{\mathrm{r}}\gamma_{\mathrm{r}}\mathrm{e}^{\gamma_{\mathrm{r}}(q_j - q_{j+1})} + \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}\mathrm{e}^{\gamma_{\mathrm{r}}(q_{j-1} - q_j)} - \alpha_{\mathrm{c}}\gamma_{\mathrm{c}}\sinh\gamma_{\mathrm{c}}q_j \end{aligned} \tag{23}$$

In this subsection, vector indices are to be interpreted modulo n. There is a fixed point at $q = 0, p = 0$, and it comes as no surprise that it is a center. To confirm this, let us examine the Jacobian matrix, which has block structure.

$$J = \begin{pmatrix} 0 & I \\ C & 0 \end{pmatrix} \tag{24}$$

The lower left block C has entries

$$\begin{aligned} c_{jk} &= \frac{\partial \dot{p}_j}{\partial q_k} \\ &= \begin{cases} -\alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2\mathrm{e}^{\gamma_{\mathrm{r}}(q_{j-1} - q_j)} - \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2\mathrm{e}^{\gamma_{\mathrm{r}}(q_j - q_{j+1})} - \alpha_{\mathrm{c}}\gamma_{\mathrm{c}}^2\cosh\gamma_{\mathrm{c}}q_j & \text{if } j = k \\ \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2\mathrm{e}^{\gamma_{\mathrm{r}}(q_j - q_{j+1})} & \text{if } j \equiv k+1 \pmod{n} \\ \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2\mathrm{e}^{\gamma_{\mathrm{r}}(q_{j-1} - q_j)} & \text{if } j \equiv k-1 \pmod{n} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{25}$$

At the fixed point in question, C is a circulant matrix,

$$c_{jk}\big|_{q=0,p=0} = \begin{cases} -2\alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2 - \alpha_{\mathrm{c}}\gamma_{\mathrm{c}}^2 & \text{if } j = k \\ \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2 & \text{if } j \equiv k+1 \pmod{n} \\ \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2 & \text{if } j \equiv k-1 \pmod{n} \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Using theorem 3.2.1 of [19], the eigenvalues of C are

$$\begin{aligned} \lambda_k &= -2\alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2 - \alpha_{\mathrm{c}}\gamma_{\mathrm{c}}^2 + \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2\left(\mathrm{e}^{2\pi ik/n} + \mathrm{e}^{2\pi ik(n-1)/n}\right) \\ &= -2\alpha_{\mathrm{r}}\gamma_{\mathrm{r}}^2\left(1 - \cos\left(\frac{2\pi k}{n}\right)\right) - \alpha_{\mathrm{c}}\gamma_{\mathrm{c}}^2 \end{aligned} \tag{27}$$

all of which are negative. Returning to the Jacobian, note that two n-element vectors u and v can be stacked to take advantage of the block structure,

$$\begin{pmatrix} 0 & I \\ C & 0 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} v \\ Cu \end{pmatrix} \tag{28}$$

Thus, each eigenvalue λ of J satisfies

$$\begin{pmatrix} v \\ Cu \end{pmatrix} = \lambda \begin{pmatrix} u \\ v \end{pmatrix} \tag{29}$$

or

$$Cu = \lambda^2 u \tag{30}$$

Thus, the eigenvalues of J are the square-roots of the eigenvalues of C. At the fixed point $q = 0, p = 0$, the eigenvalues of C are all negative, so the eigenvalues of J are pure imaginary, which means the fixed point is a center.

The interpretation of this is that the particles have no preferred order. As the particles oscillate, energy flows between kinetic and potential, so some states have a lower ranking potential than others.

5. Low-dimensional test cases

Let us explore a few test cases of (8) in which there is clearly some intuition about what "ranking" should mean. These will establish that the dynamics do something reasonable.

The Hamiltonian (8) includes several parameters. Some trial and error suggests that we consider two general extremes, where the confinement potential is strong or weak compared to the ranking potential. To strongly confine the particles, the parameters for *narrow confinement* are

$$\begin{aligned} \alpha_{\mathrm{r}} &= 1.0 \\ \gamma_{\mathrm{r}} &= 0.5 \\ \alpha_{\mathrm{c}} &= 0.1 \\ \gamma_{\mathrm{c}} &= 1.0 \end{aligned} \tag{31}$$

To weakly confine the particles, the parameters for *wide confinement* are

$$\begin{aligned} \alpha_{\mathrm{r}} &= 1.0 \\ \gamma_{\mathrm{r}} &= 0.5 \\ \alpha_{\mathrm{c}} &= 0.01 \\ \gamma_{\mathrm{c}} &= 0.1 \end{aligned} \tag{32}$$

5.1. Short chain

Consider a chain of four items,

$$W^{\mathrm{CH}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{33}$$

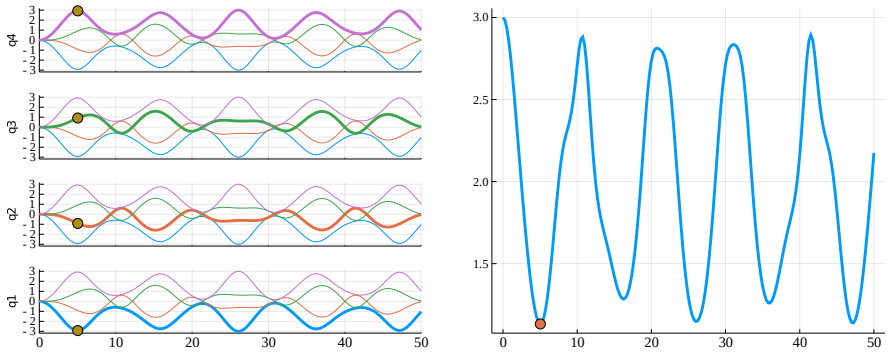

Figure 1: Left: Trajectories for a chain of four items. Right: Ranking potential. Narrow confinement parameters.

and its transitive closure, which is a perfect complete tournament with no upsets,

$$W^{\mathrm{CT}} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{34}$$

The dynamics are shown in figs. 1 to 4. Trajectories for these small systems are shown as a column of plots, each of which shows the positions of all the particles as a function of time, but with the trace of particle j drawn thicker in the j-th plot. Dots are drawn at $(\tau^{\mathrm{TMR}}, q_j(\tau^{\mathrm{TMR}}))$ in the j-th plot, and a dot is drawn at $(\tau^{\mathrm{TMR}}, R(q^{\mathrm{TMR}}))$ in the ranking potential plot. The initial conditions are $q = 0$ and $p = 0$. The particles stay roughly in order throughout, and both τ^{TMR} and τ^{AI} are $(1, 2, 3, 4)$ in narrow and wide confinement. Note that more than one particle can be in one location at the same time. Such meetings are not considered collisions and the particles are free to pass through each other. Meetings of particles cause a spike in the ranking potential, as the term $\mathrm{e}^{\gamma_r(q_j - q_k)}$ goes from being exponentially small to being quite large.

5.2. Cycles

Consider a short cycle of three items,

$$W^{\mathrm{CY}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{35}$$

As noted in section 4, the point $q = 0, p = 0$ is a center, so the system state must be started away from it to exhibit non-constant trajectories. The initial conditions in fig. 5 are

$$\begin{aligned} q(0) &= (-1.5, 0.5, 1.0) \\ p(0) &= (0, 0, 0) \end{aligned} \tag{36}$$

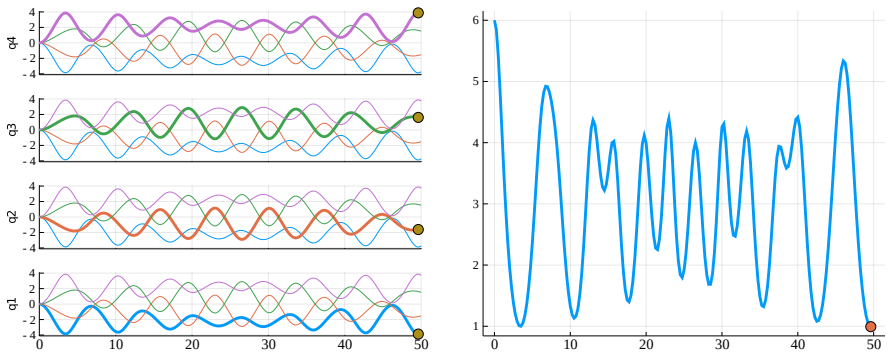

Figure 2: Left: Trajectories for a perfect complete tournament of four items. Right: Ranking potential. Narrow confinement parameters.

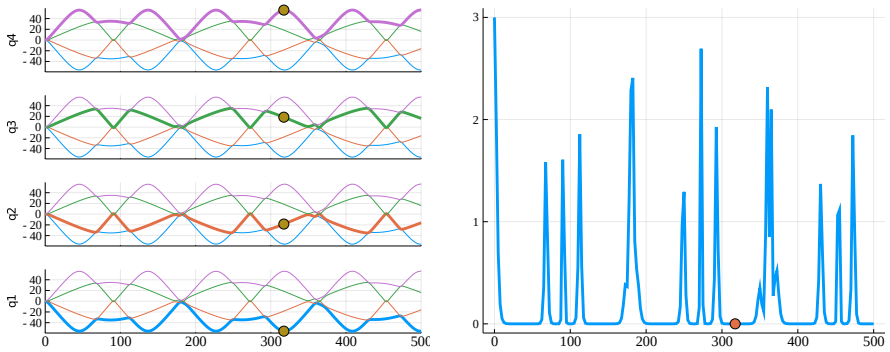

Figure 3: Left: Trajectories for a chain of four items. Right: Ranking potential. Wide confinement parameters.

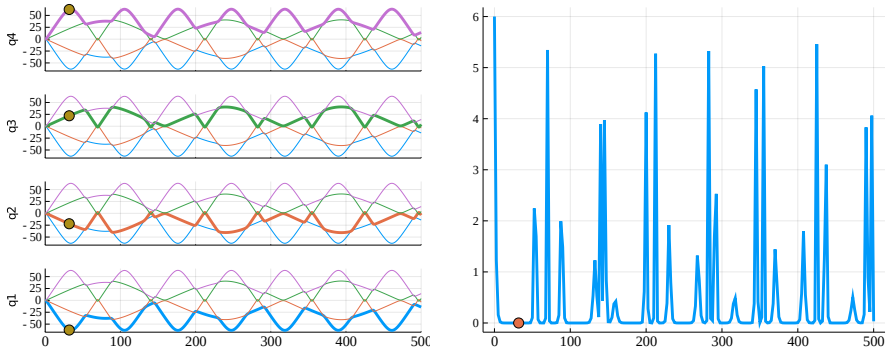

Figure 4: Left: Trajectories for a perfect complete tournament of four items. Right: Ranking potential. Wide confinement parameters.

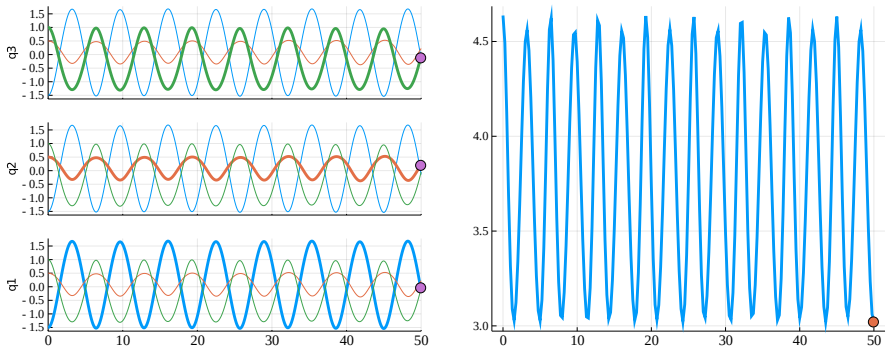

Figure 5: Left: Trajectories for cycle of three items. Right: Ranking potential. Narrow confinement parameters.

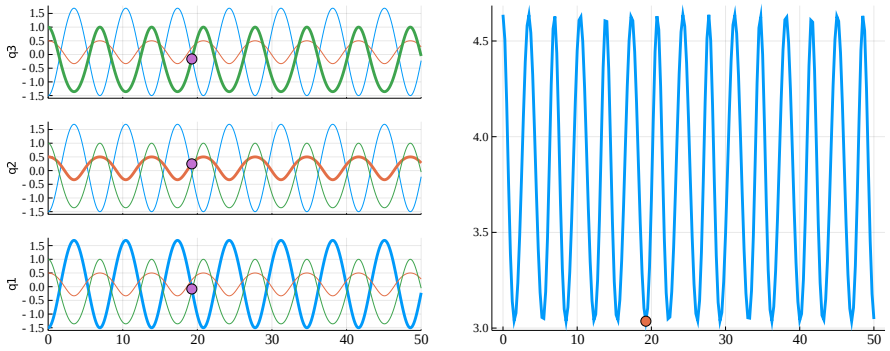

Figure 6: Left: Trajectories for cycle of three items. Right: Ranking potential. Wide confinement parameters.

It is possible to compute τ^{TMR} and τ^{AI} for this cycle, but they are essentially meaningless. Note that the ranking potential never goes lower than $3 = R(0)$. In comparison, for the chain (33) and complete tournament (34), the ranking potential drops to around 1 under narrow confinement, and nearly to 0 under wide confinement. It is also important to notice the range over which the particles move. Referring back to figs. 3 and 4, the particles in the chain and tournament spread out over intervals on the scale of -50 to 50, whereas the particles in the cycle are seen in fig. 6 to spread out only from -1.5 to 1.5. The greater the distance between q_j and q_k, the more confidence the dynamics have that j is weaker than k. Given items that form cycle, it is unsurprising that they remain minimally separated.

For a more interesting system, suppose that there are three items $(1, 2, 3)$ that form a cycle, plus an additional item 4 that is known to be significantly

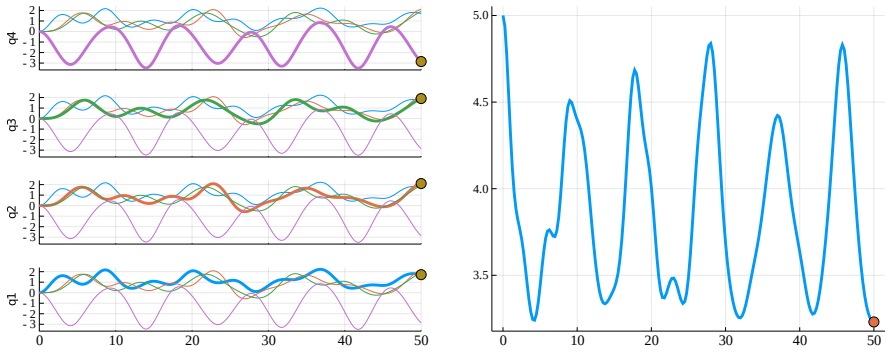

Figure 7: Left: Trajectories for cycle of three items, and one weaker item. Right: Ranking potential. Narrow confinement parameters.

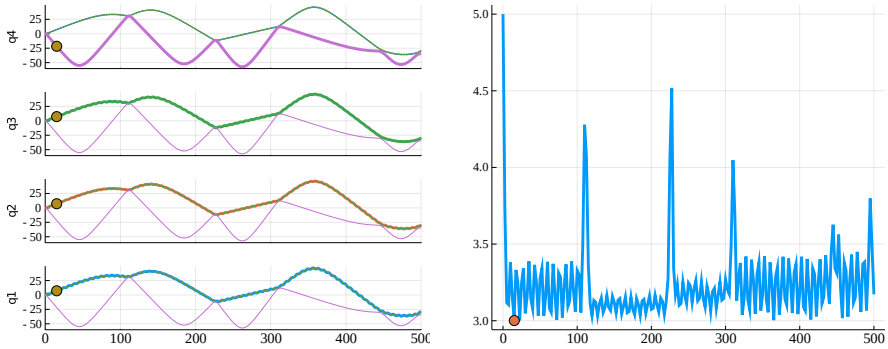

Figure 8: Left: Trajectories for cycle of three items, and one weaker item. Right: Ranking potential. Wide confinement parameters.

weaker than 1.

$$W^{\mathrm{CP}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix} \tag{37}$$

Since the data is not just a cycle, it is possible to use the initial condition $q = 0, p = 0$. As shown in figs. 7 and 8, the weak item generally satisfies $q_4 < q_1$, $q_4 < q_2$, $q_4 < q_3$. Even though it is not explicitly stated in the data (37) that item 4 is weaker than 2 and 3, the data suggests that 1, 2, and 3 are equally strong, and the dynamics pick up on this.

6. Random test data

In this section, we compare rankings derived from Hamiltonian dynamics to rankings derived from other methods, using a large set of synthetic ranking

problems. There are six algorithms to compare: trajectory minimum R with narrow (TMRN) and wide (TMRW) confinement; average integral with narrow (AIN) and wide (AIW) confinement; direct min R (DMR); and RankBoost (RB). Trajectory-based methods are run on the time interval $[0, 500]$. RankBoost runs for 800 iterations.

Given that we are interested in predicting future comparisons, let us posit that the items have some underlying order, and that the comparison data largely reflects that order, but with some possibility for upsets.

Suppose that comparisons are derived from a *strength* $s_j \in \mathbb{R}$ assigned to each item, $j = 1, 2, \ldots n$. In general, if $s_j < s_k$, that should mean that item k is stronger than item j and more likely to win when they are compared. To introduce the possibility of random upsets, the probability that j loses to k is

$$P(j,k) = \frac{1}{1 + \mathrm{e}^{\beta(s_j - s_k)}} \tag{38}$$

where $\beta > 0$ is a parameter. The larger s_k is compared to s_j, the greater the probability that k will defeat j. A random matrix W is built by considering each pair of items. With probability $P(j,k)$, $w_{jk} = 1$ and $w_{kj} = 0$; otherwise, $w_{jk} = 0$ and $w_{kj} = 1$. Note that if $s_j < s_k$, then $P(j,k) = \frac{1}{2}$ when $\beta = 0$, and $P(j,k) \to 1$ as $\beta \to \infty$. The larger β is, the more likely each random comparison is to agree with the underlying strengths.

The tournament score (3) provides a means of quantifying how compatible an ordering τ is with comparison data W. Even though τ^{TMR} and τ^{AI} are not specifically designed to maximize $\mathrm{TS}(\tau; W)$, its values are still informative.

6.1. Data from linearly ordered strengths

Let us consider the case when the items are linearly ordered and evenly spaced by setting $s_j = j$. Using $n = 20$, four sets of 1000 random samples of W were created, one for each value of β in $\{0.25, 0.5, 1.0, 2.0\}$. Each item is compared once with each other item, so each W matrix has 190 entries of 1 and the rest are zeros. Thus, the maximum possible tournament score is 190.

6.2. Analysis of tournament scores

Since there are six algorithms to compare, there are $\binom{6}{2} = 15$ pairwise comparisons to consider. For each pair of algorithms A and B, we compute the difference in tournament score for each sample of W, and compute a 95% confidence interval for the mean of those differences using the t-test. The results are shown in table 1.

Overall, the algorithms behave comparably. Most of the intervals in table 1 do not include 0, which means that it is likely that one algorithm generally outperforms the other, but the actual difference is relatively small. There are few general statements to be made about which algorithm performed best. The average integral method with narrow confinement (AIN) is a bit better than the others except for a few cases where TMRN and TMRW outperform it. The average integral method with wide confinement (AIW) is a bit weak overall. Interestingly, direct minimization of R (DMR) does not perform especially well.

	TMRN	TMRW	AIN	AIW	DMR
$\beta = 0.25$					
TMRW	$(-.02, .19)$				
AIN	$(.55, .79)$	$(.46, .70)$			
AIW	$(1.74, 2.04)$	$(1.66, 1.96)$	$(1.12, 1.33)$		
DMR	$(.82, 1.05)$	$(.73, .97)$	$(.19, .34)$	$(-1.06, -.86)$	
RB	$(.80, 1.03)$	$(.71, .95)$	$(.17, .33)$	$(-1.07, -.88)$	$(-.04, .01)$
$\beta = 0.5$					
TMRW	$(.15, .38)$				
AIN	$(-.59, -.44)$	$(-.90, -.66)$			
AIW	$(.70, .91)$	$(.41, .67)$	$(1.21, 1.42)$		
DMR	$(.64, .80)$	$(.32, .58)$	$(1.14, 1.32)$	$(-.15, -.02)$	
RB	$(.45, .62)$	$(.14, .40)$	$(.96, 1.14)$	$(-.34, -.20)$	$(-.22, -.14)$
$\beta = 1.0$					
TMRW	$(-.34, -.21)$				
AIN	$(-.93, -.79)$	$(-.67, -.49)$			
AIW	$(-.17, -.09)$	$(.08, .22)$	$(.66, .80)$		
DMR	$(.00, .08)$	$(.25, .39)$	$(.83, .97)$	$(.12, .21)$	
RB	$(-.06, .00)$	$(.18, .32)$	$(.76, .90)$	$(.06, .14)$	$(-.11, -.03)$
$\beta = 2.0$					
TMRW	$(-.02, .00)$				
AIN	$(-.15, -.08)$	$(-.14, -.07)$			
AIW	$(-.06, -.02)$	$(-.05, -.00)$	$(.05, .11)$		
DMR	$(-.01, .02)$	$(-.00, .03)$	$(.09, .16)$	$(.02, .07)$	
RB	$(-.01, .00)$	$(-.01, .02)$	$(.08, .15)$	$(.01, .05)$	$(-.03, .01)$

Table 1: 95% confidence intervals for differences in tournament scores, that is, the result from the algorithm in the column label minus the result from the algorithm in the row label. Positive numbers indicate that the column-labeled algorithm performed better.

	TMRN	TMRW	AIN	AIW	DMR
$\beta = 0.25$					
TMRW	$(-.17, .06)$				
AIN	$(-.23, .03)$	$(-.18, .10)$			
AIW	$(-.84, -.52)$	$(-.79, -.46)$	$(-.69, -.48)$		
DMR	$(-.34, -.08)$	$(-.29, -.01)$	$(-.18, -.04)$	$(.38, .57)$	
RB	$(-.36, -.10)$	$(-.31, -.04)$	$(-.20, -.06)$	$(.36, .55)$	$(-.05, .01)$
$\beta = 0.5$					
TMRW	$(-.59, -.34)$				
AIN	$(-.17, -.01)$	$(.25, .51)$			
AIW	$(-.74, -.53)$	$(-.30, -.03)$	$(-.65, -.44)$		
DMR	$(-.60, -.42)$	$(-.17, .10)$	$(-.51, -.32)$	$(.06, .20)$	
RB	$(-.52, -.35)$	$(-.10, .17)$	$(-.44, -.25)$	$(.12, .28)$	$(.02, .12)$
$\beta = 1.0$					
TMRW	$(-.21, -.07)$				
AIN	$(-.73, -.54)$	$(-.60, -.39)$			
AIW	$(-.74, -.57)$	$(-.61, -.42)$	$(-.11, .07)$		
DMR	$(-.82, -.65)$	$(-.69, -.50)$	$(-.20, -.01)$	$(-.16, .00)$	
RB	$(-.18, -.06)$	$(-.06, .09)$	$(.41, .61)$	$(.45, .61)$	$(.53, .69)$
$\beta = 2.0$					
TMRW	$(-.07, .00)$				
AIN	$(-.46, -.34)$	$(-.43, -.31)$			
AIW	$(-.44, -.32)$	$(-.40, -.29)$	$(-.03, .08)$		
DMR	$(-.22, -.12)$	$(-.19, -.09)$	$(.17, .29)$	$(.15, .27)$	
RB	$(.02, .07)$	$(.05, .11)$	$(.38, .51)$	$(.37, .48)$	$(.17, .26)$

Table 2: 95% confidence intervals for differences in inversion count, that is, the result from the algorithm in the column label minus the result from the algorithm in the row label. Negative numbers indicate that the column-labeled algorithm performed better.

6.3. Analysis of inversion counts

To measure the suitability of an ordering τ for making predictions, a reasonable statistic for this batch of test problems is the *inversion count* $\mathrm{IC}(\tau)$, which is how many swaps are needed to bubble sort the inferred ordering to the correct ordering of $1, 2, \ldots n$.

For each pair of algorithms A and B, we compute the difference in IC for each sample of W, and compute a 95% confidence interval for the mean of those differences using the t-test. The results are shown in table 2.

Again, the algorithms give fairly similar results. The actual differences are small. The average integral method with narrow confinement (AIN) generally does well, but TMRW and TMRN outperform it in most circumstances.

It is unclear why AIN works as well as it does. A possible explanation is that trajectories spend lots of time orbiting around centers surrounded by a large basin of stability. The average position vector q^{AI} will be close to that center.

	TMRN	TMRW	AIN	AIW
$\beta = 0.25$				
TMRW	$(.13, .25)$			
AIN	$(-1.13, -.90)$	$(-1.34, -1.06)$		
AIW	$(.96, 1.17)$	$(.79, .95)$	$(1.91, 2.25)$	
DMR	$(1.85, 2.02)$	$(1.69, 1.81)$	$(2.79, 3.12)$	$(.84, .92)$
$\beta = 0.5$				
TMRW	$(4.65, 5.36)$			
AIN	$(-13.97, -13.43)$	$(-19.28, -18.12)$		
AIW	$(5.89, 6.59)$	$(1.12, 1.35)$	$(19.37, 20.49)$	
DMR	$(7.23, 7.86)$	$(2.45, 2.63)$	$(20.70, 21.78)$	$(1.25, 1.37)$
$\beta = 1.0$				
TMRW	$(27.91, 28.79)$			
AIN	$(-22.87, -22.71)$	$(-51.65, -50.63)$		
AIW	$(27.77, 28.62)$	$(-.18, -.12)$	$(50.49, 51.48)$	
DMR	$(28.28, 29.11)$	$(.31, .38)$	$(51.00, 51.97)$	$(.48, .52)$
$\beta = 2.0$				
TMRW	$(39.22, 39.50)$			
AIN	$(-23.92, -23.88)$	$(-63.41, -63.12)$		
AIW	$(38.74, 39.00)$	$(-.50, -.48)$	$(62.63, 62.91)$	
DMR	$(39.40, 39.68)$	$(.17, .18)$	$(63.29, 63.58)$	$(.66, .68)$

Table 3: 95% confidence intervals for differences in ranking potential R, that is, the result from the algorithm in the column label minus the result from the algorithm in the row label. Negative numbers indicate that the column-labeled algorithm found position vectors q at which R is lower.

It is surprising that DMR consistently under-performs compared to TMRN. It seems that DMR is overfitting the data. Confidence intervals for the differences between the ranking potentials $R(q^{\mathrm{TMR}})$, $R(q^{\mathrm{AI}})$, and $R(q^{\mathrm{DMR}})$ are shown in table 3, with both narrow and wide confinement. Note that $R(q^{\mathrm{DMR}})$ is consistently less than the others. The minimization algorithm used for DMR is clearly working as intended, but the resulting ordering yields slightly higher inversion counts than TMRN. No confinement potential is included when directly minimizing R, which means no regularization is imposed. Apparently the numerical method finds values of q that take advantage of quirks in the data to reduce R, but doing so does not improve the predictive power of the resulting ordering. Minimizing the full potential including confinement produces results similar to TMR and AI, so regularization is clearly the distinguishing force.

7. Hamiltonian Monte Carlo methods

Hamiltonian dynamics can be used as the basis for a Markov-chain Monte-Carlo integration method [6, 7]. Although integration is not the problem of

interest here, it is worth considering whether the underlying Markov chain sampler may be used as a source of rankings.

Let π be a probability density over $\mathbb{R}^N$ expressed as

$$\pi(q) \propto \mathrm{e}^{-V(q)} \tag{39}$$

for a potential function V. The Markov chain is designed to produce samples distributed according to π. Starting with a position vector $q \in \mathbb{R}^N$, a momentum vector $p \in \mathbb{R}^N$ is chosen at random according to a specially designed distribution

$$\pi\left(p \mid q\right) \propto p^{\mathsf{T}} A(q) p \tag{40}$$

where the matrix $A(q)$ is associated with a Riemannian metric. The system state (q, p) is evolved according to the Hamiltonian

$$\begin{aligned} H &= -\ln \pi(q, p) + C \\ &= -\ln \pi\left(p \mid q\right) - \ln \pi(q) + C \\ &= p^{\mathsf{T}} A(q) p + V(q) \end{aligned} \tag{41}$$

The added constant C comes from the normalizing constants in (39) and (40), and is irrelevant to the dynamics. After some time, the momentum is discarded and replaced with a random sample of $\pi\left(p \mid q\right)$ using the value of q at that time. Values of q are emitted every so often and serve as nearly independent samples of $\pi(q)$.

For ranking, let us use the ranking and confinement potentials,

$$V(q) = \alpha_{\mathrm{r}} R(q) + \alpha_{\mathrm{c}} C(q) \tag{42}$$

and generate sample values of q. Rather than use them to estimate an integral, we treat them as sources of orderings. Samples should be concentrated around areas where V is minimized. Let q^{HMC} be the sample for which $R(q)$ is the least, and let τ^{HMC} be the permutation that sorts its elements, which should be a reasonable ordering of the underlying items. For each problem, $10,000$ samples of the Markov chain are considered. The algorithm is implemented using the `DynamicHMC.jl` package [20], which is based on [6].

Before discussing the numerical experiment, it is worth examining a phenomenon that adversely affects the performance of the Markov chain. For a picture of this effect, see fig. 9, which shows histograms derived from $10,000$ samples of the Hamiltonian Markov chain. The comparison matrix W is for a perfect complete tournament, $w_{jk} = 1$ if $j < k$ and 0 otherwise. This is the easiest possible ranking problem, yet none of the samples generates the correct order, and only one has an inversion count of 1. The Markov chain appears to have trouble getting into the part of its state space near the optimal position vectors.

The Markov chain seems to perform better if the particles have more room

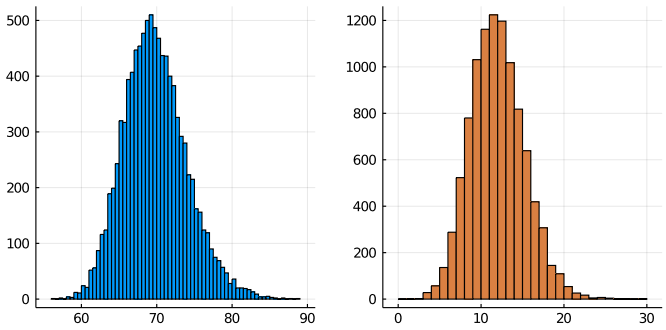

Figure 9: Histogram of ranking potentials (left) and inversion counts (right) derived from 10,000 samples of a Hamiltonian Markov chain using a perfect complete tournament matrix and wide confinement.

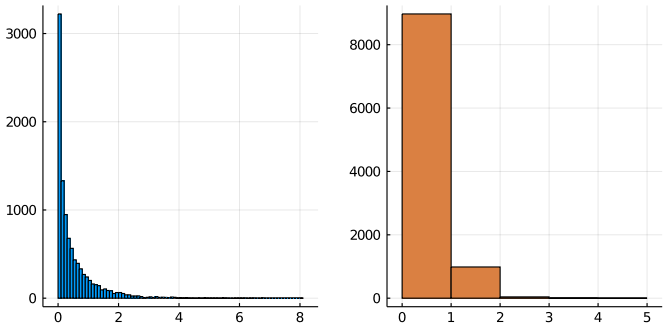

Figure 10: Histogram of ranking potentials (left) and inversion counts (right) derived from 10,000 samples of a Hamiltonian Markov chain using a perfect complete tournament matrix and extra wide confinement.

to spread out. Using *extra-wide confinement* by setting

$$\begin{aligned} \alpha_{\mathrm{r}} &= 1.0 \\ \gamma_{\mathrm{r}} &= 0.5 \\ \alpha_{\mathrm{c}} &= 0.01 \\ \gamma_{\mathrm{c}} &= 0.001 \end{aligned} \tag{43}$$

the distributions of the ranking potential and inversion count improve significantly for the perfect complete tournament matrix, as shown by the histograms in fig. 10.

With that in mind, let us test the Hamiltonian Markov chain with wide confinement (HMCW) and with extra-wide confinement (HMCXW) on the synthetic test problems from section 6. A table comparing the difference in inversion counts between HMCW and HMCXW and some of the other algorithms is shown in table 4. The data shows that HMCW results in slightly inferior orderings.

	TMRW	AIW	DMR	RB	HMCW
$\beta = 0.25$					
AIW	$(-.79, -.46)$				
DMR	$(-.29, -.01)$	$(.38, .57)$			
RB	$(-.31, -.04)$	$(.36, .55)$	$(-.05, .01)$		
HMCW	$(-1.39, -.99)$	$(-.76, -.37)$	$(-1.22, -.85)$	$(-1.20, -.84)$	
HMCXW	$(-1.38, -.96)$	$(-.73, -.36)$	$(-1.19, -.84)$	$(-1.17, -.82)$	$(-.20, .24)$
$\beta = 0.5$					
AIW	$(-.30, -.03)$				
DMR	$(-.17, .10)$	$(.06, .20)$			
RB	$(-.10, .17)$	$(.12, .28)$	$(.02, .12)$		
HMCW	$(-.73, -.38)$	$(-.53, -.25)$	$(-.65, -.38)$	$(-.73, -.45)$	
HMCXW	$(-.69, -.37)$	$(-.50, -.23)$	$(-.62, -.36)$	$(-.70, -.43)$	$(-.13, .19)$
$\beta = 1.0$					
AIW	$(-.61, -.42)$				
DMR	$(-.69, -.50)$	$(-.16, .00)$			
RB	$(-.06, .09)$	$(.45, .61)$	$(.53, .69)$		
HMCW	$(-1.15, -.93)$	$(-.63, -.42)$	$(-.55, -.34)$	$(-1.16, -.95)$	
HMCXW	$(-.93, -.71)$	$(-.40, -.21)$	$(-.32, -.13)$	$(-.94, -.74)$	$(.11, .33)$
$\beta = 2.0$					
AIW	$(-.40, -.29)$				
DMR	$(-.19, -.09)$	$(.15, .27)$			
RB	$(.05, .11)$	$(.37, .48)$	$(.17, .26)$		
HMCW	$(-.61, -.46)$	$(-.25, -.13)$	$(-.47, -.32)$	$(-.68, -.54)$	
HMCXW	$(-.57, -.44)$	$(-.22, -.10)$	$(-.43, -.30)$	$(-.65, -.52)$	$(-.03, .09)$

Table 4: 95% confidence intervals for differences in inversion count, that is, the result from the algorithm in the column label minus the result from the algorithm in the row label. Negative numbers indicate that the column-labeled algorithm performed better.

The differences between $R(q^{\mathrm{HMC}})$ and $R(q^{\mathrm{TMR}})$, $R(q^{\mathrm{AI}})$, and $R(q^{\mathrm{DMR}})$ reveal that HMCW is not emitting samples of q with especially low R, as shown in table 5. There is some evidence that HMCXW outperforms HMCW. It clearly finds position vectors at lower R, however, the improvements to inversion count are small.

It is worth mentioning that these Markov chain samplers are extremely time consuming, far more so than any of the other algorithms, which makes them relatively impractical as ranking algorithms. However, they produce a lot of information in addition to an ordering, discussion of which is beyond the scope of this article. Furthermore, the calculations in this section show that there is some interesting behavior hiding in their dynamics. Further study would be appropriate.

8. Spread, confidence, and rankability

The ranking algorithms under consideration assign positions to the items as well as ordering them. The distances between particles potentially contain

	TMRW	AIW	DMR	HMCW
$\beta = 0.25$				
AIW	$(.79, .95)$			
DMR	$(1.69, 1.81)$	$(.84, .92)$		
HMCW	$(-.09, .05)$	$(-.93, -.85)$	$(-1.79, -1.75)$	
HMCXW	$(-.10, .03)$	$(-.95, -.87)$	$(-1.80, -1.77)$	$(-.04, .01)$
$\beta = 0.5$				
AIW	$(1.12, 1.35)$			
DMR	$(2.45, 2.63)$	$(1.25, 1.37)$		
HMCW	$(.84, 1.03)$	$(-.36, -.24)$	$(-1.63, -1.59)$	
HMCXW	$(1.04, 1.23)$	$(-.15, -.04)$	$(-1.43, -1.38)$	$(.18, .23)$
$\beta = 1.0$				
AIW	$(-.18, -.12)$			
DMR	$(.31, .38)$	$(.48, .52)$		
HMCW	$(-2.07, -1.94)$	$(-1.90, -1.80)$	$(-2.39, -2.31)$	
HMCXW	$(.00, .05)$	$(.16, .20)$	$(-.34, -.30)$	$(1.97, 2.09)$
$\beta = 2.0$				
AIW	$(-.50, -.48)$			
DMR	$(.17, .18)$	$(.66, .68)$		
HMCW	$(-3.60, -3.54)$	$(-3.10, -3.05)$	$(-3.78, -3.72)$	
HMCXW	$(.17, .17)$	$(.65, .67)$	$(-.01, -.01)$	$(3.71, 3.77)$

Table 5: 95% confidence intervals for differences in ranking potential, that is, the result from the algorithm in the column label minus the result from the algorithm in the row label. Negative numbers indicate that the column-labeled algorithm found position vectors at lower R.

confidence information. The difference $q_k - q_j$ should be large when the data suggests that item k is distinctly stronger than item j, and it should be close to zero when the items appear to be of similar strengths. Overall, if the data indicates that the items are strongly ordered, there should be a fairly large *spread*, meaning the distance between the left-most particle and the right-most. A relatively large spread means that the particles spread out along a line, and the result of the ranking process is a meaningful ordering of the original items. Thus, they are highly rankable. A relatively small spread means that the particles remain clumped, and although each algorithm outputs a linear ordering, it may not be meaningful. The items are inherently unrankable in that case.

To test this, let us consider again the samples generated in section 6.1. That data will be called the *linear* set since the items were linearly ordered with equally spaced strengths. For comparison, additional sets of 1000 random sample problems were synthesized. In the first, called *two-groups*, items 1 through 10 are of strength 0 and items 11 through 20 are of strength 1. In the second, called *unordered*, all the items are of equal strength. Confidence intervals for the mean spreads for all of these data sets are shown in table 6.

Spreads calculated from the different algorithms are on different scales and not directly comparable. The trajectory-based methods, TMRN and AIN, yield a narrower spread because of the confinement potential. The spread for HM-CXW can be quite large since the confinement potential grows very slowly for q_j's below 1000. The spread for DMR can potentially be infinite because there is no confinement potential, but numerical limitations keep it from growing too much. The spread for RankBoost can likewise grow without bound as the number of iterations is increased. Several trends are apparent. All the algorithms agree that, holding β constant, the linear problems yield the greatest spread by far and are therefore the most rankable. The two-groups problems are more rankable than the unordered, but not by much. For lower values of β, upsets in the random data are more likely, so it is unsurprising that the linear and two-groups problems yield distinctly lower spread when β is lower. The unordered problems have the same distribution regardless of β, so the spread is also the same.

9. Further test cases

9.1. LOLIB input-output matrices

For an additional batch of test cases, let us consider the LOLIB data set, described in [1]. This test suite has become somewhat difficult to locate online despite its popularity. Currently, the best place to download it is the Internet Archive [21]. Among other data, it includes a set of 49 input-output matrices based on economic data, along with best known tournament scores. Ordering of items based on these matrices is an economic tool for identifying how resources flow through large-scale supply chains, rather than a means of predicting future contests. Matrix entries are non-negative integers spanning many orders of magnitude. In some of these, there are indices j for which row j and column

	TMRN	AIN	RB
$\beta = 0.25$:			
Linear	$(6.52, 6.71)$	$(5.29, 5.35)$	$(3.99, 4.18)$
Two groups	$(1.88, 1.94)$	$(1.95, 2.01)$	$(.98, 1.01)$
Unordered	$(1.79, 1.84)$	$(1.85, 1.91)$	$(.93, .96)$
$\beta = 0.5$:			
Linear	$(11.17, 11.27)$	$(6.62, 6.67)$	$(10.59, 11.08)$
Two groups	$(2.10, 2.16)$	$(2.19, 2.25)$	$(1.10, 1.14)$
Unordered	$(1.79, 1.84)$	$(1.85, 1.91)$	$(.93, .96)$
$\beta = 1.0$:			
Linear	$(11.92, 11.94)$	$(7.16, 7.19)$	$(27.92, 28.97)$
Two groups	$(2.70, 2.76)$	$(2.85, 2.92)$	$(1.45, 1.49)$
Unordered	$(1.79, 1.84)$	$(1.85, 1.91)$	$(.93, .96)$
$\beta = 2.0$:			
Linear	$(12.03, 12.04)$	$(7.39, 7.40)$	$(50.40, 51.70)$
Two groups	$(4.09, 4.19)$	$(4.12, 4.19)$	$(2.27, 2.34)$
Unordered	$(1.79, 1.84)$	$(1.85, 1.91)$	$(.93, .96)$

	DMR	HMCXW
$\beta = 0.25$:		
Linear	$(15.73, 18.98)$	$(116.46, 156.72)$
Two groups	$(1.97, 2.02)$	$(2.11, 2.17)$
Unordered	$(1.86, 1.92)$	$(2.00, 2.06)$
$\beta = 0.5$:		
Linear	$(133.09, 153.02)$	$(670.24, 733.84)$
Two groups	$(2.21, 2.27)$	$(2.34, 2.41)$
Unordered	$(1.86, 1.92)$	$(2.02, 2.08)$
$\beta = 1.0$:		
Linear	$(1058.48, 1106.56)$	$(1278.70, 1296.81)$
Two groups	$(2.90, 2.98)$	$(3.01, 3.10)$
Unordered	$(1.86, 1.92)$	$(2.03, 2.09)$
$\beta = 2.0$:		
Linear	$(1358.04, 1369.51)$	$(1354.09, 1361.27)$
Two groups	$(4.60, 5.04)$	$(4.78, 10.29)$
Unordered	$(1.86, 1.92)$	$(2.02, 2.08)$

Table 6: 95% confidence intervals for the mean of the particle spread

TMRN	AIN	DMR	RB	HMCXW
(.89, .91)	(.84, .93)	(.93, .94)	(.93, .94)	(.93, .94)

Table 7: 95% confidence intervals for ratios of tournament scores generated by four algorithms to the best known values, on the IO matrices from LOLIB.

j are all zeros, indicating no data at all for that commodity. Such row-column pairs are removed from the matrices before the ranking algorithms are applied, as they have no meaningful impact on the outcome.

Confidence intervals for the mean of the ratio of each algorithm's tournament score to the best known are shown in table 7. In general, all of these algorithms perform respectably compared to the best known results. DMR, HMCXW, and RankBoost yield distinctly higher tournament scores than the trajectory-based methods. The performance of the Markov chain method is impressive given its lackluster performance on the synthetic test data in section 6.

Unfortunately, there is no ground truth to use as a basis for comparison for this data, as there was with the synthetic data. Very high tournament scores may sometimes indicate overfitting, and there is no way to detect or rule out that possibility.

9.2. Southern Conference Basketball

In [2], data from the 2008-9 season of basketball games within the Southern Conference (SoCon) of Division I of the National Collegiate Athletic Association (NCAA) was used as a test case. In this section, we will apply the algorithms under consideration to this data. The cost matrix, derived from point spreads of basketball games, is reproduced in table 8. Each entry c_{jk} indicates how strongly the data supports putting team j before team k, so it serves as the W matrix. The method described in that article, called MVR, seeks orderings that minimize the number of something called *hillside violations* using an integer program. Four optimal orderings were found by MVR, differing in the orders of two pairs of teams (3 & 4, and 11 & 1) that were effectively tied. See table 9.

When RankBoost and all of the Hamiltonian and energy-based methods were applied to this data, they all arrived at the same ordering. It is the second line in table 9, and differs by a total of 4 swaps from the most similar ordering found by MVR, which is the first line. The two orderings essentially disagree on the placement of teams 8 and 9. A scatter plot of (j, q_j) for the average integral method with narrow confinement (AIN) is show in fig. 11. Positions produced by the other algorithms (including RankBoost) are very similar. Neither of the ties identified by MVR are evident in the scatter plot.

Overall it seems that the ranking potential R is measuring something a bit different from the count of hillside violations, but they largely agree.

9.3. NFL 2018-19

In this section, we apply ranking algorithms to game data from the 2018-19 season of the National Football League (NFL). There are 32 teams. During the

0	14	15	15	17	11	9	8	12	12	12	11
9	0	12	13	16	11	9	5	11	10	9	12
9	12	0	11	16	7	7	9	10	5	9	10
8	11	13	0	16	9	9	8	11	9	10	11
7	8	8	8	0	6	3	6	9	5	6	8
12	13	17	15	18	0	13	12	13	8	12	15
15	15	17	15	21	11	0	12	17	12	14	15
16	19	14	16	18	12	12	0	14	11	15	13
12	13	14	12	15	11	7	10	0	9	11	13
12	13	19	15	18	15	11	13	15	0	13	16
12	14	15	14	18	12	10	9	12	11	0	14
13	12	14	13	16	8	9	10	11	8	10	0

Table 8: Cost matrix C from [2].

MVR	8	7	10	6	11	1	12	2	9	4	3	5
RB, etc.	7	10	8	6	11	1	9	12	2	4	3	5

Table 9: Orderings of SoCon basketball teams produced by MVR (top) and RankBoost and the other algorithms (bottom) using the cost matrix of table 8 as W.

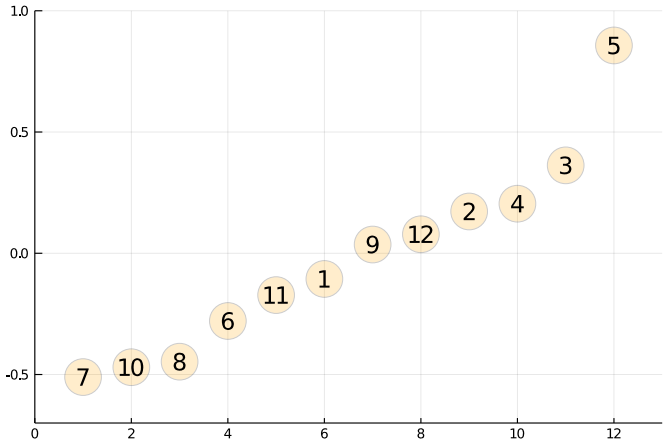

Figure 11: Positions of particles associated to the 12 SoCon basketball teams by AIN.

regular season, each team plays 16 games. In the post season, 11 teams play each other in a single elimination tournament called the *playoffs.* For this test, the W matrix consists of point spreads summed over all games between each pair of teams,

$$w_{jk} = \text{points scored by } k \text{ against } j - \text{points scored by } j \text{ against } k \tag{44}$$

A scatter plot of (j, q_j) for AIN is shown in fig. 12. Each point is labeled with the associated team. The name of each team is abbreviated to two or three letters derived from the city or region in which they are based.

Once an algorithm yields an ordering based on W, the particle positions are used to make predictions about which team will win in each playoff game. The results are shown in table 10. Column four is the point spread, that is, the number of points x_j scored by team j minus the number of points x_k scored by team k, in that game. Columns five through eight are $q_j - q_k$ divided by the spread of the q's, as computed by four algorithms. (TMRW and AIW are not shown, but they produced similar results.)

The winner of a game is correctly predicted when $x_j - x_k$ and $q_j - q_k$ have the same sign. Every algorithm correctly predicted the winner of games 5, 6, and 8. AIN, RB, and HMCXW correctly predicted the winner of game 3, but with low confidence, evidenced by the small magnitudes of $q_j - q_k$. All the other predictions were wrong. None of the algorithms was particularly successful.

Some comments are in order. All the teams in the playoffs are very good. Differences between them are small and upsets are likely. Game 2 was decided by two points. Game 4 was decided by a single point. Games 9 and 10 were the division championships, and both went into overtime because the score was tied when time ran out. Both included controversial rulings by the referees that might have affected the outcome. Overall, if a relatively small number of plays had worked out differently, the outcomes of the playoffs might have been very different.

In games 7, 10, and 11, all the algorithms predicted that New England (NE) would lose. They under-performed during the regular season, losing to some weak teams early on. They improved later in the season, did well in the playoffs, and ultimately won the league championship game, known as the Superbowl (game 11). The algorithms are only given a comparison matrix W, and do not have access to temporal information.

It is disappointing that these algorithms correctly predicted only 3 or 4 out of 11 playoff games. Such an outcome results from the overall difficulty of making predictions when upsets are likely, and the fact that a team can improve or decline substantially over the course of a single season.

10. Conclusion

Practically speaking, RankBoost is a great general-purpose ranking algorithm. It is efficient and flexible and performs well on all the problems considered in this article. However, the purpose of this article is not just to produce

	j	k	$x_j - x_k$	TMRN	AIN	RB	HMCXW
1	HOU	IND	−14	.025	.142	.135	.137
2	DAL	SEA	2	−.201	−.195	−.198	−.240
3	BAL	LAC	−6	.008	−.074	−.058	−.036
4	CHI	PHI	−1	.336	.201	.204	.237
5	KC	IND	18	.177	.322	.322	.329
6	LA	DAL	8	.344	.277	.256	.244
7	NE	LAC	13	−.011	−.199	−.180	−.158
8	NO	PHI	6	.369	.284	.285	.278
9	NO	LA	−3	.046	.022	.041	.035
10	KC	NE	−6	.144	.321	.332	.333
11	LA	NE	−10	.124	.245	.224	.210

Table 10: NFL playoffs with predictions.

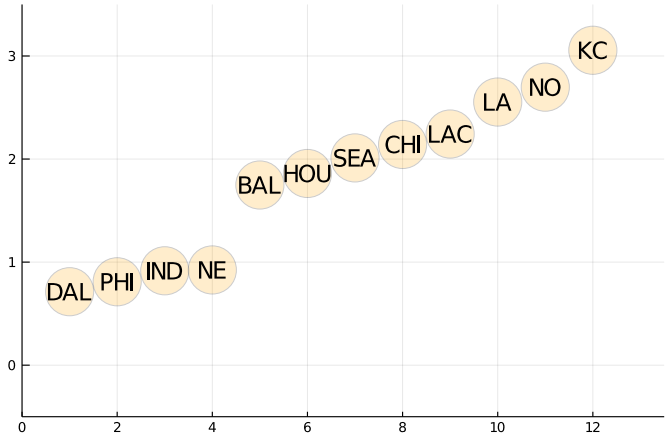

Figure 12: Positions of particles associated with the 11 NFL teams in the playoffs, using AIN.

ranking algorithms, but to begin developing connections between dynamical systems theory and data science, and to formulate interesting problems in dynamics. Hamiltonian methods are worth investigating, despite complexities such as the need for numerical integration of a system of differential equations.

The low-dimensional and highly symmetric cases discussed in section 2 confirm that the dynamics given by the Hamiltonian (8) do effectively rank the items in question when possible. When the items form a cycle, no ranking is possible, and the corresponding particles oscillate around a center.

The comparison data in section 6.1 shows that TMR, AI, and DMR are competitive with the widely used RankBoost algorithm. The confinement potential is important, as without it, ranking by directly minimizing the ranking potential alone yields lower quality predictions.

The LOLIB test cases examined in section 9.1 demonstrate that all of algorithms proposed in this article perform reasonably well on the input-output matrices. DMR, HMCXW, and RankBoost achieved higher tournament scores than the trajectory based methods.

The basketball example shows that expressing a ranking algorithm in terms of different objectives can have a noticeable impact on the outcome.

The format of the data is also important. This article has focused on rankings derived only from pairwise comparisons. RankBoost can be applied when other information is available, but it is unclear how the other algorithms could be adapted. Importantly, they are unable to accommodate data covering a time span during which items change in strength. The NFL example shows how the W matrix may be misleading in this way, which inevitably results in poor predictions.

An additional feature of these energy-based approaches to ranking is that they generate particle positions q as well as an ordering τ. Similarly, RankBoost generates a function from items to real numbers indicating their order. The overall spread of the item positions indicates a measure of confidence in the ordering, which is not available from the integer programming approach. It can also be interpreted as a measure of how rankable the data set is. Additionally, the individual particle positions can yield a measure of confidence in the ordering of pairs of items or even clusters. Further investigation is necessary to develop a rigorous understanding of how to apply these observations.

References

[1] R. Martí, G. Reinelt, A. Duarte, A benchmark library and a comparison of heuristic methods for the linear ordering problem, Computational Optimization and Applications 51 (3) (2012) 1297–1317. doi:10.1007/s10589-010-9384-9.

[2] K. E. Pedings, A. N. Langville, Y. Yamamoto, A minimum violations ranking method, Optimization and Engineering 13 (2) (2012) 349–370. doi:10.1007/s11081-011-9135-5.

[3] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, The Journal of Machine Learning Research 4 (2003) 933–969.

[4] H. Connamacher, N. Pancha, R. Liu, S. Ray, Rankboost+: An improvement to Rankboost, Machine Learning 109 (1) (2020) 51–78. doi:10.1007/s10994-019-05826-x.

[5] C. Rudin, C. Cortes, M. Mohri, R. E. Schapire, Margin-based ranking meets boosting in the middle, in: P. Auer, R. Meir (Eds.), Learning Theory, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005, pp. 63–78. doi:10.1007/11503415_5.

[6] M. Betancourt, A conceptual introduction to Hamiltonian Monte Carlo (Jan. 2017).

[7] M. Betancourt, S. Byrne, S. Livingstone, M. Girolami, The geometric foundations of Hamiltonian Monte Carlo, Bernoulli 23 (4A) (2017) 2257–2298. doi:10.3150/16-BEJ810.

[8] P. Anderson, T. Chartier, A. Langville, The rankability of data, SIAM Journal on Mathematics of Data Science 1 (1) (2019) 121–143. doi:10.1137/18M1183595.

[9] P. K. Mogensen, A. N. Riseth, Optim: A mathematical optimization package for Julia, Journal of Open Source Software 3 (24) (2018) 615. doi:10.21105/joss.00615.

[10] G. Teschl, Almost everything you always wanted to know about the Toda equation, Jahresbericht der Deutschen Mathematiker-Vereinigung (DMV) 103 (4) (2001) 149–162.

[11] N. M. Ercolani, H. Flaschka, S. Singer, The geometry of the full Kostant-Toda lattice, in: O. Babelon, Y. Kosmann-Schwarzbach, P. Cartier (Eds.), Integrable Systems: The Verdier Memorial Conference Actes Du Colloque International de Luminy, Progress in Mathematics, Birkhäuser Boston, Boston, MA, 1993, pp. 181–225. doi:10.1007/978-1-4612-0315-5.

[12] B. Kostant, The solution to a generalized Toda lattice and representation theory, Advances in Mathematics 34 (3) (1979) 195–338. doi:10.1016/0001-8708(79)90057-4.

[13] D. Bilman, I. Nenciu, On the evolution of scattering data under perturbations of the Toda lattice, Physica D: Nonlinear Phenomena 330 (2016) 1–16. doi:10.1016/j.physd.2016.03.017.

[14] H. Flaschka, The Toda lattice. I. Existence of integrals, Physical Review. B. Condensed Matter. Third Series 9 (4) (1974) 1924–1925. doi:10.1103/PhysRevB.9.1924.

[15] H. Flaschka, On the Toda Lattice. II. Inverse-scattering solution, Progress of Theoretical Physics 51 (3) (1974) 703–716. doi:10.1143/PTP.51.703.

[16] M. Chu, The generalized Toda flow, the QR algorithm and the center manifold theory, SIAM Journal on Algebraic Discrete Methods 5 (2) (1984) 187–201. doi:10.1137/0605020.

[17] P. Deift, L. C. Li, C. Tomei, Matrix factorizations and integrable systems, Communications on Pure and Applied Mathematics 42 (4) (1989) 443–521. doi:10.1002/cpa.3160420405.

[18] P. Deift, T. Nanda, C. Tomei, Ordinary differential equations and the symmetric eigenvalue problem, SIAM Journal on Numerical Analysis 20 (1) (1983) 1–22.

[19] P. J. Davis, Circulant Matrices, Pure and Applied Mathematics, Wiley, New York, 1979.

[20] T. Papp, DynamicHMC.jl.

[21] Linear Ordering Problem, https://web.archive.org/web/20180702045352/http://www.optsicom.es/lolib/ (Jul. 2018).

Appendix A. RankBoost

The RankBoost algorithm [3] is a rank aggregation process based on boosting. There are several variations. This section gives the details of the version used in these experiments.

The input is a finite list X of items to be ranked, and a non-negative weight matrix W. To simplify the notation, the items will be denoted by integer indices, so $X = \{1, \ldots, n\}$. RankBoost seeks a function $h\colon X \to \mathbb{R}$ that minimizes the empirical risk function

$$B = \sum_{(j,k)\in \operatorname{support} W} \mathrm{e}^{-(h(k)-h(j))} \tag{A.1}$$

where the support of W is the pairs for which $w_{jk} > 0$. Note that B is similar in structure to R.

The function h is built incrementally. At step t, a weak ranking algorithm is called to construct a function h_t and a scalar α_t based on W and a probability distribution D_t over pairs of items. The intention is that D_t represents the relative importance of the possible comparisons, and as the algorithm proceeds, it assigns greater importance to certain comparisons to better resolve the ordering of those items. Once h_t and α_t have been chosen, the algorithm proceeds to the next iteration using the updated distribution

$$D_{t+1}(j,k) = \left(\frac{1}{Z_t}\right)\left(D_t(j,k)\mathrm{e}^{\alpha_t(h_t(k)-h_t(j))}\right) \tag{A.2}$$

where Z_t is a normalizing constant that ensures $\sum_{j,k} D_{t+1}(j,k) = 1$. After a specified number of steps, the final ranking function is

$$h(j) = \sum_t \alpha_t h_t(j) \tag{A.3}$$

and the final ordering τ^{RB} is given by sorting the elements of X in increasing order of h.

The initial distribution D_0 is W divided by a normalization constant. For the results in this article, 800 iterations were performed.

There are many possible weak ranking algorithms. For these experiments, α_t is chosen and h_t is constructed as follows to produce low values of

$$\tilde{Z}_t = \sum_{j,k} D_t(j,k)\mathrm{e}^{\alpha_t(h_t(k)-h_t(j))} \tag{A.4}$$

Specifically, each column of $W^{\mathsf{T}} - W$ is interpreted as a *feature*, that is,

$$f_k(j) = \begin{cases} w_{kj} - w_{jk} & \text{if it is non-zero} \\ \perp & \text{otherwise} \end{cases} \tag{A.5}$$

and f_k is understood as giving an ordering of the items based only on comparisons with item k. The use of a special "nothing" result $\perp$ for $f_k(j)$ indicates that no direct comparison between items j and k has been given, or that the data includes perfectly contradictory information. A feature index k, a threshold $\theta \in \mathbb{R}$, and a default value $\delta \in \{0,1\}$ are chosen, to give the weak ranking function

$$h_t(j) = \begin{cases} 1 & \text{if } f_k(j) > \theta \\ 0 & \text{if } f_k(j) \le \theta \\ \delta & \text{if } f_k(j) = \perp \end{cases} \tag{A.6}$$

They are chosen to maximize $|r|$ where

$$r = \sum_{j,k} D(j,k)\,(h_t(k) - h_t(j)) \tag{A.7}$$

following the algorithm in figure 3 of [3]. The choice of α_t is

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right) \tag{A.8}$$

using the Third Method from section 3.2 of [3].