# Exact Algorithms for Finding Fixed-Length Cycles in Edge-Weighted Graphs

R. Lewis
*School of Mathematics*
*Cardiff University*
Cardiff, Wales
LewisR9@cardiff.ac.uk

F. Carroll
*Cardiff School of Technologies*
*Cardiff Metropolitan University*
Cardiff, Wales
fcarroll@cardiffmet.ac.uk

***Abstract*—We describe our recent work on the problem of producing fixed-length cycles in edge-weighted graphs. We give two exact methods for this $\mathcal{NP}$-hard problem and briefly consider their scaling-up characteristics.**

***Index Terms*—Graph theory, networks, cycles, algorithms.**

## I. Introduction

In this research we consider the combinatorial problem of forming fixed-length cycles in edge-weighted graphs. These cycles must also start and end at a specific, user-defined vertex. In transport, this problem is relevant in the design of exercise routes and cycling tours [6]. Applications also arise in network visualisation [9], protein analysis [8], and drawing metabolic pathways [1].

The problem is formally defined as follows.

*Definition 1:* Let $G = (V, E)$ be a simple edge-weighted graph with $n$ vertices and $m$ edges. In addition, let $s \in V$ define a source vertex, $k$ define a desired length, and $w(u, v)$ denote the weight (length) of each edge $\{u, v\} \in E$. The *$k$ $s$-cycle problem* involves identifying a cycle $C = (s = u_1, u_2, \ldots, u_l = s)$ that starts and ends at $s$, and whose length $L(C) = \sum_{i=1}^{l-1} w(u_i, u_{i+1})$ minimises $\|k - L(C)\|$.

We recall that a cycle in a graph is a connected subgraph in which all vertices have a degree of exactly two. We also use the term $s$-cycle here to denote a cycle containing the vertex $s$. Two examples are shown in Fig. 1

Little work seems to have been conducted on the problem of finding fixed-length cycles in edge-weighted graphs. Various complexity results are known, however. The shortest $s$-cycle can be computed in polynomial time [4]; however, computing longest $s$-cycles is $\mathcal{NP}$-hard, both for weighted and unweighted graphs [2]. Similarly, shortest paths between pairs of vertices can be identified in polynomial time [3], while the problem of finding longest paths is $\mathcal{NP}$-hard [5] (though certain topologies such as trees and directed acyclic graphs can be solved in polynomial time). The problem of counting the number of $s$-$t$-paths and $s$-cycles in a graph is also known to be $\#\mathcal{P}$-complete [7].

As a generalisation of the longest $s$-cycle problem, the $k$ $s$-cycle problem is also $\mathcal{NP}$-hard. That said, it is still useful to investigate suitable exact algorithms for this problem, particularly for use with small- and medium-sized problem instances. Two such algorithms are presented in the following sections, one based on integer programming (IP) and one that uses Yen's algorithm [10].

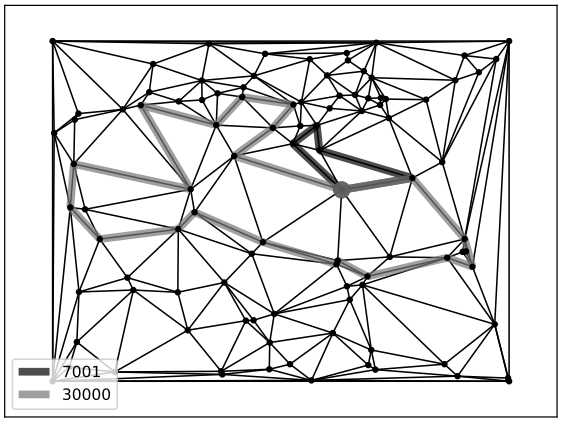


Fig. 1. A planar graph $G$ comprising $n = 100$ vertices placed randomly in a $10,000 \times 10,000$ square. Edge weights correspond to Euclidean distances. The two $s$-cycles give optimal solutions for $k = 7000$ and $k = 30,000$ using the indicated source vertex.

## II. IP Formulation

Our IP formulation is adapted from a recent model for finding longest $s$-cycles in unweighted graphs [2]. Given $G = (V, E)$, let $V = \{v_1, v_2, \ldots, v_n\}$, and assume (w.l.o.g.) that $s = v_1$. The adjacencies and weights of $G$ are now stored in the matrices $\mathbf{A}_{n\times n}$ and $\mathbf{W}_{n\times n}$ such that $A_{ij} = 1$ if $\{v_i, v_j\} \in E$ (and $A_{ij} = 0$ otherwise); and $W_{ij} = w(v_i, v_j)$ if $\{v_i, v_j\} \in E$ (and $W_{ij} = \infty$ otherwise).

In this formulation, a cycle is stored using binary variables $X_{ij}$, where $X_{ij} = 1$ signifies that a transition is made from $v_i$ to $v_j$ in the cycle, and $X_{ij} = 0$ otherwise. The objective is to now

$$\text{minimize: } \left\| k - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} W_{ij} X_{ij} \right\| \tag{1}$$

subject to:

$$X_{ij} \leq A_{ij} \qquad \forall i \in \{1,\ldots,n\} \quad \forall j \in \{1,\ldots,n\} \quad (2)$$

$$\sum_{i=1}^{n} X_{ij} - \sum_{i=1}^{n} X_{ji} = 0 \qquad \forall j \in \{1,\ldots,n\} \quad (3)$$

$$\sum_{i=1}^{n} X_{ij} + \sum_{i=1}^{n} X_{ji} \leq 2 \qquad \forall j \in \{1,\ldots,n\} \quad (4)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}X_{ij} \geq 3 \quad (5)$$

$$0 \leq Y_{ij} \leq (n-1)X_{ij} \qquad \forall i \in \{2,\ldots,n\}, \quad \forall j \in \{1,\ldots,n\} \quad (6)$$

$$2\sum_{\substack{j=1:\\ j\neq i}}^{n} Y_{ij} - 2\sum_{\substack{j=2:\\ j\neq i}}^{n} Y_{ji} - \sum_{\substack{j=1:\\ j\neq i}}^{n} X_{ij} - \sum_{\substack{j=2:\\ j\neq i}}^{n} X_{ji} = 0 \qquad \forall i \in \{2,\ldots,n\}. \quad (7)$$

Here, the term $\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}W_{ij}X_{ij}$ in (1) gives the length of the cycle, while (2) ensures that $X_{ij} = 1$ if and only if $\{v_i, v_j\} \in E$. Constraints (3) and (4) ensure that a vertex $v_i$ is in the cycle only when it has exactly one edge entering and one edge leaving, else it has no such edges. Constraint (5) also ensures that cycles contain at least three edges. Finally, (6) and (7) make use of the auxiliary variables $Y_{ij} \in \mathbb{R}$. These represent flows on the edges and are used to ensure that solutions comprise a single $s$-cycle (as opposed to several disjoint cycles) [2].

## III. Yen's Algorithm

Our second exact method for this problem operates by first adding a vertex $s'$ to $G$ and setting its neighbouring vertices to those of $s$. Solving the $k$ $s$-cycle problem now involves identifying an $s$-$s'$-path of length $k$ that has at least three edges.

A suitable algorithm for this task is due to Yen [10]. Yen's algorithm determines the $K$ shortest paths between any two given vertices and operates by finding the shortest path, followed by the second shortest path, the third shortest path, and so on. For our current problem Yen's algorithm needs to be executed until an $s$-$s'$-path of length greater or equal to $k$ is observed. The generated $s$-$s'$-path whose length is closest to $k$ (and that contains at least three edges) then corresponds to an optimal solution.

Note that the complexity of Yen's algorithm is $\mathcal{O}(Kn(m + n\lg n))$, where $\mathcal{O}(m + n\lg n)$ is the asymptotic complexity of Dijkstra's shortest path algorithm. This means that larger values for $K$ lead to increased run times. For this current application, $K$ is not known beforehand so the algorithm must iterate until a suitable $s$-$s'$-path is identified. This can bring scaling-up issues.

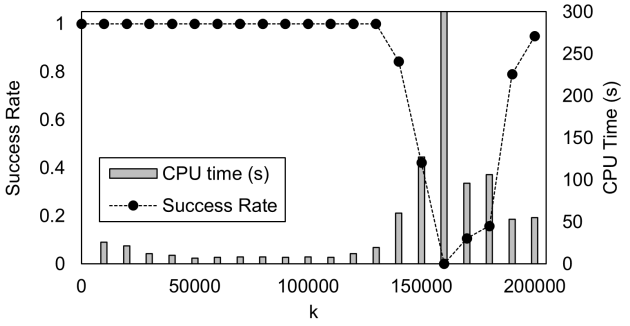


Fig. 2. Performance of the IP method across twenty 100-vertex Euclidean planar graphs for various values of $k$. Success rates give the proportion of instances solved to optimality within a 300-second time limit. CPU times give the mean solution times across successful runs.

## IV. Preliminary Results and Discussion

Figure 2 gives some preliminary results for our IP method under a five-minute time limit.[1] For these 100-vertex graphs, most values of $k$ are solved to optimality within one minute. In contrast, our approach using Yen's algorithm was less favourable, with optimality never being achieved within the time limit for values of $k > 10000$.

In our experiments, larger planar graphs of more than 1000 vertices have also been solved to optimality using these IP methods, though success rates seem to drop quite quickly for larger values of $k$. To deal with these scaling issues, our current work is concerned with developing methods based on local search. We are also considering differing graph topologies and methods of problem decomposition. Finally, we are also extending these methods for use with the *$k$ cycle problem*. Here the aim is to find *any* cycle of length $k$ in a graph, as opposed to an $s$-cycle.

## References


[1] M. Becker and I. Rojas, "A graph layout algorithm for drawing metabolic pathways," *Bioinformatics*, vol. 17, no. 5, pp. 461–467, 05 2001.

[2] D. Chalupa, P. Balagan, K. Hawick, and N. Gordon, "Computational methods for finding long simple cycles in complex networks," *Knowledge-Based Systems*, vol. 125, pp. 96–107, 2017.

[3] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 269-271, 1959.

[4] G. Ducoffe, "Faster approximation algorithms for computing shortest cycles on weighted graphs," *SIAM Journal on Discrete Mathematics*, vol. 35, no. 2, pp. 953–969, 2021.

[5] M. Garey and D. Johnson, *Computers and Intractability - A guide to NP-completeness*. San Francisco: W. H. Freeman and Co., 1979.

[6] R. Lewis, "A heuristic algorithm for finding attractive fixed-length circuits in street maps," in *Computational Logistics*, ser. Lecture Notes in Computer Science. Springer, 2020, no. 12433, pp. 384–395.

[7] B. Roberts and D. Kroese, "Estimating the number of $s$-$t$ paths in a graph," *Journal of Graph Algorithms and Applications*, vol. 11, no. 1, pp. 195–214, 2007.

[8] L. Salwinski, C. Miller, A. Smith, F. Pettit, J. Bowie, and D. Eisenberg, "The database of interacting proteins: 2004 update," *Nucleic Acids Research*, vol. 32, no. 1, pp. 449–451, 2004.

[9] R. Tamassia, Ed., *Handbook of Graph Drawing and Visualization*, ser. Discrete Mathematics and Its Applications. Chapman and Hall, 2016.

[10] J. Yen, "Finding the $K$ shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 661–786, 1971.


[1] Executed on a 3.2 GHz PC with 8 GB RAM using Fico Xpress 8.13.