An Alternative ILP Model and Algorithmic Ideas for the Maximum Edge-Disjoint Paths Problem

Christian Blum[1], Maria J. Blesa[2], Abraham Duarte[3], Jesús Sánchez-Oro[3]

[1] Artificial Intelligence Research Institute (IIIA-CSIC)
Campus of the UAB
Bellaterra, Spain
christian.blum@iiia.csic.es

[2] Computer Science Department
Universitat Politècnica de Catalunya - BarcelonaTech
Barcelona, Spain
mjblesa@cs.upc.edu

[3] Dept. Computer Science
Universidad Rey Juan Carlos
Móstoles, Madrid, Spain
{abraham.duarte, jesus.sanchezoro}@urjc.es

Abstract

This document describes an alternative integer linear programming (ILP) model for the so-called edge-disjoint paths (EDP) problem in undirected graphs. EDP is an $\mathcal{NP}$-hard problem where exact methods are not able to produce high quality solutions. Therefore, we propose two different algorithms for combining exact and heuristic methods. On the one hand, we consider a restricted model that limits the number of paths between two given nodes in the graphs (which reduces the search space exploration). On the other hand, the application of a mat-heuristic algorithm known as Construct, Merge, Solve and Adapt (CMSA) is considered. In this document we show some preliminary results concerning the restricted model. These results indicate the potential usefulness of the presented ideas.

1 Introduction

Let $G = (V, E)$ be an edge-weighted undirected graph representing a network in which the nodes may represent hosts and switches, and the edges links between them. The weight $w(e) \in R^+$ of an edge $e \in E$ corresponds to the distance between its endpoints. Let $T = \{(s_j, t_j)|j = 1, \ldots, |T|; s_j \neq t_j \in V\}$ be a list of commodities, i.e., pairs of nodes in G, representing endpoints demanding to be connected by a simple path in G. T is said to be realizable in G if there exist mutually edge-disjoint paths from s_j to t_j in G, for every $j = 1, \ldots, |T|$. Deciding whether a given set of pairs is realizable in a given graph is one of Karp's original $\mathcal{NP}$-complete problems [2]. The problem remains $\mathcal{NP}$-complete for various graph types such as, for example, two-dimensional meshes.

The combinatorial optimization version of this problem consists in satisfying as many of the requests as possible, which is equivalent to finding a realizable subset of T of maximum cardinality. An EDP solution S to the combinatorial optimization problem is a set of disjoint paths, in which each path satisfies the connection request for a different commodity. The objective function value $f(S)$ of a solution S is defined as $f(S) = |S|$.

2 Alternative ILP model

Let P_j be the ordered set of all different paths connecting the two endpoints of commodity j (that is, s_j and t_j) in G. The term ordered refers here to the fact that it is possible to refer to the i-th path in the set. Let the i-th path in P_j be denoted by $p_{i,j}$. For being able to phrase the alternative ILP model we introduce a binary variable $x_{i,j}$ for each path $p_{i,j} \in P_j$, for $i = 1, ..., |P_j|$ and $j = 1, \ldots, |T|$. Setting $x_{i,j}$ to one means that path $p_{i,j}$ is selected to form part of the solution. Moreover, two variables $x_{i,j}$ and

$x_{k,l}$ (with $j \neq l$) are said to be in conflict if the corresponding paths $p_{i,j}$ and $p_{k,l}$ share at least one edge. With this definition we can phrase the following ILP for the EDP problem.

$$\max \sum_{j=1}^{|T|} \sum_{i=1}^{|P_j|} x_{i,j} \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{|P_j|} x_{i,j} \leq 1 \qquad j = 1, \ldots, |T| \tag{2}$$

$$x_{i,j} + x_{k,l} \leq 1 \qquad \text{for each pair of variables } x_{i,j} \neq x_{k,l} \text{ in conflict} \tag{3}$$

$$x_{i,j} \in \{0, 1\} \qquad j = 1, \ldots, |T| \text{ and } i = 1, \ldots, |P_j| \tag{4}$$

Hereby, constraints 2 ensure that at most one path per commodity is chosen. And constraints 3 ensure that the chosen paths are mutually edge-disjoint. For normal-size instances, this model is not practical if the aim is to solve it directly with a general-purpose ILP solver. This is because it has an exponential number of variables and an exponential number of constraints. This is a classical model that could, in principle, be tackled by column generation. However, a considerable amount of expert knowledge in operations research is necessary for doing so. Therefore, we propose some alternative algorithmic ideas for generating approximate solutions on the basis of the above-described ILP model in the next section.

3 Algorithmic proposal

This Section is devoted to present two different ideas for exploiting this model reducing the computational effort needed to obtain high quality solutions. On the one hand, we describe a restricted model with a lower number of variables and constraints. On the other hand, we propose a mat-heuristic algorithm known as Construct, Merge, Solve & Adapt (CMSA).

3.1 Solving a restricted model

The main problem of the original ILP model [3] is the large number of available paths existing between two nodes of a given commodity. Therefore, instead of considering the full sets $P_j, j = 1, \ldots, |T|$, we propose to restrict each set P_j to the X shortest paths for each commodity. Let $P_j^X \subseteq P_j$ be the set of the X shortest paths for commodity $j, j = 1, \ldots, |T|$. By replacing all P_j in the ILP model by sets P_j^X we obtain a restricted model that, depending on the value of X, may be solved to optimality. The computational experiments will show an extensive study for different values of X in order to find out for which values of X it would be feasible to solve the ILP model with a solver. Moreover, it would be interesting to see how the results compare to the best previous heuristic algorithms concerning the state of the art and with the proposed mat-heuristic algorithm presented in Section 3.2.

3.2 Construct, Merge, Solve & Adapt

Construct, Merge, Solve & Adapt (CMSA) is a recent generic mat-heuristic that performs something very similar to column generation. However, instead of a lot of expert knowledge, it is only necessary to have a way for probabilistically constructing solutions to the tackled problem, and to have an ILP model for the tackled problem.

The CMSA algorithm was introduced in [1] with the same motivation that led already to the development of large neighborhood search. More specifically, the CMSA algorithm is designed in order to be able to take profit from an efficient complete solver even in the context of problem instances that are too large to be solved directly by the complete solver. The general idea of CMSA is as follows. At each iteration, solutions to the tackled problem instance are generated in a probabilistic way. The solution components found in these solutions are then added to a sub-instance of the original problem instance.

Subsequently, an exact solver such as, for example, Gurobi, is used to solve the sub-instance to optimality. Moreover, the algorithm is equipped with a mechanism for deleting seemingly useless solution components from the sub-instance. This is done such that the sub-instance has a moderate size and can be solved rather quickly to optimality.

4 Computational experiments

This Section presents some preliminary results obtained when applying the general-purpose ILP solver Gurobi to the (restricted) alternative ILP formulation (AILP) and to the best ILP model from the literature. Table 1 shows the obtained results when considering 10, 25, and 50 shortest paths in the AILP, that is, $X \in \{10, 25, 50\}$. Note that all numbers are averages over 60 different sets of commodities, and the maximum allowed computing time is 100 seconds. We do not report the results obtained by CMSA algorithm since it is a work in progress, and there are no preliminary results yet.

Regarding small instances, the best results (concerning the AILP model) are obtained when considering the highest number of shortest paths, since the solver is able to optimally solve the restricted model. However, when considering larger instances, better results are obtained with the lowest number of shortest paths, since the number of constraints is proportional to the number of paths.

Furthermore, regarding the set of most challenging instances (based on mesh25x25), the results concerning the AILP drastically outperform the results obtained by the standard ILP formulation, which shows the relevance of the proposal. Finally, it is worth mentioning the difference between the computing time required by the ILP (5000 seconds) and the AILP (193 seconds in the worst case).

Instances	**ILP**	**AILP(10)**	**AILP(25)**	**AILP(50)**
AS-BA.R-Wax.v100e217	13.47	12.47	12.78	12.98
bl-wr2-wht2.10-50.sdeg	44.20	37.18	34.53	33.42
graph3	30.08	27.62	29.50	28.55
graph4	42.95	33.70	36.40	37.85
mesh25x25	24.28	64.58	52.95	51.90
Average Time (s)	5000	17.30	43.57	193.09

Table 1: Preliminary results of the alternative ILP formulation when considering 10, 25, and 50 shortest paths, respectively.

Acknowledgments

This research has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness (grant ref. TIN2015-65460-C2-2-P) and by the Agency for Management of University and Research Grants (AGAUR) of the Government of Catalonia (project ref. SGR 2014-1034).

References

[1] C. Blum, P. Pinacho, M. López-Ibáñez, and J. A. Lozano. Construct, Merge, Solve & Adapt: A new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68:75 – 88, 2016.

[2] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[3] Sánchez A. Beltrán C. Martín, B. and A. Duarte. A Matheuristic Approach for Solving the Edge-Disjoint Paths Problem. In V. Maniezzo and T. Stützle, editors, *Matheuristics 2016 - Proceedings of the Sixth International Workshop on Model-based Metaheuristics*, pages 25–34, Brussels, Belgium, 2016. IRIDIA, Université Libre de Bruxelles.