A graph theoretic approach to the brachistochrone problem

Hamzeh H. Roomany

Citation: Computers in Physics **4**, 303 (1990); doi: 10.1063/1.168370
View online: https://doi.org/10.1063/1.168370
View Table of Contents: https://aip.scitation.org/toc/cip/4/3
Published by the American Institute of Physics

A graph theoretic approach to the brachistochrone problem

Hamzeh H. Roomany
Department of Computer Science, KAAU, P. O. Box 4679, Jeddah 21412, Saudi Arabia

(Received 22 September 1989; accepted 14 December 1989)

The algorithm for finding the shortest path in a graph is applied to the brachistochrone problem of classical mechanics, discretized on a spatial lattice. The results are in agreement with those obtained by variational techniques within the error limits introduced by the lattice. The new approach provides more insight and promises to be more flexible than conventional methods.

INTRODUCTION

Two seemingly unrelated problems, one in classical physics and one in computer science, are shown to be closely related. The physics problem originates in classical mechanics, and is to determine the shape of the curve between two points such that a particle constrained to it and acted upon by gravity would traverse it in the least possible time. This problem was first solved by Johann Bernoulli in 1696 and is thought to be a milestone in the invention of the calculus of variations.[1] The computer science problem originates in graph theory and is to find the shortest path between two vertices in a graph. The problem has many applications in the areas of traffic planning and project management and there exist several algorithms for solving it,[2] of which Dijkstra's[3] is the most efficient.

In this article we solve the physics problem by using the algorithm of the graph theory problem mentioned above. To that end, we reformulate the physics problem in a form isomorphic to a graph. We apply a slightly modified version of Dijkstra's algorithm to the resulting graph to obtain the brachistochrone and the minimum transit time. The process is repeated for varying sizes of the lattice.

The two problems along with their solutions are stated in Secs. I and II. The reformulation of the physics problem in graph theory language is presented in Sec. III. Section IV describes the implementation and discusses the results. Concluding remarks are given in Sec. V.

I. THE BRACHISTOCHRONE PROBLEM

This problem seeks to find the frictionless path between two given points along which a particle, acted upon by a uniform force field and starting at rest from one end, would reach the other end in the least possible time. The solution, obtained via the Euler–Lagrange equation, is the segment of the cycloid that connects the two points and whose directrix passes through the starting point perpendicular to the force field.[4] If the coordinate system is chosen so that the origin is at the starting point and the y axis is in the direction of the field (as in Fig. 1), then the parametric equations of the cycloid are

$$x = c(\Theta - \sin\Theta),$$
$$y = c(1 - \cos\Theta), \qquad (1)$$

where the constant c is determined so that the cycloid passes through the ending point. The transit time T, computed using energy conservation and integrating along the cycloid, is given by

$$T = \Theta^*\sqrt{mc/F}\ , \qquad (2)$$

where m is the particle's mass, F is the field strength, and Θ^* is the value of Θ at the ending point. For definition (and without loss of generality), we consider the gravitational field (in which $F = mg$) and compute the transit time for three special points (Fig. 1): P1 at $(L,0)$, P2 at $(L,L/2)$, and P3 at $(0,L)$, where L is some length scale. We find $T(\mathrm{P1}) = 2.507\alpha$, $T(\mathrm{P2}) = 1.784\alpha$, and $T(\mathrm{P3}) = 1.414\alpha$, where $\alpha = \sqrt{(L/g)}$ is the time scale of the system. The point P3 was included for verification purposes only. For

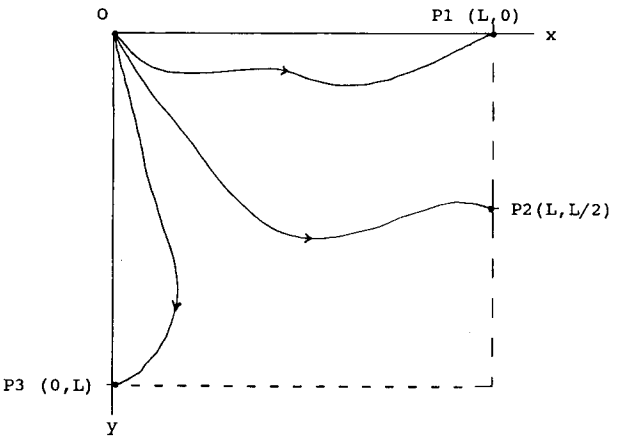

FIG. 1. Three examples of the brachistochrone problems: A particle starts at rest from the origin under the influence of its weight, oriented toward the positive y axis. What is the shape of the wire along which the particle can slide so as to reach P1 (or P2 or P3) in the least possible time?

it, the problem becomes trivial and the cycloid reduces to a straight line with a transit time derived from $y = \frac{1}{2}gt^2$.

II. THE SHORTEST PATH PROBLEM

This problem can be stated as follows: Given the distance matrix of a directed weighted graph $G(V,E)$, find the shortest path between two given vertices in it. If we label the vertices V as 1,2,3,...,N such that vertex 1 is the beginning of the path and vertex k is its end, then the problem is to find the path $P(1 \rightarrow k)$ such that the sum of the weights of its edges is minimum compared to all other paths linking 1 and k. An example of this problem is shown in Fig. 2. Dijkstra's algorithm solves this problem if the weights associated with the edges E are all positive. The algorithm runs in $O(N^2)$ time and generates $P(1 \rightarrow i)$ for all $i \in V$ (not just $1 \rightarrow k$) along with the minimum distance of each.

From a practical point of view, all one needs to do to use Dijkstra's algorithm is to prepare the distance matrix: An $N \times N$ matrix whose matrix element (i, j) is the weight assigned to the directed edge $i \rightarrow j$ (which can be ∞ if no edge in G connects vertices i and j). We mention, in passing, that the commonly used term "distance," in graph theory terminology, refers to the weights assigned to the edges and not necessarily to the distance between sites. It could be the cost, time, or length of the intersite transition.

In addition, the logic of the algorithm requires the allocation of two arrays of N elements each: One holds the paths and is of type *integer*, and one stores the minimum distances and is of the same type as the weights. Since the weights are transit times in our case (as we shall see in Sec. III), the second array is of type *real*. Hence, the program needs to allocate $6N$ to $10N$ bytes of memory, depending on how numbers are represented in the machine's memory.

III. REFORMULATION OF THE BRACHISTOCHRONE PROBLEM

The first step toward constructing a graph isomorphic to the brachistochrone problem is to reformulate it in a finite, rather than a continuum, form. We introduce a lattice to discretize space so that the position of the particle at any time is at one of the lattice sites. Let a be the lattice spacing, M the number of sites per side, and L the length of the side. We can see from this that $L = a(M - 1)$ and that the total number of sites in the lattice is $N = M \times M$. These lattice sites are the vertices of our graph. The particle moves from one site to the next by following one of the eight paths to the eight surrounding sites (horizontally, vertically, and diagonally). These paths along with their transit times are the edges and the weights of our graph, respectively. Graph theory algorithms specify a path using the vertex number terminology, whereas in mechanics we ordinarily use the (x,y) coordinates. With the aid of the lattice, these two terminologies become interchangeable. We choose the x-y axis as in Sec. I, adopting the dimensionless coordinates $X = x/a$, $Y = y/a$, and number the vertices row-wise (see Fig. 3). It is easy to see that given a vertex number k, its corresponding (x,y) coordinates are given by

$$X = (k - 1) \bmod M,$$
$$Y = (k - 1) \operatorname{div} M, \tag{3}$$

where mod and div refer to modulo and integer division, respectively. Similarly, given the x and y coordinates of one of the lattice sites, its corresponding vertex number is

$$k = 1 + X + MY. \tag{4}$$

Next we construct the distance matrix (some books refer to it as the cost adjacency matrix). To compute the matrix element (i, j) we first express the vertex numbers i and j as a pair of points (X_1, Y_1) and (X_2, Y_2), using Eqs. (3). If the second point is not one of the eight neighbors of the first or if such a path will take the particle beyond the boundaries of the lattice, we set this matrix element to ∞. (In the program, any number greater than 3 will do since, as was shown in Sec. I, all the transit times are less than 3α.) If, on the other hand, this path is possible, we compute its transit time using energy conservation and the initial condition that the particle started from the origin with no initial velocity (i.e., with total energy = 0). Hence at any lattice site we have

$$\frac{1}{2}mv^2 - mgy = 0.$$

From this equation, which allows us to compute the speed of the particle at any lattice site regardless of how it got there, we can derive the following results:

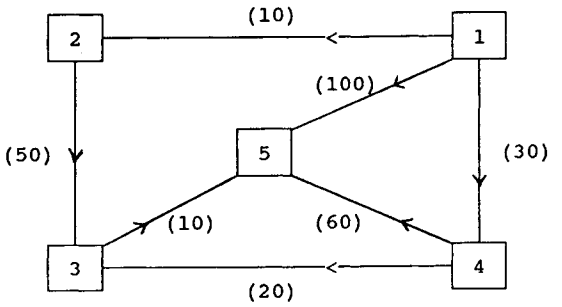

FIG. 2. An example of the shortest path problem: Given a graph of five vertices and seven directed and weighted edges, find a path from vertex 1 to vertex 5 such that the sum of the weights of its edges is minimum. Here, the answer is the path 1, 4, 3, 5, with a distance of 60.

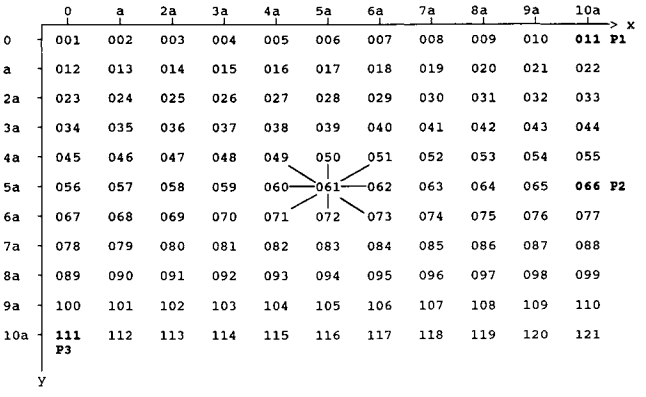

FIG. 3. The brachistochrone problem formulated as a graph shortest path problem. After a spatial lattice is introduced, the lattice sites are interpreted as the graph vertices. For each site, we allow up to eight transitions to the surrounding sites (shown here for vertex 61). These transitions, along with their transit times, are the graph edges and their weights. In an $M \times M$ lattice, there are $N = M^2$ vertices and $8N$-$12M$-8 edges.

For horizontal paths ($X_2 = X_1 \pm 1$, $Y_2 = Y_1 > 0$), the transit time is given by

$$\alpha[2Y_1(M-1)]^{-1/2}. \tag{5a}$$

For vertical paths ($X_2 = X_1$, $Y_2 = Y_1 \pm 1$) the transit time is given by

$$\alpha[2/(M-1)]^{1/2}|\sqrt{Y_2} - \sqrt{Y_1}|. \tag{5b}$$

For diagonal paths ($X_2 = X_1 \pm 1$, $Y_2 = Y_1 \pm 1$), taken as straight, the transit time is given by

$$\alpha[4/(M-1)]^{1/2}|\sqrt{Y_2} - \sqrt{Y_1}|. \tag{5c}$$

We set the matrix element (i,j) to one of these values after dropping the time scale α from them to be consistent with the variationally computed time of Sec. I, which is also linear in α.

We conclude this section by noting a point that is key to the success of our approach: The construction of the distance matrix (an essential prerequisite to Dijkstra's algorithm) involves the computation of the transit time between two given neighboring sites, *regardless of how the particle reached there*. Our ability to do so [viz. Eqs. (5)] stems from the conservative nature of gravity. Hence, our approach can be employed for any type of field, even if it is not uniform, as long as it is derivable from a potential.

IV. IMPLEMENTATION AND RESULTS

Dijkstra's standard algorithm requires that the distance matrix be stored as a two-dimensional $N \times N$ array. The memory requirement of such an array is prohibitive even for a relatively small lattice (for a 25×25 lattice, this array needs over 1.5 Mbytes). But since our isomorphism limited the number of possible paths emanating from a site (i.e., its edges) to its surrounding sites only, this matrix is highly sparse in the sense that most of its elements are ∞. According to the results of Sec. III, each row of the distance matrix has a maximum of eight noninfinite elements. Hence, instead of representing this matrix as a two-dimensional array, we represent it as a *function* of two arguments (the two vertex numbers), which is called dynamically to compute the distance between them.

The program was run for varying values of M, the number of lattice sites per side. We have used only odd values of M so that the point P2 [at $(L,L/2)$] would always be on a lattice site. For each run (which generates the shortest path from vertex 1 to *all* the other vertices), we recorded the transit times as well as the paths to the points P1, P2, and P3 of Sec. I.

The analysis of the results involves examining the transit times and their convergence as N increases, comparing with the variationally computed times and studying the paths themselves. To explain how this was done, we give an explicit analysis of the $M = 11$ case, which is small enough to be drawn on paper. We then state the results of the analysis when applied to the other cases.

Figure 4 shows the shortest paths to P1, P2, and P3 for an 11×11 lattice. The corresponding transit times are listed in Table I. We see that for P3, the shortest path as well as its transit time agree exactly with the continuum results. This exact agreement is expected for this trivial case but it does verify the correctness of the program. The computed times for P2 and P3 agree with the exact results (Table I)

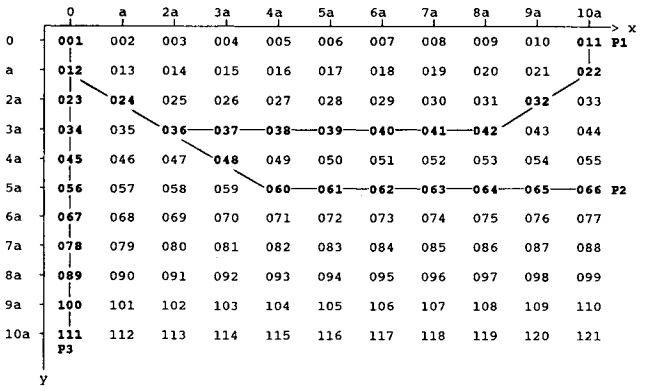

FIG. 4. The shortest paths to P1, P2, and P3 as obtained from Dijkstra's algorithm for an 11×11 lattice. The traversed sites of each path are shown in bold.

within a relative error of about 3%. This deviation is expected because, as shown in the Appendix, the lattice introduces an absolute error of up to 0.12α, which translates, in this case, to a relative error of 4.8%. As for the paths, we have calculated the Y-coordinate values of the cycloid [Eq. (1)] at the discrete values of X which correspond to lattice sites. The results are tabulated in Table II along with those obtained from our approach. The 11×11 lattice paths are consistent with the exact ones within a root-mean-square error of 0.03. This error is also expected because the lattice path can deviate from the continuum path by up to half the lattice spacing, or $1/2(m-1)$, which is 0.05 for $M = 11$. We conclude from this that even for an extremely small lattice, our approach can predict the shortest path as well as its transit time to within 3%–4% relative error.

A similar analysis was performed for a number of lattice sizes. The resulting transit times are shown in Table III. The analysis clearly shows that the method works and can provide good results even for relatively small lattices.

V. CONCLUSION

We have introduced a new interdisciplinary method for finding the minimum time path in mechanical systems. We believe that this approach has several advantages over the standard variational techniques. From an educational point of view, it offers students a deeper insight into the physics, requires far less mathematical background (the computations of the variational path $Y_{\infty\times\infty}$ of Table II involved techniques for numerical solution of nonlinear equations), and introduces the notion of the lattice as a tool for reducing the complexity of problems. In addition, it promises to work in situations where the variational ap-

TABLE I. The shortest transit times to P1, P2, and P3 obtained from Dijkstra's algorithm for an 11×11 lattice and from the exact variational computations ($\infty \times \infty$). The figures shown are dimensionless as they are measured in units of $\alpha = \sqrt{(L/g)}$.

P1		P2		P3	
$\infty \times \infty$	11×11	$\infty \times \infty$	11×11	$\infty \times \infty$	11×11
2.507	2.595	1.784	1.830	1.414	1.414

TABLE II. The shortest paths to P1 and P2 obtained from Dijkstra's algorithm for an 11×11 lattice ($Y_{11\times11}$) and from the exact variational computations ($Y_{\infty\times\infty}$). The (X,Y) coordinates are dimensionless (scaled by the lattice length): P1 = (1,0); P2 = (1,0.5).

	Path to P1		Path to P2	
X	$Y_{\infty\times\infty}$	$Y_{11\times11}$	$Y_{\infty\times\infty}$	$Y_{11\times11}$
0.0	0.000	0.000	0.000	0.000
0.1	0.206	0.200	0.168	0.200
0.2	0.306	0.300	0.241	0.300
0.3	0.377	0.400	0.286	0.300
0.4	0.430	0.500	0.310	0.300
0.5	0.468	0.500	0.318	0.300
0.6	0.495	0.500	0.310	0.300
0.7	0.511	0.500	0.286	0.300
0.8	0.517	0.500	0.241	0.300
0.9	0.513	0.500	0.168	0.200
1.0	0.500	0.500	0.000	0.000

proach fails (e.g., when the constraints are unusual) or is too complicated (nonuniform fields). The former case, which was considered recently by Stork *et al.*,[5–8] assumes the particle is sliding along the curve but not attached to it (i.e., like a block on an inclined plane, not like a bead on a wire). The latter case would be easy to handle in our approach because the field can be treated as uniform within one lattice plaqquet and hence, the distance matrix elements can be easily computed. Investigation of these two cases is currently underway. The method may provide smoother paths if parabolas, rather than straight lines, are considered for the diagonal edges of the graph.

TABLE III. The computed shortest times for P1, P2, and P3 for varying lattice sizes. The last row is from the exact variational computation. All figures are in units of α.

Lattice size	Transit time P1	Transit time P2	Transit time P3
3×3	2.828	1.914	1.414
5×5	2.707	1.871	1.414
7×7	2.648	1.842	1.414
9×9	2.613	1.832	1.414
11×11	2.595	1.830	1.414
13×13	2.584	1.828	1.414
15×15	2.582	1.829	1.414
17×17	2.576	1.831	1.414
19×19	2.577	1.830	1.414
21×21	2.574	1.828	1.414
$\infty\times\infty$	2.507	1.784	1.414

APPENDIX

We estimate the error introduced by the lattice by studying the effect of a small variation in the path on the transit time. The transit time to P1 along a path: $y(x)$ is given by the functional

$$T=\int\frac{ds}{v}=\int_0^L\frac{\sqrt{(1+y'^2)}}{\sqrt{(2gy)}}\,dx\equiv\int_0^L\Phi(y,y')\,dx\,.$$

If the path is varied by δy [consistent with the boundary conditions $y(0)=y(L)=0$], the transit time will pick up a variation δT given by

$$\delta T=\int_0^L\left(\frac{\partial\Phi}{\partial y}\delta y+\frac{\partial\Phi}{\partial y'}\delta y'\right)dx.$$

At any given x, the magnitude of δy represents the distance between the lattice path and the continuum path. Clearly, this distance is very small near the boundary points and is bounded by $a/2$ (a is the lattice spacing) at all the inner points. Hence we can obtain an estimate of $|\delta T|$ by restricting the integral to the interval $[a, L-a]$ and assuming $|\delta y|=a/2$ at all the inner points. Since $\delta y'=(\delta y)'=0$ along this path, we obtain

$$|\delta T|\leqslant\left(\frac{a}{2}\right)\int_a^{L-a}\left|\frac{\partial\Phi}{\partial y}\right|dx.$$

And upon integrating along the cycloid of Eqs. (1), this yields

$$|\delta T|\leqslant\frac{\alpha\sqrt{(2\pi)}}{2(M-1)}\cot\left(\frac{\Theta^*}{2}\right),$$

where Θ^* is the root of the equation $\Theta-\sin(\Theta)-2\pi/(M-1)=0$. We find that for $M=11$ $|\delta T|\leqslant0.12\alpha$, and for $M=21$ $|\delta T|\leqslant0.09\alpha$.

REFERENCES

1. H. H. Goldstine, *The History of the Calculus of Variations from the 17th through the 19th Century* (Springer, New York, 1980).
2. V. Chachra, P. M. Ghare, and J. M. Moore, *Applications of Graph Theory Algorithms* (North-Holland, New York, 1979), pp. 54; see also E. Horowitz and S. Sahni, *Fundamentals of Data Structures* (Computer Science Press, Potomac, 1976), pp. 301–308.
3. E. W. Dijkstra, Num. Math. **1**, 269 (1959).
4. Jerry Marion, *Classical Dynamics of Particles and Systems* (Academic, New York, 1965), pp. 202.
5. D. G. Stork, J. Yang, and C. Stover, Am. J. Phys. **54**, 992 (1986).
6. D. G. Stork, Am. J. Phys. **51**, 132 (1983).
7. J. Yang, D. G. Stork, and D. Galloway, Am. J. Phys. **55**, 844 (1987).
8. D. G. Stork and J. Yang, Am. J. Phys. **56**, 22 (1988).