Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-12 (2016) 117–120

An Efficient Matheuristic for the Multicommodity Fixed-Charge Network Design Problem

Bernard Gendron * **Saïd Hanafi** ** **Raca Todosijević** ***

* *Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada, (e-mail: Bernard.Gendron@cirrelt.ca)*

** *LAMIH UMR CNRS 8201 - Université de Valenciennes, 59313 Valenciennes Cedex 9, France, (e-mail: Said.Hanafi@univ-valenciennes.fr).*

*** *LAMIH UMR CNRS 8201 - Université de Valenciennes, 59313 Valenciennes Cedex 9, France, (e-mail: racatodosijevic@gmail.com)*

Abstract: In this paper we study the Multicommodity Fixed-Charge Network Design problem. We propose an Iterative linear programming-based heuristic for solving this NP hard problem. The proposed heuristics have been tested on the benchmark instances from the literature. The quality of solutions obtained by each of them has been disclosed comparing them with corresponding solutions of the current state-of-the-art heuristics, i.e., Cycle-Based Evolutionary algorithms.

Keywords: network design problem; heuristic; pseudo-cut; matheuristic

1. INTRODUCTION

Network design models arise in large applications in telecommunications, transportation, logistics and production planning Balakrishnan et al. (1997, 1991); Minoux (1989); Gavish (1991). The multicommodity capacitated fixed-charge network design problem (MCND), is an NP-hard discrete optimization problem Magnanti and Wong (1984). It is defined on a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs. Each commodity $k \in K$ has a demand $d_k > 0$ to be routed from an origin O_k to a destination D_k. Each arc (i, j), has the capacity $u_{ij} > 0$ on the flow of all commodities circulating on the arc, the transportation cost $c_{ij} \geq 0$ and the fixed design cost $f_{ij} \geq 0$. The problem consists of minimizing the total cost while satisfying the demands and respecting the capacity constraints. The total cost includes the total transportation cost for transferring commodities from the origins to the destinations and the total fixed cost for using arcs.

Let b_{ij}^k be equal to $\min\{d_k, u_{ij}\}$, $N_i^+ = \{j \in N | (i, j) \in A\}$ and $N_i^- = \{j \in N | (j, i) \in A\}$. Then, the MCND problem may be modelled as a mixed-integer program (MIP) using continuous flow variables x_{ij}^k that represent the amount of flow on each arc (i, j) for each commodity k, and 0-1 design variables y_{ij} that indicate if the arc (i, j) is used or not:

$$\min z = f(x, y) = \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \quad (1)$$

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} d_k, & \text{if } i = O_k \\ 0, & \text{if } i \neq O_k, D_k \\ -d_k, & \text{if } i = D_k \end{cases} \quad (2)$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \ \forall (i, j) \in A \quad (3)$$

$$x_{ij}^k \leq b_{ij}^k y_{ij}, \ \forall (i, j) \in A, k \in K \quad (4)$$

$$x_{ij}^k \geq 0, \ \forall (i, j) \in A, k \in K \quad (5)$$

$$y_{ij} \in \{0, 1\}, \ \forall (i, j) \in A \quad (6)$$

Equations (2) are the flow conservation constraints for each node and each commodity. The capacity constraints (3) ensure that the capacity of each arc is respected. Additonally, they forbidds any flow to circulate through an arc that is not chosen as part of the design. The so-called strong inequalities, (4), ensure the same, therefore, they are redundant. However, they significantly improve the linear programming (LP) relaxation bounds (Crainic et al., 1999). Note that model (1)-(6) represents the strong formulation of the MCND, while the weak formulation being obtained by removing constraints (4). Hence, their corresponding LP relaxations are, respectively, the strong and the weak relaxations.

For solving this NP hard problem a plenty of exact and heuristic approaches have been proposed in the literature up to now. Regarding exact approaches a lot of work has been dedicated for developing Benders decomposition methods (Costa et al., 2009, 2012), Lagrangian based procedures (Crainic et al., 2001, 1999; Holmberg and Yuan, 2000; Frangioni and Gorgone, 2013; Gendron and Crainic,

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.07.560

1994; Kliewer and Timajev, 2005; Sellmann et al., 2002), branch & price and cutting plane methods (Chouman et al., 2011; Hewitt et al., 2013; Gendron and Larose, 2014). On the other hand, regarding heuristic approaches able to produce high quality solutions there are: A slope scaling/Lagrangean perturbation heuristic (Crainic et al., 2004), a simplex based tabu search proposed by Crainic et al. (2000), heuristics that exploit cycle-based neighborhood (Ghamlouche et al., 2003, 2004; Paraskevopoulos et al., 2013, 2014), scatter search based heuristic (Crainic and Gendreau, 2007), heuristics that use parallel cooperative strategies (Crainic and Gendreau, 2002; Crainic et al., 2006); capacity scaling heuristic proposed by (Katayama et al., 2009); a hybrid approach that combines simulated annealing and column generation technic (Yaghini et al., 2012), local branching based heuristic (Rodríguez-Martín and Salazar-González, 2010), hybrid approach that combines large neighborhood search and IP solver (Hewitt et al., 2010), a matheuristic combining an exact MIP method and a Tabu search metaheuristic (Chouman and Crainic, 2010).

In this paper, we propose a matheuristic for solving the multicommodity capacitated fixed-charge network design problem. The proposed matheuristic is based on adding pseudo-cuts in order to exclude portion of solution space already examined and solving reduced problems deduced from the initial one.

The rest of the paper is organized as follows. In the next section, we give an overview of a convergent algorithm based on the LP-relaxation and pseudo-cuts, and we describe how to convert this algorithm into an heuristic approach. Section 3 contains comparison of the proposed heuristic with the state-of-the-art approaches, while Section 4 concludes the paper.

2. ITERATIVE LINEAR PROGRAMMING-BASED HEURISTIC (ILPH)

In this paper, we develop an algorithm based on the LP-relaxation and pseudo-cuts for solving MCND problem. Those algorithms are based on ideas of solving (optimally or near optimally) a series of small sub-problems obtained from a series of linear programming relaxations within the search for an optimal solution of a given problem. The first algorithm of such type had been proposed in Soyster et al. (1978). About twenty years later, Hanafi and Wilbaut revisited ideas of Soyster et al. and proposed several algorithms for solving multidimensional knapsack problem Hanafi and Wilbaut (2011); Wilbaut and Hanafi (2009). All these papers contain description of exact algorithms along with proofs of their convergence toward to optimal solutions.

Formally, work of such an algorithm may be described in the following way. At each iteration, the LP-relaxation of the current MIP problem P is solved to generate one constraint. Then, a reduced problem induced from an optimal solution of the LP-relaxation is solved to obtain a feasible solution for the initial problem. If the stopping criterion is satisfied, then the best lower bound and the best upper bound are returned. Otherwise, a pseudo cut is added to P and the process is repeated.

Algorithm 1: CALPPC

Function CALPPC(P);

$Q = P$, $v^* = +\infty$;

repeat

  if Q infeasible then break;

  Solve the LP-relaxation of Q to obtain an optimal solution $\bar{y}$ and its objective value $\underline{v}$;

  Solve the reduced problem $P(\bar{y}, J)$;

  Update the best known-value: $v^* = \min\{v^*, v(P(\bar{y}, J))\}$;

  Update the current problem Q by adding the pseudo-cut:

  $Q = (Q|\{\delta(y, \bar{y}, J) \geq 1\})$;

until $v^* - \underline{v} < \epsilon$;

The main idea of the exact algorithm is adding a $pseudo-cut$ at each iteration in order to eliminate reduced problems already examined in the previous solution process. A pseudo-cut consists of linear inequality that excludes certain solutions from being feasible as solutions of the considered problem and it may not be valid in the sense of guaranteeing that at least one globally optimal solution will be retained in the feasible set.

The pseudo-code of a convergent algorithm based on the LP-relaxation and pseudo-cuts (CALPPC) is given in Algorithm 1. At each iteration, the LP-relaxation of the current problem Q is solved to obtain a lower bound $\underline{v}$ and its corresponding optimal solution $\bar{y}$. After that the reduced problem $P(\bar{y}, J)$, defined by chosen subset $J \subset N$, is solved to obtain an upper bound and the best upper bound is updated. The reduced problem $P(\bar{y}, J)$ associated to the solution $\bar{y}$ and subset J is obtained from the problem P by adding the constraints $y_j = \bar{y}_j$ for $j \in J$. Next, the current problem Q is updated by adding the pseudo-cut $\delta(y, \bar{y}, J) = \sum_{j \in J} |y_j - \bar{y}_j| \geq 1$ in order to cut off the region explored by solving the reduced problem. The algorithm stops if the required tolerance between the upper and the lower bounds is reached.

Due to the slow convergence of CALPPC algorithm it cannot be used as an exact algorithm for large instances in practice. In that case, it is preferable to use it as a heuristic approach, imposing some other stopping criterion instead of requiring proof of optimality. The motivation for proposing heuristics based on the CALPPC algorithm stem from the fact that for many instances CALPC algorithm finds an optimal solution quickly but it needs a lot of CPU time to prove its optimality. Additionally, sometimes solving reduced problems optimally is time consuming process itself and therefore it is better to use a heuristic approach for that purpose. All heuristic approaches derived from CALPPC framework we will refer as Iterative Linear Programming-based Heuristics (**ILPH**).

For example, a simplest ILPH approach can be obtained by just stopping the CALPPC algorithm after certain number of iterations or after reaching the predefined CPU time limit. Moreover, the difficulty of resulting reduced problem required to solve in each iteration has big impact on the overall solution process. Hence, sometimes it is more beneficial to impose a CPU time limit for solving a reduced problem instead of solving it to optimality or to additionally reduce size of the reduced problem.