ITM Web of Conferences **14**, 00003 (2017) DOI: 10.1051/itmconf/20171400003
APMOD 2016

Multicommodity Flow Problems with Commodity Compatibility Relations

Zhiyuan Lin[1,⋆] and *Raymond* Kwan[1,⋆⋆]

[1] *School of Computing, University of Leeds, Leeds, United Kingdom, LS2 9PT*

Abstract.
We present a class of Multicommodity Flow Problems with Commodity Compatibility Relations (MCFP-CCR), in which compatibility relations among commodities used at each node are required. This class of problems has application in the Train Unit Scheduling Problem (TUSP) [1, 2], where train units of different traction types may not be coupled with each other to serve the same train trip. Computational complexity issues are discussed and solution methods are proposed. Computational experiments using the proposed solution methods are reported.

1 Introduction

The Multicommodity Flow Problem [3] is characterized by a set of commodities $k \in K, |K| > 1$ to be flowed through a network represented by a graph formed by a node set N and an arc set A as $G = (N, A)$. Each commodity k has a source s_k and a sink t_k. Different types of the problem may differ in features such as minimum supply requirements and/or capacity bounds shared by all commodities on arcs and/or nodes, and other side constraints.

The multicommodity flow problem can be traced back to Ford and Fulkerson [4]. Surveys on multicommdity flow problems can be found in Hu [5], Kennington [6], Ahuja et al. [3]. In general, the continuous Multicommodity Flow Problem is solvable by a polynomial LP solver [3] while its integer version is $\mathcal{NP}$-hard [7, 8].

In a standard multicommodity flow problem, if a nodes and/or an arc is allowed to be flowed by different types of commodities, commodity flows can share the node and/or arc as long as all constraints are satisfied. In this paper, *compatibility relations* are imposed on commodities used at nodes and/or arcs where only certain subsets of them can coexist. For instance, when five commodities k_1, k_2, k_3, k_4, k_5 are allowed at node $j \in N$, commodities used at j should be either from k_1 and k_2 or from k_3, k_4 and k_5, but not otherwise. We refer to this problem as the Multicommodiy Flow Problem with Commodity Compatibility Relations (MCFP-CCR). If the flows are required to be integers, then the corresponding problem is called Integer MCFP-CCR, or IMCFP-CCR. *Families of commodities* are established, where commodities are compatible within the same family. Moreover, families will partition commodities, i.e. no commodity can be compatible with other commodities from two or more families, and a commodity can only belong to one family. Therefore, at most one family of

⋆ e-mail: Z.Lin@leeds.ac.uk
⋆⋆ e-mail: R.S.Kwan@leeds.ac.uk

commodities can be used at a node/arc. In this paper, we focus on the case that the compatibility relations are imposed on nodes on a directed acyclic graph (DAG) as exhibited in the Train Unit Scheduling Problem (TUSP) [1, 2]. Some conclusions are applicable to a general directed graph by analogy.

Given a train operator's timetable and a fleet of train units, the TUSP aims to cover each timetabled train service by a train unit or units. The Train Unit Assignment Problem (TUAP) [9] is also relevant. In [9] and [2], integer multicommodity flow models are used where nodes represent trips to be covered by train units as commodities of different types. A notable feature of the TUSP is that more than one train unit can be coupled to serve the same trip, which gives the requirement on unit type compatibility relations, as often in practice, not any arbitrary train unit types can be coupled.

In [1, 2], in forming feasible train unit schedules, the above unit type compatibility requirement is satisfied by either creating additional variables or by customized branching. We generalize this class of problems as (I)MCFP-CCR. We give complexity analysis and propose recently developed solution methods. Computational experiments will be reported. Conclusion and future research are discussed finally.

2 (I)MCFP-CCR Models

Let $G = (N, A)$ be a DAG with commodities $k \in K, |K| > 1$ partitioned by families $f \in F$ and commodity-graphs $G^k = (N^k, A^k)$ as sub-graphs of G. Let P^k (P^k_j) be the set of commodity k paths (passing through node j). The path formulation of (I)MCFP-CCR with path variables x_p can be "formulated" by the following "(I)LP":

$$(PF) \quad \min \quad \sum_{k\in K} \sum_{p\in P^k} c_p x_p \tag{1}$$

$$\sum_{p\in P^k} x_p = b^k, \quad \forall k \in K \tag{2}$$

$$\sum_{k\in K_j} \sum_{p\in P^k_j} q^k x_p \geq r_j, \quad \forall j \in N \tag{3}$$

$$\sum_{k\in K_j} \sum_{p\in P^k_j} v^k x_p \leq u_j, \quad \forall j \in N \tag{4}$$

$$\text{used commodities at } j \text{ are compatible}, \quad \forall j \in N \tag{5}$$

$$x_p \in \mathbb{R}_+(\mathbb{Z}_+), \quad \forall p \in P^k, \forall k \in K \tag{6}$$

In (1), c_p is the cost for path p in P^k. (2) sets the total demand for each commodity. (3) gives the minimum supply requirement $r_j \geq 0$ where K_j is the set of allowed commodities at j and q^k is the "task contribution" by a unit flow of k. (4) gives the capacity upper bound $u_j \geq 0$, where v^k is the "capacity consumption" by a unit flow of k. "Constraints" (5) are only descriptive. They require used commodities at each node to be compatible. Finally (6) gives the variable domains for the continuous and integer variants respectively.

3 Computational Complexity

Proposition 1. *The IMCFP-CCR is strongly $\mathcal{NP}$-hard.*

Proof. Any TUAP [9] instance is a special case of IMCFP-CCR with only one family of commodities. TUAP is strongly $\mathcal{NP}$-hard [9]. Therefore, IMCFP-CCR is strongly $\mathcal{NP}$-hard. □

Next, it is shown that the decision problem of the MCFP-CCR, denoted as MCFP-CCR(D), is a reduction from an $\mathcal{NP}$-complete problem 3SAT [10]. Suppose there are n boolean variables $X = \{x_1, \ldots, x_n\}$ and m clauses $C_1, \ldots, C_m$, 3SAT asks whether there exists an assignment to X such that $C_1 \wedge \cdots \wedge C_m = \mathrm{T}$, where each clause C_j is a conjunction of three logic variables such that each is either a variable in X or the negation of a variable in X. For example, $C_1 = x_1 \vee x_2 \vee \neg x_3$ is a possible clause.

Proposition 2. *The MCFP-CCR is $\mathcal{NP}$-hard.*

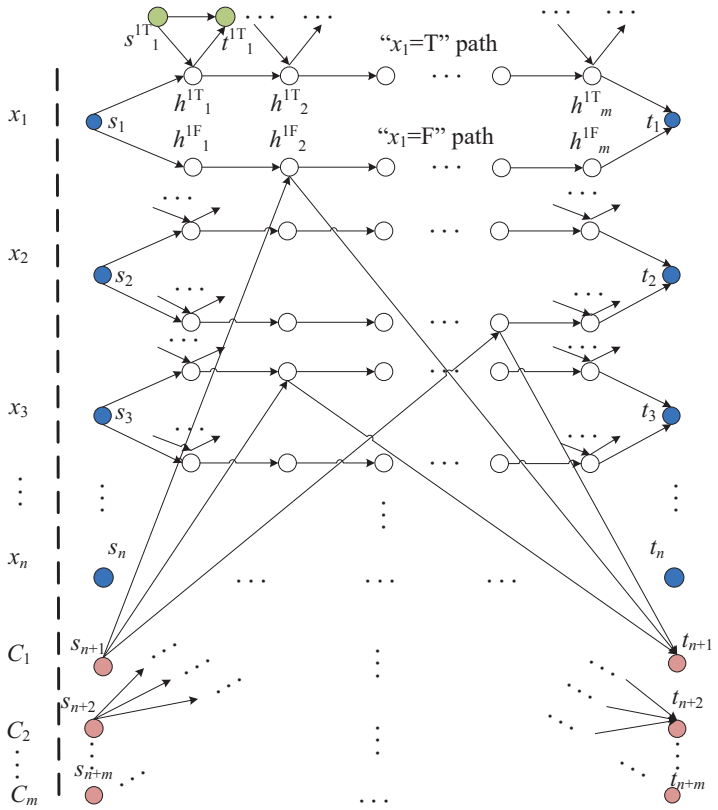

Figure 1: An illustration of an MCFP-CCR(D) instance in the proof, and highlights on dummy sources/sinks wrt node h^{1T}_1 and clause $C_1 = x_1 \vee x_2 \vee \neg x_3$.

Proof. (Sketch) We show that any 3SAT problem can be polynomially reduced to a special MCFP-CCR(D) where each family has only one commodity, as illustrated in Fig. 1. For every boolean variable $x_i \in X$, we create a family source s_i and a family sink t_i, as well as a "true" path and a "false" path each containing m nodes, linked with s_i and t_i. For each of the $2m$ nodes, we add a dummy

family source/sink pair and arcs as shown in Fig. 1. For very clause C_j, $j = 1, \ldots, m$, we create a family source s_{n+j} and a family sink t_{n+j}. Three arcs are created from s_{n+j} and three arcs are linked to t_{n+j} in the following way. If x_i is in C_j as itself, we add an arc from s_{n+j} to a node in the "false" path between s_i and t_i and another arc from this node to t_{n+j}; if x_i is in C_j as its negation, we add an arc from s_{n+j} to a node in the "true" path between s_i and t_i and another arc from this node to t_{n+j}. If more than one clauses use the same variable, their arcs should be connected to distinct nodes in the relevant paths. All family sources/sinks have a demand of 1, and for each internal node its r and u values are also set as 1. It can be verified that any 3SAT problem can be polynomially transformed to an MCFP-CCR(D) as described above. □

4 Solution Approaches: Node-family Branching

In [2] a method is proposed for the TUSP by adding extra binary variables associated for train-family associations. One disadvantage of the above node-family variables is the difficulty in incorporating them into a local convex hull scheme [11]. To resolve this, in [1, 11], an alternative way for ensuring family compatibility by branching is used. We generalize this idea as *node-family branching* and present ongoing research. It is suitable for both IMCFP-CCR and MCFP-CCR.

Let N' be the set of nodes that can be potentially covered by multiple families and let F_j be the set of families allowed at j. Denote the formulation by removing (5) from (PF) as (PF_0). Node-family branching removes incompatible commodities at nodes by branch-and-bound (BB). Suppose linear programming (LP) relaxation is used for getting lower bounds. The solution of the LP relaxation of (PF_0) at the current BB tree node will be checked and either (i) a network node covered by more than one family is identified or (ii) no such a network node can be found and node-family branching is stopped. Suppose a "multi-family" network node $j^* \in N'$ is detected with families $f_1, \ldots, f_r$ covering it, and a family $f^* \in \{f_1, \ldots, f_r\}$ is selected. Then two branches will be formed in the following way: (assuming the left has a higher priority)

- Left branch: only f^* is allowed to cover j^*.
- Right branch: f^* is forbidden to cover j^*.

The following discusses two technical issues regarding node-family branching. The first is on branching variable selection, that is, which (j^*, f^*) pair will be selected. The second is only relevant to the IMCFP-CCR, which investigates how to integrate node-family branching with integer branching.

4.1 Variable selection on (j^*, f^*)

Branch on the Most/Least Diverse Node.

This corresponds to branching on the "most/least fractional" (decimal part closest/farthest to 0.5) schemes used in fractional-to-integer branching. Given an LP relaxation of the tree node n_{BB} to be branched, we define the flow proportion of family f over a network node j as

$$R_{f,j} := \frac{\sum_{k \in f} \sum_{p \in P_j^k} x_p}{\sum_{k \in K_j} \sum_{p \in P_j^k} x_p}. \tag{7}$$

Then the *family diversity* of node j can be defined following Shannon's diversity index [12] that is commonly used as a quantitative measure in reflecting how many and how diverse of different types in a set of collection of them. The family diversity over node j is defined as

$$H_j = -\sum_{f \in F_j} R_{f,j} \ln R_{f,j}. \tag{8}$$

When all used families have the same proportion $\frac{1}{|F_j|}$, the family diversity takes its maximum value $H_j^{\max} = \ln|F_j|$. The more "unequal" the abundances of families, the smaller the value of H_j. When there is only one family, H_j takes its minimum value $H_j^{\min} = 0$. Then j^* can be selected either by $j^* = \arg\max_{H_j>0}(H_j)$, or its opposite $j^* = \arg\min_{H_j>0}(H_j)$. In either case, when j^* is chosen, we choose f^* by $f^* = \arg\max(R_{f,j^*})$.

Branching on Fixed Order.

An ordered list of all nodes in N' is set before the BB process as: $L(N') : \langle j_1, j_2, \ldots, j_{|N'|}\rangle$. The selection can be simply accomplished by finding the first multi-family covered node in the list. It avoids looping over the entire N' at the price of poor flexibility. This naive strategy can be especially useful for DAG, provided the list is constructed according to the "topological sorting" [3] of the nodes. The "single-direction" nature of a DAG suggests that a branching decision made on a node will often have significant impact on its successors.

Branching on Pseudocosts.

Pseudocost branching [13–16] tries to estimate the objective changes caused by branching on potential candidates and chooses the one yielding a largest change, based on empirical evidence that the pseudocost of each variable is relatively "constant". Although without explicit variables, the idea can be applied to node-family branching. Suppose at tree node n, the actual down and up pseudocosts due to branching on (j, f) can be calculated by

$$\psi_{j,f}^{n-} = \frac{z_{j,f}^{n-} - z^n}{R_{j,f}^n}, \quad \psi_{j,f}^{n+} = \frac{z_{j,f}^{n+} - z^n}{1 - R_{j,f}^n}, \tag{9}$$

where $R_{j,f}^n$ is the flow proportion at n as defined in (7), z^n is the objective value of n, and $z_{j,f}^{n-}$ and $z_{j,f}^{n+}$ are the objective values of the right and left branches if (j, f) is chosen. By collecting sufficient information of (j, f) over tree nodes $n_1, ..., n_M$, the "artificial" pseudocosts of (j, f), denoted by $\Psi_{j,f}^-, \Psi_{j,f}^+$, can be estimated from $\psi_{j,f}^{n_1-}, ..., \psi_{j,f}^{n_M-}$ and $\psi_{j,f}^{n_1+}, ..., \psi_{j,f}^{n_M+}$ respectively. Then the objective change at a subsequent tree node n can be estimated by

$$\Delta_{j,f}^{n-} = \Psi_{j,f}^- R_{j,f}^n, \quad \Delta_{j,f}^{n+} = \Psi_{j,f}^+ (1 - R_{j,f}^n). \tag{10}$$

Finally, j^*, f^* can be determined at n by some rules. For example, applying the rules in [15], we have $(j^*, f^*) = \arg\max\{\alpha_1 \min(\Delta_{j,f}^{n-}, \Delta_{j,f}^{n+}) + \alpha_2 \max(\Delta_{j,f}^{n-}, \Delta_{j,f}^{n+})\}$, $\alpha_1 + \alpha_2 = 1, 0 < \alpha_1, \alpha_2 < 1$, with α_1, α_2 set from experiences.

4.2 IMCFP-CCR: Integration with Integer Branching

One way to solve the IMCFP-CCR is to integrate node-family branching with integer branching on the same BB tree. Since there are two branching systems, a priority arrangement has to be set if both of them can be used. We use "nf>int" to denote the case when node-family branching (nf) has priority over integer branching (int). This priority can be updated either in a *static* or *dynamic* way. In either case, an initial priority is set. For the static strategy, the priority remains the same throughout the BB process. For the dynamic strategy, the priority is updated such that if no branching candidate can be found according to the current branching system, the secondary system will take over from the current.

5 Computational Experiments

Real-world datasets based on a UK rail network were used. Two coupling incompatible train unit types, c360 and c321, differing in seat capacities and coupling upper bounds, were allowed to serve all trips. The objective is to minimize the number of used units. Constraints (3) and (4) were replaced by train convex hull constraints [1, 11]. A customized branch-and-price solver was used based the simplex solver of Xpress-MP 7.7. The maximum number of BB tree nodes was set to 5000 and the maximum time was set to 24 hours.

Table 1. Experiments on MCFP-CCR

instance	(j^*, f^*) selection	opt feas obj val	unit # (360/321)	feas sol #	BB#	time (second)	solution quality
	"one-family"	*23*	*8/15*	–	–	*2*	*optimal*
small network	fixed: toplg-sort	23	8/15	3	448	126	optimal
(137 nodes,	fixed: random	fail	–	0	5000	6637	–
2690 arcs)	**most diverse**	**23**	**8/15**	**2**	**101**	**24**	**optimal**
	least diverse	24	13/11	2	5000	11426	sub-opt
	"one-family"	*133*	*33/100*	–	–	*79*	*optimal*
large network	fixed: toplg-sort	134	34/100	5	445	86400	sub-opt
(512 nodes,	fixed: random	fail	–	0	1012	86400	–
23015 arcs)	most diverse	137	37/100	2	910	86400	sub-opt
	least diverse	**133**	**33/100**	**2**	**181**	**592**	**optimal**

Table 1 gives the results for MCFP-CCR on a small and a large network respectively. Four train-family selection strategies and an instance where the two families were set to be compatible ("one-family") were compared. There is evidence that topological sorting based on trains' departure times is advantageous compared with branching on a random order of the trains. Most/least diverse branching outperformed the rest. Table 2 gives the results on experiments for IMCFP-CCR on the small network. Different branching strategies were compared. The initial priority setting can be crucial as all the runs with "int>tf" except one failed. The differences between static and dynamic strategies were not very significant. The "most diverse" train-family selection strategy outperformed the rest. Experiments regarding large scaled network IMCFP-CCR and the selection strategy based on pseudocosts will be conducted in the future.

Acknowledgments. This research is supported by an EPSRC project EP/M007243/1. We would like to also thank FirstGroup for their collaboration.
Data Statement FirstGroup has provided relevant data for the research, part of which is commercially sensitive. Where possible, the data that can be made publicly available is deposited in `http://archive.researchdata.leeds.ac.uk/`.

References

[1] Z. Lin, R.S.K. Kwan, Electron. Notes Discrete Math **41**, 165 (2013)
[2] Z. Lin, R.S.K. Kwan, Public Transport **6**, 35 (2014)
[3] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network flows: theory, algorithms and applications* (Prentice Hall, Englewoods Cliffs, USA, 1993)
[4] L.R. Ford, D.R. Fulkerson, Manag. Sci. **5**, 97 (1958)

Table 2. Experiments on IMCFP-CCR: small network (137 nodes, 2690 arcs)

initial priority	(j^*, f^*) selection	update strategy	opt feas obj val	unit # (360/321)	feas sol #	BB#	time (second)	rule % (tf/int)	solution quality
"one-family"	–	–	*23*	*9/14*	–	*169*	*58*	–	*optimal*
tf>int	fixed: toplg-sort	dynamic	23	8/15	3	1441	858	94/6	optimal
tf>int	fixed: toplg-sort	static	23	8/15	3	1449	873	94/6	optimal
tf>int	fixed: random	dynamic	26	11/15	1	5000	8000	100/0	sub-opt
tf>int	fixed: random	static	24	9/15	3	5000	24268	99/1	sub-opt
tf>int	most diverse	dynamic	23	8/15	2	585	426	75/25	optimal
tf>int	**most diverse**	**static**	**23**	**8/15**	**2**	**197**	**127**	**50/50**	**optimal**
tf>int	least diverse	dynamic	24	13/11	2	5000	20962	85/15	sub-opt
tf>int	least diverse	static	24	13/11	2	5000	11901	98/2	sub-opt
int>tf	fixed: toplg-sort	dynamic	fail	–	0	5000	9217	4/96	–
int>tf	fixed: toplg-sort	static	fail	–	0	5000	5991	6/94	–
int>tf	fixed: random	dynamic	fail	–	0	5000	9458	97/3	–
int>tf	fixed: random	static	fail	–	0	5000	9005	8/92	–
int>tf	most diverse	dynamic	fail	–	0	5000	14823	97/3	–
int>tf	most diverse	static	fail	–	0	5000	14406	9/91	–
int>tf	least diverse	dynamic	25	10/15	1	5000	4237	96/4	sub-opt
int>tf	least diverse	static	fail	–	0	5000	10612	4/96	–

[5] T.C. Hu, Oper. Res. **11**, 344 (1963)
[6] J.L. Kennington, Oper. Res. **26**, 209 (1978)
[7] R.M. Karp, Networks **5**, 45 (1975)
[8] S. Even, A. Itai, A. Shamir, SIAM J. Comput. **5**, 691 (1976)
[9] V. Cacchiani, A. Caprara, P. Toth, Math. Prog. B **124**, 207 (2010)
[10] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, NY, USA, 1979), ISBN 0716710447
[11] Z. Lin, R.S.K. Kwan, Comput. Optim. Appl. pp. 1–39 (2016)
[12] C.E. Shannon, Bell System Technical Journal **27**, 379 (1948)
[13] M. Benichou, J.M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, O. Vincent, Math. Prog. **1**, 76 (1971)
[14] J.M. Gauthier, G. Ribière, Math. Prog. **12**, 26 (1977)
[15] J.T. Linderoth, M.W.P. Savelsbergh, INFORMS J. on Comput **11**, 173 (1999)
[16] T. Achterberg, T. Koch, A. Martin, Oper. Res. Letters **33**, 42 (2005)