Adaptation of a branching algorithm to solve the multi-objective Hamiltonian cycle problem

Maialen Murua[1], Diego Galar[1,2], and Roberto Santana[3]

[1] TECNALIA, Mikeletegi Pasealeakua 7, 20009 Donostia-San Sebastián (Spain).
maialen.murua@tecnalia.com
[2] Division of Operation and Maintenance Engineering, Luleå University of Technology, 91787 Luleå (Sweden).
[3] Department of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU), Lardizabal Pasealeakua 1, 20018 Donostia-San Sebastián (Spain).

Abstract. The Hamiltonian cycle problem (HCP) consists of finding a cycle of length N in an N-vertices graph. In this investigation, a graph G is considered with an associated set of matrices, in which each cell in the matrix corresponds to the weight of an arc. Thus, a multi-objective variant of the HCP is addressed and a Pareto set of solutions that minimizes the weights of the arcs for each objective is computed. To solve the HCP problem, the Branch-and-Fix algorithm is employed, a specific branching algorithm that uses the embedding of the problem in a particular stochastic process. To address the multi-objective HCP, the Branch-and-Fix algorithm is extended by computing different Hamiltonian cycles and fathoming the branches of the tree at earlier stages. The introduced anytime algorithm can produce a valid solution at any time of the execution, improving the quality of the Pareto Set as time increases.

Keywords: Multi-objective optimization, Discrete optimization problems, Hamiltonian cycle problem, Branching algorithm

1 Introduction

The Hamiltonian cycle problem (HCP) consists of finding a cycle of length N in an N-vertices graph [4]. These cycles are called Hamiltonian cycles (HC)s. If the graph G contains at least one HC, G is said to be a Hamiltonian graph. The traveling salesman problem (TSP) is the problem of finding the shortest route to go to N different cities visiting each city once and returning to the city of origin [3]. The distances between all pairs of cities are known. Let G be a directed graph of N vertices with a weight c_{ij} for each arc (i, j), then a solution of the TSP corresponds to a HC of minimum total weight [5]. The main difference between these two problems is that finding a HC in a graph might be complex.

In this investigation, the HCP is considered for a graph G of size N, in which the Branch-and-Fix (BF) algorithm [1] has been employed to solve the problem. G has an associated set of matrices $\boldsymbol{W}^1, ..., \boldsymbol{W}^k$, where the cell $w^l_{i,j}$ represents the l^{th} weight associated to an arc (i, j) for $l \in \{1, ..., k\}$. Each matrix represents a different way to evaluate HCs in G. We address a multi-objective variant of the HCP, considering only graphs that are Hamiltonian. The optimal Pareto set (PS) will comprise HCs that minimize the sum of weights for the different matrices. Notice that we simultaneously address the problem of finding a HC and minimizing the weights associated to the arcs. Our approach consists of modifying and extending the BF to deal with the multi-objective variant of the HCP.

The BF algorithm uses the embedding of the HCP in a discounted Markov Decision Process (MDP), a particular stochastic process that provides a mathematical framework for modeling decision making [2]. The problem addressed in this investigation consists of finding the set of non-dominated HCs (HCs that have associated non-dominated objective functions), where HCs will be found using the BF algorithm. The algorithm is modified in order to stop exploring a branch of the tree that will lead to a HC that is dominated by other previously found HC.

2 Branch-and-fix method

The BF was proposed in 2009 by Ejov et al. [1] to solve the HCP in a given graph. It is based on the idea that the HCP can be embedded in a MDP, and uses the polytope $\boldsymbol{X}_\beta$ defined by Feinberg [2] in 2000. In that investigation it was shown that the extreme points of the polytope correspond to HCs, though in 2009 it was found [6] that the extreme points could also be 1-randomized policies. The BF avoids arriving to those undesirable extreme points that induce 1-randomized policies. It solves sequences of linear programs (LP)s, two at each branching point of the logical tree.

1. **Initialization**. The original LP is solved to find $x \in \mathbf{X}_\beta$. If the feasible solution x_0 induces a deterministic policy $\boldsymbol{\zeta}_0$, the solution is found, if not, $\boldsymbol{\zeta}_0$ is a 1-randomized policy.
2. **Branching**. The 1-randomized policy $\boldsymbol{\zeta}_0$ serves to identify the splitting node i where the randomization occurs. If d is the number of arcs $\{(i, a_1), ..., (i, a_d)\}$ that emanate from i, d subgraphs $G_1, G_2, ..., G_d$ are constructed. In each G_k, where $k = 1, ..., d$, (i, a_k) is fixed. These graphs are identical to G in all other vertices and arcs.
3. **Fixing**. In this step some other arcs are fixed, as in some graphs fixing one arc implies fixing other arcs.
4. **Iteration**. A second LP is solved to check the feasibility of the current fixed arcs. If that is the case, the algorithm is called recursively with the updated graph. If it is not feasible, the algorithm returns to step 2 fixing the following arc that emanates from i. The algorithm terminates when a HC is found or after exploring all the branches (in that case the graph is not Hamiltonian).

3 Multi-objective HCP

Let $G = (V, \mathcal{A})$ be a graph of size N, where V is the set of nodes $|V| = N$ and $\mathcal{A}$ the set of the undirected arcs. We define k asymmetric weight-matrices, $\boldsymbol{W}^l$, $l \in \{1, ..., k\}$, where each arc $(i, j) \in \mathcal{A}$ has associated a weight $w^l_{ij} > 0$ in each of the k matrices. At the same time, we denote δ^{HC}_l as the objective function associated to a matrix $\boldsymbol{W}^l$ and specific HC, defined in Equation (1).

$$\delta^{HC}_l = \sum_{m=1}^{|\mathcal{A}|} x_m \times w^l_m \qquad x_m \in \{0, 1\} \tag{1}$$

Where $x_m = 1$ if the m^{th} arc belongs to the HC and $x_m = 0$ otherwise. w^l_m represents the entry of $\boldsymbol{W}^l$ corresponding to arc m. Finally, $\boldsymbol{\delta^{HC}}$ is the multiple-objective function associated to a HC defined in Equation (2).

$$\boldsymbol{\delta^{HC}} = \{\delta^{HC}_1, ..., \delta^{HC}_k\} \tag{2}$$

An important constraint in this framework is that G is not a complete graph, therefore finding a HC is not trivial. A straightforward approach for finding the non-dominated HCs is by enumerating all possible solutions and selecting the optimal ones. However, complete enumeration is not a feasible approach even for small values of N. BF searches the space of HCs by pruning branches that are guaranteed not to lead to a valid HC. We extend the same rational but adding the possible dominance of a point as a new criterion for fathoming.

Example 1. Consider the directed graph of size 6 shown in Figure 1 and two asymmetric weight-matrices. There are a total of 6 HCs in the graph:

$$\{(4, 6, 2, 3, 1, 5), (2, 6, 4, 5, 1, 3), (2, 3, 6, 1, 4, 5), (4, 1, 2, 5, 6, 3), (5, 3, 4, 1, 6, 2)$$
$$(5, 1, 6, 3, 4, 2)\}$$

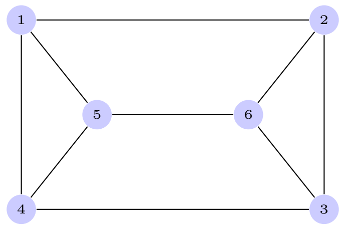

Fig. 1. An undirected graph of 6 vertices.

For instance, for the first $HC_1 = (4, 6, 2, 3, 1, 5)$, the set of arcs belonging to the HC is $\{(1, 4), (2, 6), (3, 2), (4, 3), (5, 1), (6, 5)\}$. To calculate $\delta_1^{HC_1}$, the w_{ij}^1 of the first weight matrix $\boldsymbol{W}^1$ that corresponds to each arc in the set are summed up.

$$\delta_1^{HC_1} = w_{14}^1 + w_{26}^1 + w_{32}^1 + w_{43}^1 + w_{51}^1 + w_{65}^1$$

In the same way $\delta_2^{HC_1}$ is calculated using $\boldsymbol{W}^2$. The bi-objective function $\boldsymbol{\delta^{HC_1}}$, is compound by $\delta_1^{HC_1}$ and $\delta_2^{HC_1}$. The following bi-objective functions are considered:

$$\boldsymbol{\delta^{HC_1}} = \{5309, 613.4\}\ \boldsymbol{\delta^{HC_2}} = \{6416, 514.3\}\ \boldsymbol{\delta^{HC_3}} = \{5347, 516.98\}$$

$$\boldsymbol{\delta^{HC_4}} = \{4338, 516.98\}\ \boldsymbol{\delta^{HC_5}} = \{5256, 613.4\}\ \boldsymbol{\delta^{HC_6}} = \{5422, 514.3\}$$

.

Based on this, the PS and non-dominated HCs can be calculated. The non-dominated HCs are HC_4 and HC_6 since the vectors of objectives $\boldsymbol{\delta^{HC_4}}$ and $\boldsymbol{\delta^{HC_6}}$ are non-dominated.

$$\text{PS} = \{(4338, 516.98), (5422, 514.3)\}$$

3.1 Adaptation of the BF to compute a PS

The BF algorithm is extended to deal with the multi-objective HCP computing a PS of a given graph. In principle, the algorithm allows more than two objectives. The user may define an objective using the corresponding matrix as input. For that purpose the following changes were made to the original algorithm:

1. **Initialization:** Save a HC whenever it is found without stopping the search and continuing exploring the tree.
2. **Initialization:** Compute the objective functions for the HCs and calculate the PS.
3. **Iteration:** Stop exploring a branch that will return a HC which is dominated by any solution already in the PS. A branch will be fathomed if the $\boldsymbol{\delta^{\mathcal{U}}}$ is dominated by a $\boldsymbol{\delta^{HC}}$, being $\mathcal{U}$ the set of fixed arcs at the current stage. This check is performed in the fourth stage of the method, called iteration, after computing the second LP.

An anytime algorithm is an algorithm whose quality of results enhances progressively as computation time increases. One of the main properties of these algorithms is the interruptibility, as the algorithm can be stopped at any time and provide a solution. Provided that at least one HC has been found, our approach can be understood as an anytime algorithm, as a PS is returned at any time and it is improved as computation time increases.

4 Experiments

The algorithm was implemented in Python (version 3.5.2) and CPLEX (version 12.7.1) was used to solve the LPs. 20 random Hamiltonian graphs of size 20 were generated to perform the experiments, using the following procedure: 1) Generate random symmetric matrices of 20×20 dimension of zeros and ones; 2) Use a random permutation of length 20 to assure that at least there is one HC, connecting the vertices indicated by the permutation; 3) Fill the diagonal of the matrices with zeros.

For each of the graphs, two weight matrices were also generated randomly. To generate the first matrix, the samples were extracted randomly from a normal distribution, $w^1_{ij} \sim \mathcal{N}(\mu, \sigma) \times c$, where $\mu = 10$, $\sigma = 3$ and $c = 10$. The values of the second matrix were generated using linear regression $w^2_{ij} = \alpha \times w^1_{ij} + |\epsilon|$, where $\alpha = 0.01$ and $\epsilon \sim \mathcal{N}(\mu, \sigma)$, $\mu = 0$ and $\sigma = 100$.

Table 1. Number of HCs found by the BF and the cardinality of the PS for the 20 Hamiltonian graphs in three time periods.

	10 minutes		20 minutes		30 minutes	
	HCs	PS	HCs	PS	HCs	PS
G_1	175	8	347	11	417	11
G_2	337	8	558	7	768	8
G_3	758	15	1468	17	2227	18
G_4	983	12	1622	13	2025	13
G_5	487	7	1085	5	1573	7
G_6	830	16	1324	15	1617	16
G_7	397	6	614	6	691	6
G_8	788	6	1430	6	2027	6
G_9	399	13	713	10	974	9
G_{10}	1073	14	1631	17	2234	18
G_{11}	538	7	877	8	1278	8
G_{12}	287	6	604	6	905	12
G_{13}	501	9	841	9	1277	15
G_{14}	767	13	1290	13	1915	11
G_{15}	622	9	1297	13	1817	11
G_{16}	606	14	1436	13	1768	14
G_{17}	443	4	881	7	1188	12
G_{18}	576	14	898	15	1240	17
G_{19}	1056	15	1811	11	2592	11
G_{20}	515	3	862	5	1102	5

Table 1 shows the number of HCs and the cardinality of the PS computed by the BF in 10 minutes, 20 minutes and 30 minutes of time execution. It can be observed that the number of HCs increases with the time as expected, though the size of the increase differs from graph to graph. The reason for this is that the difficulty of finding a HC depends on the graph, as in some cases the algorithm has to explore deeper levels of the tree. However, this phenomenon is not repeated for the cardinality of the PS since more HCs do not necessarily imply more elements in the PS.

5 Conclusions

We have introduced an extension of the BF algorithm to solve a multi-objective variant of the HCP. It was shown that the extended BF algorithm can simultaneously find several HCs and compute continuously an improving PS. The research presented can be improved executing the BF simultaneously by changing the labels of the graph vertices, as different HCs will be found for each of the representations. As a consequence, the obtained PSs will also be different and the algorithm could be considered as a parallel algorithm.

Acknowledgment

This project was funded by the ELKARTEK Research Programme of the Basque Government (project KK-2019/00068). This work has been possible thanks to the support of the computing infrastructure of the i2BASQUE academic network. The work of Roberto Santana was funded by the Basque Government (project IT-1244-19), and Spanish Ministry of Economy and Competitiveness MINECO (project TIN2016-78365-R).

References

1. V. Ejov, J. Filar, M. Haythorpe, and G. Nguyen. Refined MDP-based branch-and-fix algorithm for the Hamiltonian cycle problem. *Mathematics of Operations Research*, 34:758–768, 2009.
2. E. Feinberg. Constrained discounted Markov decision process and Hamiltonian cycles. *Mathematics of Operations Research*, 25:130–140, 2000.
3. D. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, pages 154–159, 1985.
4. M. Haythorpe. *Markov Chain Based Algorithms for the Hamiltonian Cycle Problem.* PhD thesis, University of South Australia, 2011.
5. J. Lenstra and A. R. Kan. Some simple applications of the travelling salesman problem. *Journal of the Operational Research Society*, 26:717–733, 1975.
6. G. Nguyen. *Hamiltonian cycle problem, Markov decision processes and graph spectra.* PhD thesis, University of South Australia, 2009.