Finding Hamiltonian cycles using an interior point method

Michael Haythorpe*

Abstract

We present an unconstrained logarithmic barrier algorithm to solve the Hamiltonian cycle problem. The interior point method described here takes advantage of significant improvements in efficiency gained by the use of a special LU decomposition. Some initial results and an example are presented to illustrate the potential effectiveness of this method.

1. Introduction

The Hamiltonian cycle problem (HCP) is a difficult problem in graph theory. Consider a graph Γ, containing nodes $V(\Gamma)$ and arcs $E(\Gamma)$ without self-loops. If the number of nodes $|V(\Gamma)| = N$, a Hamiltonian cycle (HC) is defined as a simple cycle of length N. The problem is to determine whether a graph possesses at least one Hamiltonian cycle. Despite this simple definition, the HCP is NP-complete [9]. The HCP is in fact a special case of the travelling salesman problem. The HCP is, generally, easier to solve for dense graphs (for the intuitive reason that a greedy heuristic is more likely to find an HC in such a graph) so we will consider sparse graphs for the remainder of this paper. While we will consider directed edges (arcs) in this paper, it is assumed that any graph is symmetric, so that an arc (i, j) exists in the graph if and only if arc (j, i) also exists.

There have been a variety of algorithms designed to solve the HCP. Some of these have been designed for particular subsets of graphs, such as in Eppstein [3] where only cubic graphs are considered. An excellent survey of algorithms designed to solve the HCP is included in Vandegriend [12]. Most approaches to the HCP have been either to employ graph theory techniques (usually on a subset of graphs, so that properties of the subset may be exploited) to develop a heuristic that finds an HC, or to model the HCP as a discrete optimisation problem. We note that the first algorithm to exploit both a Markov decision process embedding of the HCP

Received 2 March 2009; accepted 6 April 2010.
*School of Mathematics and Statistics, University of South Australia, Mawson Lakes, SA 5095. E-mail: Michael.Haythorpe@unisa.edu.au

Joint work with my thesis supervisors Professor Jerzy Filar (UniSA) and Professor Walter Murray (Stanford). I gratefully acknowledge support from the Australian Research Council Discovery Grant No. DP0666632 and the Australian Research Council Linkage International Grant No. LX0881972.

Michael Haythorpe was a winner of the T.M. Cherry Prize for the best student presentation at the ANZIAM 2008 conference. This peer-reviewed technical paper is an invited contribution to the *Gazette*.

and an interior point method is due to Ejov *et al.* [2]. However, the algorithm in [2] works in a space of occupational measures, rather than in the policy space.

Our approach to solving the HCP has been to model the problem using Markov decision processes (MDPs), and to solve an associated continuous optimisation problem, where the variables in the program correspond to the probabilities of travelling from one node to another. These probabilities can be represented as entries in a probability transition matrix P. Our hope is that, by embedding the HCP in an MDP, we will be able to take advantage of the tools and techniques of Markov decision processes that are not available in graph theory. The fundamental matrix, an important matrix in the analysis of MDPs, motivated the following result.

Consider the set $\mathcal{DS}$ of doubly stochastic matrices induced by the graph Γ that is defined as

$$\mathcal{DS} = \{P \mid P\boldsymbol{e} = \boldsymbol{e},\, P^T\boldsymbol{e} = \boldsymbol{e},\, p_{ij} \geq 0,\, p_{ij} = 0 \text{ if } (i,j) \notin E(\Gamma)\},$$

where $\boldsymbol{e}$ is an N-dimensional vector containing unity in every entry, and consider the constrained optimisation problem

$$\min_{P\in\mathcal{DS}} -\det\left(I - P + \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T\right). \tag{1}$$

It was proved in [1] that if Γ is a Hamiltonian graph, the set of global maximisers for (1) is the set of P corresponding to Hamiltonian cycles in Γ, and that the objective function of (1) has optimal value $|V(\Gamma)| = N$. Furthermore, it was proved that if Γ is a non-Hamiltonian graph, then (1) has optimal value that is strictly less than N. As yet, no algorithms have been designed that attempt to solve (1). The present paper seeks to redress this, by attempting to obtain the optimal value using a logarithmic barrier algorithm, in the spirit of those discussed in [11]. For further discussion as to the merit of this formulation and the benefits of using the interior point method described in this paper to solve it, the reader is referred to [7].

Traditional logarithmic barrier algorithms attempt to solve an inequality constrained optimisation problem by essentially eliminating the inequality constraints. This is done by adding logarithmic terms to the objective function. By starting at an initial estimate that strictly satisfies the inequalities, the algorithms are designed so that all subsequent iterates strictly satisfy them as well. In our case, we have an additional reason for using this approach. Adding a logarithmic term, which is convex, results in the new objective being 'more' convex than the original. Indeed if the barrier parameters are sufficiently large the new objective is strictly convex and has a unique minimiser. Our approach requires a different strategy for modifying the barrier parameters than that traditionally used. In our case, we replace the $p_{ij} \geq 0$ constraints by appending

$$-\sum_{(i,j)\in\Gamma} \alpha_{ij} \ln p_{ij} \tag{2}$$

to $-\det(I-P+(1/N)\boldsymbol{e}\boldsymbol{e}^T)$ to form an auxiliary logarithmic barrier objective function. Define $A(P) = (I-P+(1/N)\boldsymbol{e}\boldsymbol{e}^T)$, and the auxiliary objective function $f(P)$

consisting of $-\det(A(P))$ augmented by the logarithmic barrier function (2) as

$$f(P) := -\det(A(P)) - \sum_{(i,j)\in\Gamma} \alpha_{ij} \ln p_{ij}.$$

Note that if P is in the interior of $\mathcal{DS}$, $p_{ij} \in (0,1)$ and hence (2) is well defined. Then, we attempt to solve the following optimisation problem using a logarithmic barrier algorithm:

$$\min_P f(P)$$

subject to

$$P\boldsymbol{e} = \boldsymbol{e}, P^T\boldsymbol{e} = \boldsymbol{e}, p_{ij} = 0 \text{ if } (i,j) \notin E(\Gamma).$$

The initial choice of α_{ij} must be of 'sensible' size to ensure that the global minimum is well inside the interior of the feasible region but not at the centre.

Once the iterates converge to the global minimiser of $f(P)$ for a particular selection of α values, we reduce the α values, and the global minimum changes accordingly. Although $-\det(A(P))$ is nonconvex, it is hoped that this sequence of global minimisers will converge towards an extreme point of $\mathcal{DS}$ that corresponds to a Hamiltonian cycle (if one exists), but this cannot be guaranteed. In [7] it was conjectured that no strictly interior local minima exist for (1), which would imply that the sequence of global minimisers will converge to a point on the boundary of the $\mathcal{DS}$ polytope.

2. Null space

In order to define the iterates to solve our transformed problem it is useful to use a basis for the null space of the linear equality constraints. Define the action space $\mathcal{A}(i) = \{j \mid (i,j) \in \Gamma\}$, for each node $i \in V(\Gamma)$. The equality constraints in the optimisation problem then take the form:

$$\sum_{j\in\mathcal{A}(i)} p_{ij} = 1, \text{for all } i \tag{3}$$

$$\sum_{j\in\mathcal{A}(i)} p_{ji} = 1, \text{for all } i. \tag{4}$$

We represent constraints (3)–(4) using a transportation-like coefficient matrix W, so that the above constraints can be rewritten $W\boldsymbol{p} = \boldsymbol{1}$. These constraints are always rank-deficient. Once enough rows are removed to make it full-rank, row and column swaps can be performed to transform the matrix into a form where finding a null space matrix Z is trivial. The null space matrix Z that we construct retains the sparsity inherent in the original graph, and only contains values of -1, 0 and 1. We can take advantage of this by using a sparse matrix multiplication algorithm to perform any multiplications. This algorithm will only have to perform additions and subtractions, making it extremely efficient and numerically stable.

The use of the null space allows us to drop the equality constraints (3) and (4). When the algorithm finds a descent direction $\boldsymbol{x}$, it can be projected into the null space by finding $\boldsymbol{d} = Z\boldsymbol{x}$. This will ensure that as long as the starting point is feasible, subsequent points will remain feasible.

3. Descent directions

At each step of the logarithmic barrier algorithm, we need to move to a new point P^k such that $f(P^k) < f(P^{k-1})$. To do this we use a modified-Newton method. Such a method requires solving at each iteration a system of linear equations and these are solved by a modified form of the conjugate-gradient algorithm. In addition there is a need to obtain not only descent directions but directions of negative curvature. These latter directions are found by a truncated form of the Lanczos algorithm [6].

The conjugate-gradient algorithm attempts to solve the second-order approximation

$$Z^T HZ\boldsymbol{x} = -Z^T\boldsymbol{g},$$

where $\boldsymbol{g}$ and H are the gradient and Hessian of $f(P)$, respectively. Then $Z^T\boldsymbol{g}$ and $Z^T HZ$ are the reduced gradient and reduced Hessian respectively. The direction $\boldsymbol{x}$ is then itself projected into the original space to give us a feasible descent direction

$$\boldsymbol{d}_1 = Z\boldsymbol{x},$$

that will ensure that P^k remains feasible while improving the objective function value.

The Lanczos algorithm is used to attempt to approximate the eigenvector $\boldsymbol{y}$ corresponding to the most negative eigenvalue of the reduced Hessian. The Lanczos algorithm requires the reduced gradient and reduced Hessian as input. We then project $\boldsymbol{y}$ into the null space to give us a direction of negative curvature

$$\boldsymbol{d}_2 = Z\boldsymbol{y}.$$

Ordinarily, only one of the conjugate-gradient or Lanczos algorithms would be required to find a direction to improve the objective function value. However, it is useful to have both because in certain circumstances either one may fail. In some graphs, the natural starting point is a saddle point, and the conjugate-gradient algorithm is unable to find any descent direction. In these instances, the Lanczos algorithm can be used to find a direction of negative curvature. In other situations the reduced Hessian H is positive-definite, in which case no directions of negative curvature exist, and only the conjugate-gradient algorithm is required to find a direction that improves the objective function value. When both directions exist a combination of the two directions, $\hat{\boldsymbol{d}} = \beta\boldsymbol{d}_1 + (1-\beta)\boldsymbol{d}_2$, $0 \leq \beta \leq 1$, is chosen in such a way as to give the maximum improvement to the objective function.

4. LU decomposition

At each iteration of the logarithmic barrier algorithm we need to find the gradient $\boldsymbol{g}$ and Hessian H. The objective function $f(P)$ contains logarithmic terms, for which the derivatives are simple to find, and the negative of the determinant of

$A(P)$. Define $\boldsymbol{g}^D$ and $\boldsymbol{g}^L$ as the gradient of the determinant function and the logarithmic terms respectively. Similarly, define H^D and H^L as the Hessian of the determinant function and the logarithmic terms respectively. The gradient of a determinant is a vector of cofactors, and the Hessian of a determinant is a matrix of cofactors of minors [10]. Recall from (2) that $f(P) = -\det(A(P)) - \sum_{ij} \alpha_{ij} \ln p_{ij}$. Define $A^{ij}(P)$ as the matrix $A(P)$ with row i and column j removed. The general gradient element of $\boldsymbol{g}^D$ is then of the form

$$g^D_{ij}(P) = \frac{\partial \det(A(P))}{\partial a_{ij}} \frac{da_{ij}}{dp_{ij}} = (-1)^{i+j+1} \det A^{ij}(P),$$

where a_{ij} denotes the (i,j)th entry of $A(P)$. Note that the gradient has two subscripts because each element of the gradient corresponds to the arc going from node i to node j. Similarly, we define $A^{[ij],[kl]}(P)$ as the matrix $A(P)$ with rows i and j and columns k and l removed. The general Hessian element of H^D is then of the form

$$H^D_{[ij],[kl]}(P) = (-1)^{i+j+k+l} \det A^{[ij],[kl]}(P).$$

In general, the most efficient way of finding the determinant of a matrix is to first perform an LU decomposition. Then, as L and U are triangular matrices, their determinants are a product of their diagonal elements. However, this would require us to find an LU decomposition for each element of the gradient and Hessian. Instead, we can use a single LU decomposition to find every element of the gradient and Hessian.

The way this is done is by finding an LU decomposition of $I - P$, which was shown in [8] to always exist for an irreducible P, which will always be the case for $P \in \mathrm{Int}(\mathcal{DS})$. We then factorise $I - P + (1/N)\boldsymbol{e}\boldsymbol{e}^T$ in the following way:

$$I - P + \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T = LU + \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T = L(I + \boldsymbol{v}\boldsymbol{w}^T)\bar{U}, \tag{5}$$

where vectors $\boldsymbol{v}$, $\boldsymbol{w}$ and the matrix $\bar{U}$ are defined, either directly, or implicitly, by the equations

$$L\boldsymbol{v} = \boldsymbol{e}, \tag{6}$$

$$\bar{U} = U + \boldsymbol{v}\boldsymbol{e}_N^T, \tag{7}$$

$$\bar{U}^T\boldsymbol{w} = \frac{1}{N}\boldsymbol{e} - \boldsymbol{e}_N. \tag{8}$$

Note that both $\boldsymbol{v}$ and $\boldsymbol{w}$ can be found extremely efficiently by solving the sparse linear sets of equations (6) and (8), already in reduced row echelon form.

It can then be proved that for $P \in \mathrm{Int}(\mathcal{DS})$, $\det(L)\det(I + \boldsymbol{v}\boldsymbol{w}^T) = 1$ and so from (5)

$$\det\left(I - P + \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T\right) = \det \bar{U}. \tag{9}$$

Furthermore since $\bar{U}$ is, by construction in (7), a triangular matrix, (9) is equivalent to

$$\det\left(I - P + \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T\right) = \prod_{k=1}^{N} \bar{u}_{kk}. \tag{10}$$

To find elements of the gradient $\boldsymbol{g}$, elementary matrices E_{ij} are created such that $\det\{E_{ij}(I - P + (1/N)\boldsymbol{e}\boldsymbol{e}^T)\} = c_{ij}$, where c_{ij} is the (i,j)th cofactor of $I - P + (1/N)\boldsymbol{e}\boldsymbol{e}^T$. Each element of the gradient is then expressed as

$$g_{ij} = \det(E_{ij}) \prod_{k=1}^{N} \bar{u}_{kk}.$$

After some transformations it can be shown that

$$g_{ij} = \prod_{k=1}^{N} \bar{u}_{kk}(\boldsymbol{a}_j^T(I - \boldsymbol{v}\boldsymbol{w}^T)\boldsymbol{b}_i), \tag{11}$$

where $\boldsymbol{b}_i$ and $\boldsymbol{a}_j$ are the unique solutions of $L\boldsymbol{b}_i = \boldsymbol{e}_i$ and $\bar{U}^T\boldsymbol{a}_j = \boldsymbol{e}_j$, respectively. (Details are supplied in [7].) Hence, each $\boldsymbol{a}_j$ and $\boldsymbol{b}_i$ can be found efficiently in advance. Then, we can also compute in advance expressions of the form

$$\boldsymbol{a}_j^T(I - \boldsymbol{v}\boldsymbol{w}^T)\boldsymbol{b}_i, \tag{12}$$

for all $i, j = 1, \ldots, N$.

Once each element of the gradient vector $\boldsymbol{g}$ is found, using similar arguments each element of the Hessian matrix H can be found to be of the form

$$H_{[ij],[kl]} = g_{kl}(\boldsymbol{a}_l^T(I - \boldsymbol{v}\boldsymbol{w}^T)\boldsymbol{b}_i) - g_{ij}(\boldsymbol{a}_l^T(I - \boldsymbol{v}\boldsymbol{w}^T)\boldsymbol{b}_k). \tag{13}$$

The right-hand side of (13) contains the difference of two terms. We note that each of these terms contains an element of the gradient matrix, and an expression of the form (12). These are both computed when the gradient is found, and therefore every element of the Hessian can be calculated using two scalar multiplications of numbers computed in advance.

The complexity of performing all these calculations is no more than the complexity of finding a single LU decomposition. Compared to the more standard method of finding a separate LU decomposition for each entry in the gradient vector and Hessian matrix, this is a large saving in computational time.

5. Rounding

Using the logarithmic barrier algorithm to find a solution known to exist on the boundary of a feasible region is problematic because it can take many iterations to obtain a point sufficiently close to the boundary. However, we can alleviate this problem because we know what form our solution must take. At any iteration we can round off the current P matrix and check whether it corresponds to a Hamiltonian cycle, and this can be done in linear time.

The rounding can be as crude as replacing any values bigger than 0.5 with 1 and any smaller than 0.5 with 0, or more sophisticated heuristics can be used. In [4] a linear program was introduced that will find P^H corresponding to a Hamiltonian cycle if $|P - P^H| \leq \delta$ for moderately sized δ. By use of this linear program the solution of the logarithmic barrier algorithm can be found without needing to converge extremely close to the boundary.

6. Results

The algorithm outlined above was implemented in MATLAB and tested on several sets of Hamiltonian graphs. The results of these tests are outlined in Table 1.

Table 1. Results obtained from solving sets of graphs.

Graph size	Number solved	Average iterations	Average run time (secs)
$N = 20$	48	20.42	1.55
$N = 40$	40	86.98	12.05
$N = 60$	30	198.72	54.77
$N = 80$	33	372.76	196.26

Each test set contains 50 randomly generated Hamiltonian graphs of the indicated size where each node has degree between 3 and 5. For each test set, we give the number of graphs (out of the 50 generated) in which our implementation of the interior point algorithm succeeds in finding a Hamiltonian cycle, the average number of iterations performed, and the average running time for each graph. Note that since this algorithm is implemented in MATLAB, the running times are not competitive when compared to other similar models implemented in a compiled language. However, we provide the running times here to demonstrate how they grow as N increases.

Example 1. We ran the interior point algorithm outlined above on a 14-node cubic graph, specifically the graph with the following adjacency matrix:

$$\begin{bmatrix}
0&1&1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&1&1&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&1&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&1&0&0&0&1&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&1&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&1&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1&1&1&0
\end{bmatrix}.$$

A Hamiltonian cycle was found after eight iterations. The probability assigned to each arc is displayed in Figures 1–5.

Note that at iteration 1, $P(\boldsymbol{x})$ assigned equal probabilities to all 42 arcs, but at iteration 8, one arc from each node contains most of the probability. The rounding process at the completion of iteration 8 assigns these arcs to a Hamiltonian cycle $1 \to 4 \to 6 \to 10 \to 12 \to 14 \to 13 \to 9 \to 7 \to 11 \to 8 \to 5 \to 3 \to 2 \to 1$.

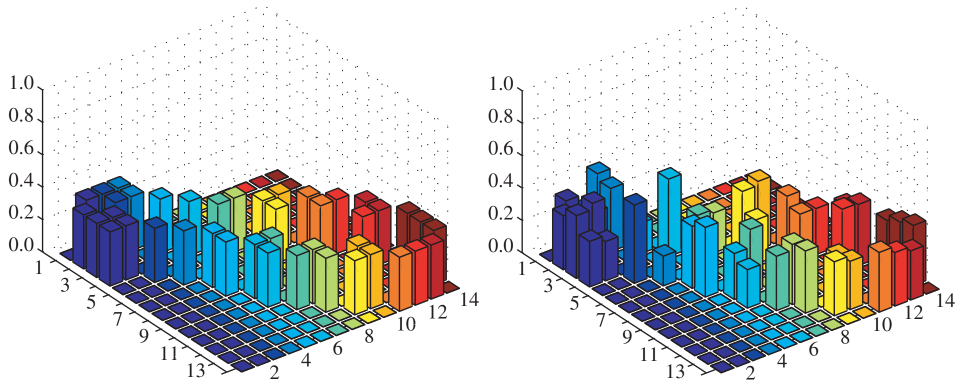

Figure 1. Iterations 1 and 2.

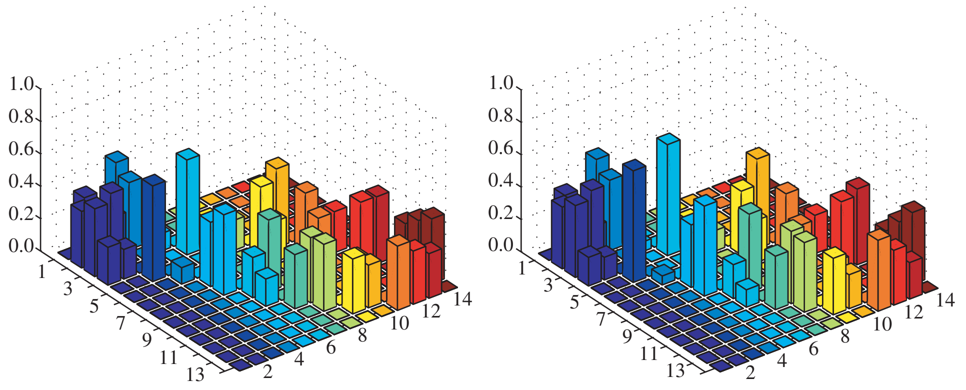

Figure 2. Iterations 3 and 4.

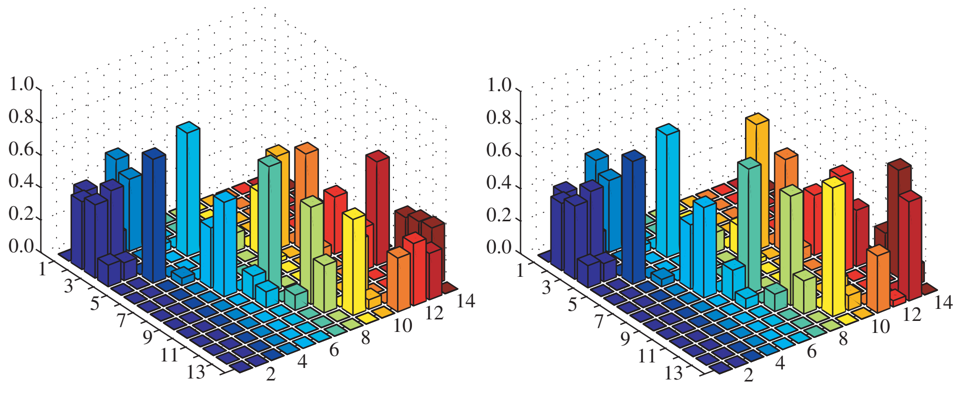

Figure 3. Iterations 5 and 6.

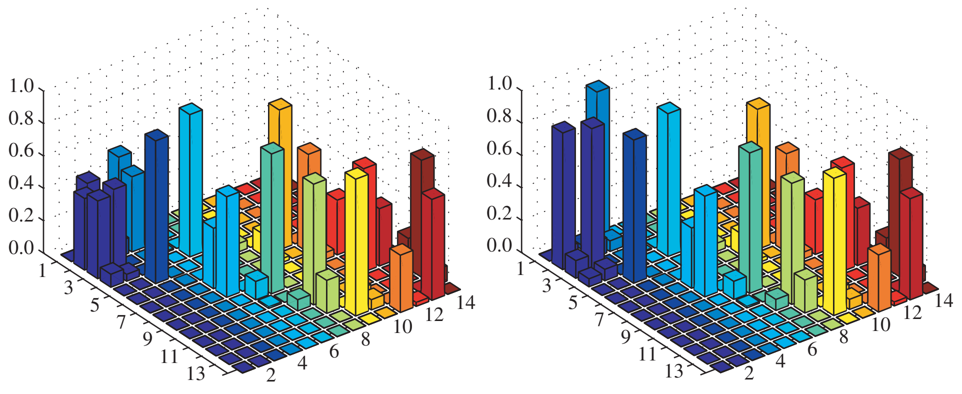

Figure 4. Iterations 7 and 8.

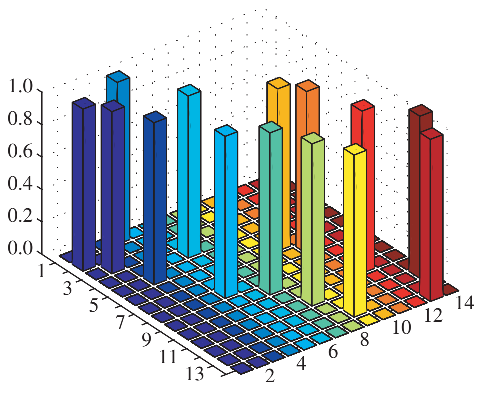

Figure 5. Final Hamiltonian cycle.

References

[1] Ejov, V., Filar, J.A., Murray, W. and Nguyen, G.T. (2009). Determinants and longest cycles of graphs. *SIAM Journal of Discrete Mathematics* **22,** 1215–1225.

[2] Ejov, V., Filar, J.A. and Gondzio, J. (2004). An interior point heuristic for the Hamiltonian cycle problem via Markov decision processes. *Journal of Global Optimization* **29,** 315–334.

[3] Eppstein, D. (2003). The traveling salesman problem for cubic graphs. In *Algorithms and Data Structures* (Lecture Notes Comput. Sci. **2748**), Springer, Berlin, pp. 307–318.

[4] Eshragh, A., Filar, J.A. and Haythorpe, M. (2009). A hybrid simulation-optimization algorithm for the Hamiltonian cycle problem. *Annals of Operations Research* (online 21 May 2009) DOI 10.1007/s10479-009-0565-9.

[5] Filar, J.A. (2007). Controlled Markov chains, graphs, and Hamiltonicity. *Foundations and Trends in Stochastic Systems* **1,** 77–162.

[6] Golub, G.H. and O'Leary, D.P. (1989). Some history of the conjugate gradient and Lanczos algorithms, 1948–1976. *SIAM Review* **31,** 50–102.

[7] Haythorpe, M. (2010). Markov Chain Based Algorithms for the Hamiltonian Cycle Problem. Ph.D. Thesis, University of South Australia.

[8] Heyman, D. (1995). A decomposition theorem for infinite stochastic matrices. *Journal of Applied Probability* **32,** 893–901.

[9] Karp, R.M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, eds R.E. Miller and J.W. Thatcher. Plenum, New York, pp. 85–103.

[10] May, K. (1965) Derivatives of determinants and other multilinear functions. *Mathematics Magazine* **38,** 307–308.

[11] Nocedal, J. and Wright, S.J. (1999). *Numerical Optimization.* Springer, New York.

[12] Vandegriend, B. (1998). Finding Hamiltonian cycles: algorithms, graphs and performance. Masters thesis, University of Alberta.

Michael Haythorpe is a research assistant in the School of Mathematics and Statistics at the University of South Australia, having recently received his doctorate from the same university. His areas of interest are in numerical optimisation, graph theory, and computational mathematics.