European Journal of Operational Research 237 (2014) 814–823

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete optimization

The Red–Blue transportation problem

Wim Vancroonenburg[a,*], Federico Della Croce[b,c], Dries Goossens[d,e], Frits C.R. Spieksma[e]

[a] KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC, Gebroeders De Smetstraat 1, BE-9000 Gent, Belgium
[b] Politecnico di Torino, D.A.I., Torino, Italy
[c] CNR, IEIIT, Torino, Italy
[d] Ghent University, Faculty of Economics and Business Administration, Tweekerkenstraat 2, BE-9000 Gent, Belgium
[e] KU Leuven, Faculty of Economics and Business, ORSTAT, Naamsestraat 69, BE-3000 Leuven, Belgium

ARTICLE INFO

Article history:
Received 11 April 2013
Accepted 22 February 2014
Available online 5 March 2014

Keywords:
Transportation problem
Exclusionary constraints
Complexity
Approximation
Integer programming

ABSTRACT

This paper considers the Red–Blue Transportation Problem (Red–Blue TP), a generalization of the transportation problem where supply nodes are partitioned into two sets and so-called exclusionary constraints are imposed. We encountered a special case of this problem in a hospital context, where patients need to be assigned to rooms. We establish the problem's complexity, and we compare two integer programming formulations. Furthermore, a maximization variant of Red–Blue TP is presented, for which we propose a constant-factor approximation algorithm. We conclude with a computational study on the performance of the integer programming formulations and the approximation algorithms, by varying the problem size, the partitioning of the supply nodes, and the density of the problem.

1. Introduction

Consider the well-known Transportation Problem (TP): given is a set of supply nodes S, each with supply a_i $(i \in S)$, a set of demand nodes D, each with demand b_j $(j \in D)$, with $\sum_{i \in S} a_i = \sum_{j \in D} b_j$, and a bipartite graph $(S \cup D, E)$, with a given cost c_{ij} for each edge $(i,j) \in E$, where E is not necessarily complete. The question is how to send the flow from supply nodes to the demand nodes such that total cost is minimal. In this paper, we generalize this problem by associating a color, either red or blue, to each supply node. Thus, the set of supply nodes is partitioned into two sets R (red) and B (blue) such that $S = R \cup B$, and $R \cap B = \emptyset$. The additional requirement is that the set of supply nodes that actually supply a demand node should all have the same color. In other words, a demand node is only allowed to receive flow from supply nodes that are either all red or all blue. We refer to these constraints as color constraints. Obviously, the resulting problem is a generalization of the transportation problem since if all supply nodes have the same color, the TP arises. We will refer to our problem as the Red–Blue Transportation Problem (Red–Blue TP).

In Section 1.1 we discuss the practical application that motivated our study, followed by related literature in Section 1.2.

1.1. Motivation

Although the Red–Blue TP may seem a purely theoretical generalization of the transportation problem, its motivation stems from a situation we encountered in practice. Consider a setting where patients in a hospital need to be assigned to rooms. Rooms are of limited capacity, and due to specific equipment, not all rooms are equally appropriate for each patient. For instance, a patient's pathology may require oxygen to be available at the room; rooms that do not meet this requirement, need to be equipped with a mobile oxygen supply which is, for organizational reasons, less desirable. The Patient Admission Scheduling problem (PAS) consists in assigning patients to rooms in such a way that the medical concerns and personal wishes are fulfilled as much as possible. This problem has been defined by Demeester, Souffriau, De Causmaecker, and Vanden Berghe (2010), and studied further by Ceschia and Schaerf (2011). A key constraint in the patient admission scheduling problem is that male and female patients should not be assigned to the same room, which is common practice in hospitals (all over the world). Clearly, this situation can be modeled as a (special case of) Red–Blue TP: each patient is represented as a supply node with $a_i = 1$, each room is represented as a demand node where the capacity of the room is represented by b_j, and the "appropriateness" of assigning patient i to room j is captured by cost c_{ij}.

It is not hard to think of other practical applications of Red–Blue TP. For instance, imagine a situation where a number of goods need to be transported from a port to a warehouse. Several trucks are

* Corresponding author. Tel.: +32 92658704.
E-mail address: wim.vancroonenburg@cs.kuleuven.be (W. Vancroonenburg).

http://dx.doi.org/10.1016/j.ejor.2014.02.055
0377-2217/© 2014 Elsevier B.V. All rights reserved.

available for transportation, each driving according to a schedule that fixes the departure times at the port. Depending on delivery deadlines, some truck assignments are more suitable for particular goods than others. A Red–Blue TP instance arises if the goods can be divided into two types that cannot be assigned to the same truck, for instance because of incompatibilities of content (e.g. hazardous materials), ownership (e.g. rivaling business companies that are unwilling to have their goods transported on the same truck), or size (Cao & Uebe, 1995). Another application involves the transportation of football fans to the match using public railways: when assigning fans to trains, no fans of the opposing teams should be on the same train, to avoid hooliganism (Schreuder, 1992). In Section 4, we mention further applications of a maximization variant of our problem.

1.2. Related literature

The Red–Blue transportation problem is a natural generalization of a classic problem in operations research. In the literature, several generalizations of the transportation problem have been described. The most well-known is probably the transshipment problem, in which the underlying graph need not be bipartite, and so-called *transferring* nodes, which have no net supply or demand, may exist (see e.g. Orden (1956)). The min-cost flow problem is a further generalization of the transshipment problem, introducing capacities on the arcs. In the fixed-charge transportation problem (Hirsch & Dantzig, 1968), a fixed cost may be incurred for every arc in the transportation network that is used. Numerous other generalizations of the transportation problem have been presented, for instance to solve spatial economic equilibrium problems (MacKinnon, 1975), and aircraft routing problems (Ferguson & Dantzig, 1955), or even to deal with wartime conditions where distances from some sources to some destinations are no longer definite (i.e. the *gray* transportation problem, see Bai, Mao, & Lu (2004)).

One generalized transportation problem is particularly related to the Red–Blue transportation problem, namely the Transportation Problem with Exclusionary Side Constraints (TPESC). Although the name TPESC was coined by Sun (2002), it was in fact introduced by Cao (1992). The phenomenon that not every set of supply nodes is allowed to send flow to a demand node, is something that TPESC and Red–Blue TP have in common. In TPESC, for each demand node $j \in D$, a set of pairs of supply nodes is given, denoted by $F_j = \{\{i_1, i_2\} | i_1, i_2 \in S\}$. The problem is to send the flow from supply to demand nodes at minimum cost, such that each demand node $j \in D$ only receives supply from at most one supply node for each pair of supply nodes present in F_j.

It is not hard to see that Red–Blue TP is a special case of TPESC. Goossens and Spieksma (2009) show that TPESC is NP-hard, and becomes pseudo-polynomially solvable if the number of supply nodes is fixed. Furthermore, these authors study TPESC with identical exclusionary sets: they provide a pseudo-polynomial algorithm for the case with two demand nodes, and prove NP-hardness for the case with three demand nodes.

Another problem related to Red–Blue TP is the so-called Maximum Flow problem with Conflict Graph (MFCG), a problem studied by Pferschy and Schauer (2013). In the MFCG a directed graph with capacitated arcs, a source, and a sink are given. In addition, pairs of arcs (from the directed graph) are given; for some pairs of arcs the constraint is that at most one arc of the pair can carry flow (a negative disjunctive constraint), for other pairs of arcs the constraint is that at least one arc of the pair must carry flow (a positive disjunctive constraint). Pferschy and Schauer (2013) show that the problem of finding a maximum flow in a network under these disjunctive constraints is (strongly) NP-hard; even more they show that no polynomial time constant-factor approximation algorithm can exist (unless P = NP).

Observe that Red–Blue TP is a special case of MFCG; indeed, consider some demand $j \in D$. Now, by having negative disjunctive constraints for each pair of arcs that consist of one arc emanating from a red supply node to node j, and one arc emanating from a blue supply node to node j, an instance of Red–Blue TP arises. We point out that for our special case it is possible to find polynomial time constant factor approximation algorithms (see Section 4).

2. Complexity of Red–Blue TP

As a general statement of the complexity of Red–Blue TP, we provide the following theorem.

Theorem 1. *Red–Blue TP is NP-hard, even if* $a_i = 1\ \forall i \in S$, *and* $b_j = 3\ \forall j \in D$.

Proof. We prove Theorem 1 by showing that the EXACT-3-COVER (X3C) problem can be reduced to the decision version of Red–Blue TP. The decision version of Red–Blue TP, denoted Red–Blue TP$_D$, concerns the question: does there exist a solution that sends all flow from the supply nodes to the demand nodes while satisfying demand, and while satisfying the color constraints, i.e. does there exist a feasible flow? X3C has been shown to be NP-complete (see e.g., Garey & Johnson, 1979), and is defined as follows:

> **Input:** A set X with $|X| = 3q$ and a collection C of 3-element subsets (i.e., triples) of X, with $|C| = k$.
> **Question:** Does there exist a cover in C that covers X exactly, i.e. a subcollection $C' \subseteq C$ such that every $x_i \in X$ is contained in exactly one $C_j \in C'$?

Any instance of X3C (with $|C| > q$) can be reduced to Red–Blue TP$_D$ as follows. Associate to each element $x_i \in X$ a blue supply node i with $a_i = 1$. Associate to each triple C_j a demand node j with $b_j = 3$. Create edges from supply to demand nodes corresponding to the membership relations (i.e. supply node x_i is connected to demand node $C_j \iff x_i \in C_j$). Add $3(k - q)$ red supply nodes with $a_i = 1$ that are connected to all demand nodes. Observe that total supply equals total demand. The question is: does there exist a feasible flow in this instance of Red–Blue TP$_D$?

Now we show that a yes-answer to the X3C instance directly corresponds to a yes-answer to the corresponding Red–Blue TP instance, and vice versa.

First, consider an X3C instance that is feasible, and thus has an exact cover $C' \subseteq C$. Then, each demand node corresponding to a $C_j \in C'$ can be supplied by the blue supply nodes corresponding to the $x_i \in C_j$, and the remaining demand nodes can be supplied by the red supply nodes. Thus, the corresponding Red–Blue TP$_D$ instance is also feasible.

Next, consider any feasible solution to the Red–Blue TP$_D$ instance. Each demand node is supplied by either three red supply nodes or by three blue supply nodes. Moreover, there must exist q demand nodes each supplied by three blue supply nodes. These triples of blue supply nodes correspond to the triples in X3C that form a feasible solution. □

Notice that the above reduction can be generalized to show that Red–Blue TP with $b_j = k$ is at least as hard as Exact Cover by k-sets.

If we put a cost of zero on the edges described in the above proof, and add some edges with a cost strictly larger than zero (corresponding to $x_i \notin C_j$), a polynomial-time algorithm with a constant performance ratio for Red–Blue TP would find a zero cost solution if one exists, and hence would be able to distinguish between the yes-instances and the no-instances of X3C. Therefore, the following corollary holds:

Corollary 1. *There is no constant-factor approximation algorithm for Red–Blue TP, even if* $a_i = 1\ \forall i \in S$, *and* $b_j = 3\ \forall j \in D$, *unless* $P = NP$.

Since the problem setting where, in addition to $a_i = 1$, also $b_j = 1$ reduces to the assignment problem, the only setting for which the complexity is left open is the case when $b_j = 2$. The special case of Red–Blue TP on a complete bipartite graph also has relevance. Relating back to the practical application of assigning patients to hospital rooms, patients can also be assigned to unsuitable rooms if necessary, when all other rooms are at capacity. Therefore, the assignment graph is, in this case, complete. The following theorem shows that this does not make the problem easy, even when all edge-costs are equal.

Theorem 2. *Red–Blue TP is NP-hard, even if G is a complete bipartite graph, all edge-costs are equal and there are only 2 supply nodes with equal supply.*

Proof. We prove Theorem 2 by showing that PARTITION can be reduced to Red–Blue TP$_D$. PARTITION has been shown to be NP-Complete (see e.g., Garey & Johnson, 1979) and is defined as follows:

Input: A set of integers $X = \{x_1, x_2, \ldots, x_n\}$ with $\sum_{i=1}^{n} x_i = q$
Question: Does there exist a partition of X into $\{X_1, X_2\}$ such that

$$\sum_{x_i \in X_1} x_i = \sum_{x_j \in X_2} x_j$$

The reduction is as follows. Create a demand node for each $x_i \in X$ with $b_j = x_i$. Next, create a blue and a red supply node, each with a supply of $\frac{q}{2}$. Create edges between each supply/demand node pair so that the resulting bipartite graph is complete. Observe that total supply equals total demand by construction. The question is: does there exist a feasible assignment?

Consider a feasible PARTITION instance with a partition $\{X_1, X_2\}$ of X such that $\sum_{x_i \in X_1} x_i = \sum_{x_j \in X_2} x_j = \frac{q}{2}$. It is clear that the corresponding Red–Blue TP$_D$ instance is also feasible by supplying each demand node $x_i \in X_1$ by the blue node and each demand node $x_j \in X_2$ by the red one.

Next, consider that the Red–Blue TP$_D$ instance is feasible. In any feasible solution, a demand node $x \in X$ will be entirely supplied by either the blue supply node or the red supply node. Since total supply equals total demand, it must be so that in any feasible solution the sum of the demand nodes supplied by blue (red) supply nodes is equal to $\frac{q}{2}$. Therefore, it must be so that when:

$X_1 = \{x \in X \mid$corresponding demand node is supplied by the *blue* supply node$\}$

$X_2 = \{x \in X \mid$corresponding demand node is supplied by the *red* supply node$\}$

then:

$$\sum_{x_i \in X_1} x_i = \sum_{x_j \in X_2} x_j = \frac{q}{2}$$

Thus the corresponding PARTITION instance is also feasible. □

Finally, consider the special case of Red–Blue TP where the number of demand nodes is fixed, but the capacity of the demand nodes is still part of the input. In this case, we provide the following lemma:

Lemma 1. *If* $|D|$ *is fixed, Red–Blue TP is polynomially solvable.*

Proof. Red–Blue TP can be seen as a coloring problem, where the demand nodes are to be colored either blue or red in such a way that all blue (red) supply nodes can be assigned to blue (red) demand nodes. Given a coloring of demand nodes, the feasibility of Red–Blue TP can be determined by solving two transportation problems: the TP on the blue subgraph, and the problem on the red subgraph. If there are $|D|$ demand nodes, then there are $2^{|D|}$ possible colorings of the demand nodes. Thus by solving $2 \times 2^{|D|}$ transportation problems, the feasibility of Red–Blue TP can be determined. Moreover, if a feasible solution exists, this algorithm will find an optimal solution. Since the TP can be solved in polynomial time, this enumeration can be done in time polynomial in the number of supply nodes. □

3. Integer models

In this section we provide two integer programming formulations for Red–Blue TP and show that Formulation 2 is strictly stronger than Formulation 1. We refer to Table 1 for details on the notation.

Formulation 1 The decision variables are defined as follows:

$$x_{ij} = \text{amount of supply that node } i \text{ sends to node } j$$

$$y_j = \begin{cases} 1 & \text{if demand node } j \text{ is supplied by red nodes,} \\ 0 & \text{otherwise} \end{cases}$$

Red–Blue TP is modeled as follows:

$$\mathbf{Min} \sum_{i \in S} \sum_{j \in D_i} c_{ij} \cdot x_{ij} \tag{1}$$

subject to:

$$\sum_{j \in D_i} x_{ij} = a_i \quad \forall i \in S \tag{2}$$

$$\sum_{i \in R_j} x_{ij} = b_j \cdot y_j \quad \forall j \in D \tag{3}$$

$$\sum_{i \in B_j} x_{ij} = b_j \cdot (1 - y_j) \quad \forall j \in D \tag{4}$$

$$x_{ij} \in \mathbb{N} \quad \forall i \in S, j \in D_i \tag{5}$$

$$y_j \in \{0, 1\} \quad \forall j \in D \tag{6}$$

The objective function minimizes the total cost of sending supply to demand nodes j. Constraints (2) ensure that each supply node i sends its supply a_i to its appropriate demand nodes D_i. Constraints (3) and (4) ensure that each demand node j receives b_j units of supply from either red or blue supply nodes. The decision variables x_{ij} are defined for all feasible (i,j) pairs in Expression (5). The LP-relaxation of (1)–(6) arises when we replace (5) and (6) by $x_{ij} \geqslant 0, \forall i \in S, j \in D_i$, and $0 \leqslant y_j \leqslant 1, \forall j \in D$. The corresponding objective function value is denoted by V_{LP1}.

Formulation 2 The second formulation corresponds to the integer model described by Sun (2002). It uses the same decision variables x_{ij} as Formulation 1, but also uses decision variables y_{ij} that are defined as follows:

Table 1
Notation for describing the Red–Blue TP formulations.

Notation	Description
S	Set of supply nodes
D	Set of demand nodes
S_j	Set of supply nodes that can supply demand node $j \in D$
D_i	Set of demand nodes that can be supplied by supply node $i \in S$
a_i	Supply of node $i \in S$
b_j	Demand of node $j \in D$
R (B)	Set of red (blue) supply nodes, $S = R \cup B$
R_j (B_j)	Set of red (blue) nodes that can supply demand node $j \in D$
c_{ij}	The cost of sending one unit of supply from node i to demand node j

$$y_{ij} = \begin{cases} 1 & \text{if node } i \text{ supplies node } j, \\ 0 & \text{otherwise} \end{cases}$$

Red–Blue TP is modeled as follows:

$$\mathbf{Min} \sum_{i \in S} \sum_{j \in D_i} c_{ij} \cdot x_{ij} \tag{7}$$

subject to:

$$\sum_{j \in D_i} x_{ij} = a_i \quad \forall i \in S \tag{8}$$

$$\sum_{i \in S_j} x_{ij} = b_j \quad \forall j \in D \tag{9}$$

$$x_{ij} \leqslant \min(a_i, b_j) \cdot y_{ij} \quad \forall i \in S, j \in D_i \tag{10}$$

$$y_{ij} + y_{kj} \leqslant 1 \quad \forall i \in B, k \in R, j \in D_i \cap D_k \tag{11}$$

$$x_{ij} \in \mathbb{N}, y_{ij} \in \{0,1\} \quad \forall i \in S, j \in D_i \tag{12}$$

The objective function is the same as in Formulation 1, minimizing total cost. Constraints (8) ensure that each supply node i sends its supply a_i to its appropriate demand nodes D_i, while constraint (9) ensures that b_j units of supply are sent to demand node j. Constraints (10) express that y_{ij} takes the value 1, when $x_{ij} > 0$. Constraints (11) ensure that no red and blue supply node i and k supply the same demand node j.

The LP-relaxation of (7)–(12) arises when we replace (12) by $x_{ij} \geqslant 0, 0 \leqslant y_{ij} \leqslant 1, \forall i \in S, j \in D_i$. The corresponding objective function value is denoted by V_{LP2}.

Theorem 3. *$V_{LP1} \leqslant V_{LP2}$, namely Formulation 2 is strictly stronger than Formulation 1.*

Proof. For any instance, take an optimal LP-solution of Formulation 2 $(\mathbf{x_2}, \mathbf{y_2})$ with its value denoted V_{LP2}. A feasible LP-solution of Formulation 1 can be constructed by setting $\mathbf{x_1} = \mathbf{x_2}$ and $y_j = \sum_{i \in R_j} x_{ij}/b_j, \forall j \in D$. It is easy to verify that this is a feasible solution for the LP-relaxation of (1)–(6) with value V_{LP2}.

Consider the following example showing that Formulation 2 can be better than Formulation 1. Given the complete bipartite graph $G(S \cup D, E)$ with $S = R \cup B, R = \{i_1, i_2\}, B = \{i_3\}, D = \{j_1, j_2, j_3\}$ and $E = S \times D$. Also, $a_{i_1} = a_{i_3} = 2, a_{i_2} = 1, b_{j_1} = b_{j_3} = 1$, and $b_{j_2} = 3$. All drawn edges in Fig. 1 have $c_{ij} = 0$, all other edges (not drawn) have $c_{ij} = 1$. It is a fact that the LP-relaxation of Formulation 1 has an optimal value $V_{LP1} = 0$ (Fig. 2(a)), whereas the LP-relaxation of Formulation 2 has an optimal value $V_{LP2} = 1$ (Fig. 2(b)). Thus, there are instances for which Formulation 2 is better than Formulation 1. $\square$

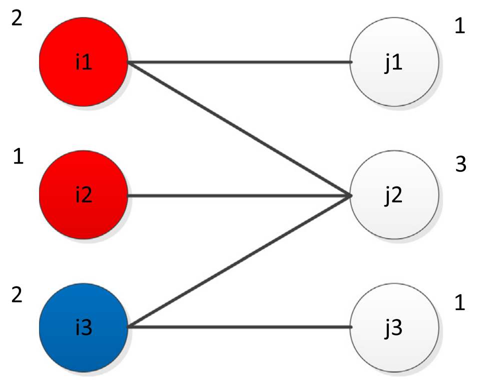

Fig. 1. A complete bipartite graph with $R = \{i_1, i_2\}$ and $B = \{i_3\}; a_{i_1} = 2, a_{i_2} = 1, a_{i_3} = 2, b_{j_1} = 1, b_{j_2} = 3$, and $b_{j_3} = 1$. Drawn edges have cost $c_{ij} = 0$, other (not drawn) edges have $c_{ij} = 1$.

Notice that the example in Fig. 1 also shows that the LP-relaxation of Formulation 1 can be arbitrarily bad compared to the integer optimum. That fact is also true for Formulation 2: in Fig. 3(a) we give an instance with $R = \{i_1\}, B = \{i_2\}$ where all drawn edges have cost $c_{ij} = 0$, all other edges (not drawn) have $c_{ij} = 1$. Also, $a_{i_1} = a_{i_2} = 2, b_{j_1} = b_{j_3} = 1$, and $b_{j_2} = 2$. This instance has no integer solution with value 0, whereas the LP-relaxation of Formulation 2 allows a solution with value $V_{LP2} = 0$ (Fig. 3(b)).

4. The maximization variant of Red–Blue TP

In this section, we consider the following variant of Red–Blue TP: we modify the objective function to maximization, with p_{ij} denoting the profit gained from supplying one unit to demand node j from supply node i. Moreover, we do not insist on sending all flow. We refer to this variant as Max-Red–Blue TP. In this setting, the patient admission scheduling problem as described in Section 1.1 can be seen as assigning as many patients as possible (weighted by e.g. their need for treatment) to rooms. Another example is assigning students to trainers in sport classes, where students may have preferences for trainers (or the corresponding timetable), taking into account the constraint that boys and girls should not be together in the same class.

It is not difficult to see that Max-Red–Blue TP remains NP-hard. Indeed, this result follows from a minor adaptation of the proof of Theorem 1.

An IP-formulation of the maximization variant (corresponding to Formulation 1) is as follows:

$$\mathbf{Max} \sum_{i \in S} \sum_{j \in D_i} p_{ij} \cdot x_{ij} \tag{13}$$

subject to:

$$\sum_{j \in D_i} x_{ij} \leqslant a_i \quad \forall i \in S \tag{14}$$

$$\sum_{i \in R_j} x_{ij} \leqslant b_j \cdot y_j \quad \forall j \in D \tag{15}$$

$$\sum_{i \in B_j} x_{ij} \leqslant b_j \cdot (1 - y_j) \quad \forall j \in D \tag{16}$$

$$x_{ij} \in \mathbb{N} \quad \forall i \in S, j \in D_i \tag{17}$$

$$y_j \in \{0,1\} \quad \forall j \in D \tag{18}$$

Observe that the LP-relaxation of (13)–(18) can be found by solving a transportation problem, consisting of (13) and (14), $\sum_{i \in R_j \cup B_j} x_{ij} \leqslant b_j, \forall j \in D$, and $x_{ij} \geqslant 0, \forall i \in S, j \in D_i$. We denote the value of this formulation by V_{LP1}^{max}. This formulation only uses the x-variables; a feasible solution for the y-variables is given by $y_j = \sum_{i \in R_j} x_{ij}/b_j, \forall j \in D$.

The Max-Red–Blue TP setting is interesting to study with respect to approximation, as it does not suffer from the feasibility issue. Indeed, in this setting the **0**-vector is always a feasible, be it the worst possible, solution. In the following subsections, we present three approximation algorithms for Max-Red–Blue TP; the final subsection deals with a number of generalizations of Max-Red–Blue TP.

4.1. A first algorithm: MAX-RB

Consider the algorithm described below. It consists of solving two transportation problems, and next selecting the best of the two corresponding solutions.

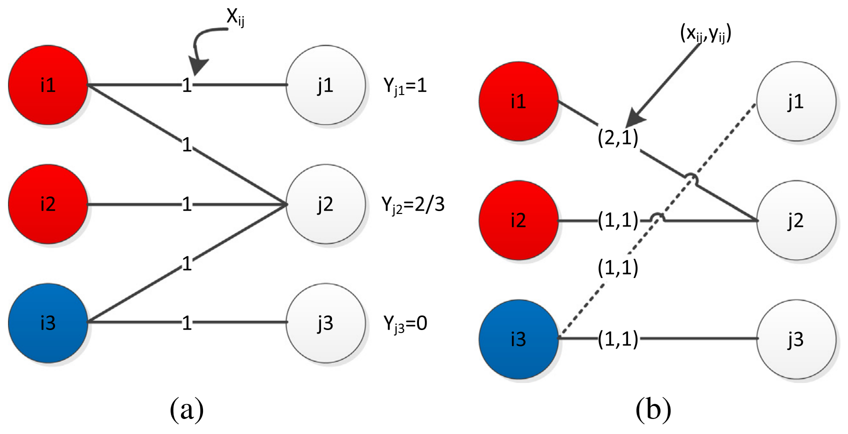

Fig. 2. Relaxation of Formulation 1 (left) and Formulation 2 (right). The dashed edge shows that the LP-relaxation of Formulation 2 uses an edge with cost $c_{i_3,j_1} = 1$, resulting in a LP-relaxation value $V_{LP2} = 1$.

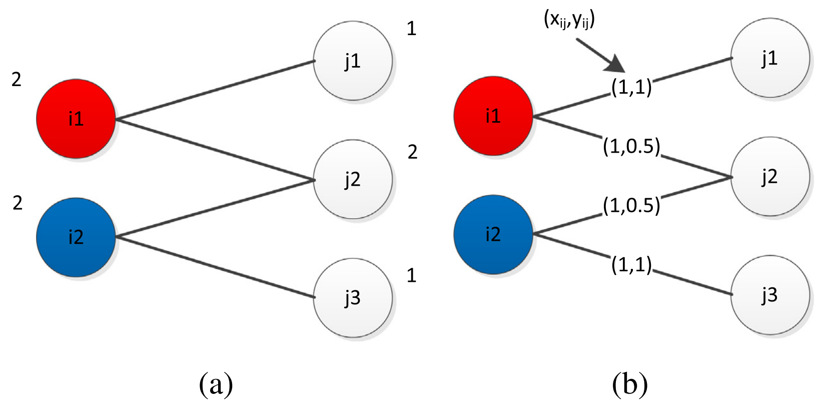

Fig. 3. Example on which Formulation 2 performs arbitrarily bad. A complete bipartite graph, with $R = \{i_1\}$ and $B = \{i_2\}$; $a_{i_1} = a_{i_2} = 2, b_{j_1} = b_{j_3} = 1$, and $b_{j_2} = 2$. Drawn edges have cost $c_{ij} = 0$, all other edges (not drawn) have $c_{ij} = 1$. An integer optimum with value 0 does not exist, however, the LP-relaxation of Formulation 2 has an optimal value of $V_{LP2} = 0$ (right).

Algorithm 1. MAX-RB

```
1: Solve a TP on the subgraph induced by $R \cup D$ (the red
   subgraph), and solve a TP on the subgraph induced by $B \cup D$
   (the blue subgraph). The respective solution values are
   denoted by $V(R)$ and $V(B)$.
2: Return the solution vector for which $\max(V(R), V(B))$ is
   attained.
```

Despite its simplicity, the resulting solution vector cannot be arbitrarily bad. We use OPT for the value of an optimal solution to the Max-Red–Blue TP problem.

Theorem 4. *The approximation ratio of MAX-RB is $\frac{1}{2}$ and this bound is tight.*

Proof.

$$\max(V(R), V(B)) \geqslant \frac{1}{2}(V(R) + V(B)) \tag{19}$$

$$\geqslant \frac{1}{2}OPT. \tag{20}$$

The first inequality is trivial, the second follows from the observation that $V(R) + V(B)$ is the value of an optimal solution to a relaxed version of our problem, namely one where the demands are doubled (i.e. $b_i := 2b_i$), and where no color constraints are present.

Finally, we show that the bound is tight. Consider the example in Fig. 4. For this instance, MAX-RB may find $V(R) = V(B) = 2$. However, clearly $OPT = 4$. □

4.2. A second algorithm: TP + R

The algorithm described below consists of solving three transportation problems, one of which corresponds to solving the LP-relaxation of (13)–(18).

Algorithm 2. TP + R

```
1: Solve the LP-relaxation of (13)–(18). Call the resulting
   solution vector $x^*_{ij}$. Set $D_R = D_B = \emptyset$.
2: for all $j \in D$ do
3:   if $\sum_{i\in R_j} p_{ij}x^*_{ij} \geqslant \sum_{i\in B_j} p_{ij}x^*_{ij}$ then
4:     $D_R := D_R \cup \{j\}$
5:   else
6:     $D_B := D_B \cup \{j\}$
7:   end if
8: end for
9: Solve two TPs, one on the subgraph induced by $R \cup D_R$, and
   one on $B \cup D_B$ and construct the overall solution with value
   $V(TP + R)$.
```

Theorem 5. *The approximation ratio of TP + R is $\frac{1}{2}$ and this bound is tight.*

Proof. Let $V(B \cup D_B)$ and $V(R \cup D_R)$ denote the value of solving the TP on the subgraph induced by $B \cup D_B$ and on the subgraph induced by $R \cup D_R$ respectively. Then:

$$\begin{aligned} V(TP+R) &= V(B \cup D_B) + V(R \cup D_R) \\ &\geqslant \sum_{j \in D_B} \sum_{i \in B_j} p_{ij} x^*_{ij} + \sum_{j \in D_R} \sum_{i \in R_j} p_{ij} x^*_{ij} \end{aligned} \tag{21}$$

This inequality holds since x^*_{ij} restricted to $j \in D_B$, $i \in B_j$ (or $j \in D_R, i \in R_j$) is a feasible solution to the transportation problem solved in line 9 of Algorithm 2. Hence, an optimal solution to that TP has a value at least as large as $\sum_{j \in D_B} \sum_{i \in B_j} p_{ij} x^*_{ij}$.

Define $v_j = \sum_{i \in S_j} p_{ij} x^*_{ij}$ for each $j \in D$. Observe that $V^{max}_{LP1} = \sum_{j \in D} v_j$. By construction of the sets D_R and D_B, we have for each $j \in D_R$:

$$\sum_{i \in R_j} p_{ij} x^*_{ij} \geqslant \frac{1}{2} \left(\sum_{i \in R_j} p_{ij} x^*_{ij} + \sum_{i \in B_j} p_{ij} x^*_{ij} \right) = \frac{1}{2} v_j \tag{22}$$

and for each $j \in D_B$:

$$\sum_{i \in B_j} p_{ij} x^*_{ij} \geqslant \frac{1}{2} \left(\sum_{i \in R_j} p_{ij} x^*_{ij} + \sum_{i \in B_j} p_{ij} x^*_{ij} \right) = \frac{1}{2} v_j \tag{23}$$

Thus:

$$\begin{aligned} \sum_{j \in D_B} \sum_{i \in B_j} p_{ij} x^*_{ij} + \sum_{j \in D_R} \sum_{i \in R_j} p_{ij} x^*_{ij} &\geqslant \sum_{j \in D_B} \frac{1}{2} v_j + \sum_{j \in D_R} \frac{1}{2} v_j = \frac{1}{2} \sum_{j \in D} v_j \\ &= \frac{1}{2} V^{max}_{LP1} \geqslant \frac{1}{2} OPT. \end{aligned} \tag{24}$$

Finally, we show that the bound is tight. Consider again the example in Fig. 4. In this case, the worst case optimal solution vector to the TP (note that there are several optimal solution vectors) is:

$$x^*_{12} = x^*_{21} = x^*_{32} = x^*_{41} = 1, \quad x^*_{11} = x^*_{22} = x^*_{31} = x^*_{42} = 0. \tag{25}$$

Thus, we get:

$$\sum_{i \in R_1} p_{i1} x^*_{i1} = \sum_{i \in B_1} p_{i1} x^*_{i1} = 1, \quad \text{and} \tag{26}$$

$$\sum_{i \in R_2} p_{i2} x^*_{i2} = \sum_{i \in B_2} p_{i2} x^*_{i2} = 1. \tag{27}$$

Therefore, the worst case coloring is:

$$D_R = \{1, 2\}, \quad D_B = \emptyset. \tag{28}$$

Thus, solving the TPs on the subgraphs leads to:

$$V(TP+R) = V(B \cup D_B) + V(R \cup D_R) = 0 + 2 = 2. \tag{29}$$

Recall from the previous that $OPT = 4$. □

Corollary 2. $V^{max}_{LP1} \leqslant 2 \cdot OPT$.

Proof. This follows from the above since we actually show that $V(TP+R) \geqslant \frac{1}{2} V^{max}_{LP1}$. Since $OPT \geqslant V(TP+R)$, the bound follows. □

4.3. A third algorithm: Iterated TP + R

In a variant of TP + R, we determine a color for demand nodes one by one, depending on which color contributes most to the objective function value in that demand node. Each time a demand node is colored, we resolve the TP taking this decision into account. We call this algorithm Iterated TP + R (ITP + R).

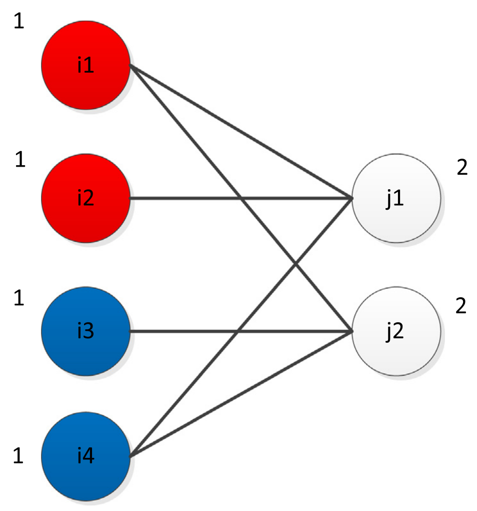

Fig. 4. A complete bipartite graph with $R = \{i_1, i_2\}$ and $B = \{i_3, i_4\}$; $a_{i_1} = a_{i_2} = a_{i_3} = a_{i_4} = 1$, and $b_{j_1} = b_{j_2} = 2$. Drawn edges have profit $p_{ij} = 1$, other (not drawn) edges have $p_{ij} = 0$.

Algorithm 3. Iterated TP + R

```
1: Set $D_R = D_B = \emptyset$.
2: Solve a TP based on profits $p_{ij}$. Call the resulting solution
   vector $x^*_{ij}$.
3: Find $j_R = \text{argmax}_{j \in D \setminus (D_R \cup D_B)} \sum_{i \in R_j} p_{ij} x^*_{ij}$ and
   $j_B = \text{argmax}_{j \in D \setminus (D_R \cup D_B)} \sum_{i \in B_j} p_{ij} x^*_{ij}$.
4: if $\sum_{i \in R} p_{ij_R} x^*_{ij_R} \geqslant \sum_{i \in B} p_{ij_B} x^*_{ij_B}$ then
5:    set $B_{j_R} := \emptyset$ and $D_R := D_R \cup \{j_R\}$
6: else
7:    set $R_{j_B} := \emptyset$ and $D_B := D_B \cup \{j_B\}$
8: end if
9: Go to line 2 until $D_R \cup D_B = D$.
10: Solve two TPs, one on the subgraph induced by $R \cup D_R$, and
    one on $B \cup D_B$, and get the overall solution value $V(ITP + R)$.
```

Notice that Iterated TP + R consists of solving $|D| + 2$ transportation problems. Although in a practical sense, this results in a good performance (see Section 5), from a worst-case point of view, this computational effort does not pay off.

Theorem 6. *The approximation ratio of Iterated TP + R is at most $\frac{1}{2}$.*

Proof. For the example in Fig. 4, the worst case optimal solution vector to the TP (note that there are several optimal solution vectors) is:

$$x^*_{12} = x^*_{21} = x^*_{32} = x^*_{41} = 1, \quad x^*_{11} = x^*_{22} = x^*_{31} = x^*_{42} = 0. \tag{30}$$

Thus, after the first iteration of the algorithm, we get:

$$\sum_{i \in R_1} p_{ij} x^*_{i1} = \sum_{i \in B_1} p_{ij_1} x^*_{i1} = 1, \tag{31}$$

$$\sum_{i \in R_2} p_{i2} x^*_{i2} = \sum_{i \in B_2} p_{i2} x^*_{i2} = 1. \tag{32}$$

Consider that we set $D_R = \{2\}$. Resolving the TP, the optimal solution vector is:

$$x^*_{12} = x^*_{21} = x^*_{41} = 1, \quad x^*_{11} = x^*_{22} = x^*_{31} = x^*_{32} = x^*_{42} = 0. \tag{33}$$

Thus:

$$\sum_{i \in R_1} p_{i1} x^*_{i1} = \sum_{i \in B_1} p_{i1} x^*_{i1} = 1, \tag{34}$$

$$\sum_{i \in R_2} p_{i2} x^*_{i2} = 1, \quad \sum_{i \in B_2} p_{i2} x^*_{i2} = 0. \tag{35}$$

And finally, $D_R = \{1, 2\}$ and $D_B = \emptyset$. Solving the TPs on the subgraphs results in:

$$V(ITP + R) = V(B \cup D_B) + V(R \cup D_R) = 0 + 2 = 2. \tag{36}$$

Again, we know that $OPT = 4$. Thus:

$$\frac{V(ITP + R)}{OPT} = \frac{1}{2} \qquad \square \tag{37}$$

4.4. Generalizations of Max-Red–Blue TP

In this section, we discuss a number of generalizations of Max-Red–Blue TP. One generalization arises when capacities are placed on the edges, i.e. for each edge, the flow transported over it cannot exceed the capacity of the edge. With a small modification (solving *capacitated* transportation problems on the red and the blue subgraph), the MAX-RB algorithm will produce a feasible solution to this problem. Moreover, MAX-RB still has a tight approximation ratio of $\frac{1}{2}$, since the arguments used in Theorem 4 remain valid.

In another generalization, we allow any topology of the underlying graph. The graph no longer needs to be bipartite; supply nodes, as well as demand nodes, can be joined by an edge. Transferring nodes (i.e. nodes with zero net supply or demand), are allowed as well. The goal is to find a profit-maximizing way to send flow from supply nodes to demand nodes, such that each demand node receives its flow from supply nodes that are either all red or all blue. The MAX-RB algorithm can be adapted as follows: we solve an uncapacitated min-cost flow problem on the red (blue) subgraph, i.e. we set the supply of the blue (red) nodes equal to zero, and return the solution vector that results in the highest solution value. Also for this generalization, a tight approximation ratio of $\frac{1}{2}$ holds.

In a third generalization, we no longer impose that $R \cap B = \emptyset$, instead we allow the existence of supply nodes that are both red and blue. We call these nodes *colorful*. The color constraints imply that a demand node can receive supply from either red nodes and colorful nodes, or blue nodes and colorful nodes. Of course, if all supply nodes are colorful, a TP arises. Again, it is easy to see that the MAX-RB keeps its approximation ratio of $\frac{1}{2}$; the transportation problem on the red subgraph, as well as the one on the blue subgraph, now include the colorful supply nodes.

Finally, Max-Red–Blue TP can be generalized to a K-color variant. In this case, we are given a weighted bipartite graph $G(S \cup D, E)$ with $C_1, C_2, \ldots, C_K \subseteq S, \bigcup_{k=1}^{K} C_k = S$ and $C_{k_1} \cap C_{k_2} = \emptyset$ for any $1 \leqslant k_1 < k_2 \leqslant K$, with $E \subseteq S \times D$. The problem is to find a maximum weighted flow from S to D with respect to supply and demand constraints, and the additional constraint that no two nodes $i_1 \in C_{k_1}, i_2 \in C_{k_2}, 1 \leqslant k_1 < k_2 \leqslant K$ can send flow to the same demand node $j \in D$. The MAX-RB algorithm can be generalized to this setting, by solving transportation problems on the subgraphs induced by $C_k \cup D$, for each color k. Also, the TP + R algorithm can be generalized to this setting by solving a transportation problem and next identifying which color gives the largest profit to each demand node. The proofs in Theorems 4 and 5 can trivially be generalized to show that both algorithms guarantee an approximation ratio of at least $\frac{1}{K}$. However, the approximation ratio of the generalization of Iterated TP + R to this K-color setting remains open.

The following example shows that the bound $\frac{1}{K}$ is tight for both algorithms. Consider a complete bipartite graph $C_1 = \{1, \ldots, K\}$, $C_2 = \{K+1, \ldots, 2K\}, \ldots, C_K = \{(K-1)K+1, \ldots, K^2\}$ and $D = \{1, \ldots, K\}$. All supply nodes have $a_i = 1$, all demand nodes have $b_j = K$. We denote:

$$\mathbf{1} = \begin{array}{c} 1 \\ 2 \\ \vdots \\ j \\ \vdots \\ K \end{array} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \quad \mathbf{e_j} = \begin{array}{c} 1 \\ 2 \\ \vdots \\ j \\ \vdots \\ K \end{array} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}. \tag{38}$$

Then the profit matrix is:

$$(p_{ij}) = \begin{pmatrix} \mathbf{1} & \mathbf{e_{K-1}} & \cdots & \mathbf{e_2} & \mathbf{e_1} \\ \mathbf{e_K} & \mathbf{1} & \cdots & \mathbf{e_2} & \mathbf{e_1} \\ & & \ddots & & \\ \mathbf{e_K} & \mathbf{e_{K-1}} & \cdots & \mathbf{1} & \mathbf{e_1} \\ \mathbf{e_K} & \mathbf{e_{K-1}} & \cdots & \mathbf{e_2} & \mathbf{1} \end{pmatrix} \tag{39}$$

Observe that the optimum value equals K^2, which is achieved by sending all supply from the nodes in C_k to demand node k, $k = 1, \ldots, K$. Further, observe that, when applying the generalization of MAX-RB, or the generalization of TP + R, it may happen that the supply of the nodes in each C_k $(1 \leqslant k \leqslant K)$ is distributed over the K demand nodes, leading to a value of K.

5. Computational study

5.1. Experimental setup

We have studied the behavior of the proposed formulations and heuristics with respect to the following problem characteristics:

- PS: the problem size ($|S| + |D|$, with $|S| = |D|$).
- PR: the proportion of red supply nodes $\left(\frac{|R|}{|S|}\right)$.
- DEN: the density of the graph $\left(\frac{|E|}{|S| \times |D|}\right)$.

To this purpose, we have generated a set of test instances according to a full factorial design with the characteristics described in Table 2. The procedure for generating these instances is described in Appendix A; the instances are available online (Vancroonenburg, 2013).

For Red–Blue TP, we have tested and compared the tightness of the LP-relaxations of Formulations 1 and 2, as well as the average computation times for each formulation and its linear relaxation. It is important to note that due to the instance generation procedure, infeasible instances may be generated for which no solution to the integer formulations or the LP-relaxations exists. Thus, the results on these instances are not considered in the forthcoming discussion and table (the number of feasible instances is included in the table).

For the maximization version, Max-Red–Blue TP, we compare the performances of the approximation algorithms, MAX-RB, TP + R and Iterated TP + R, with respect to the integer optimum of Formulation 1. In this setting, feasibility is not an issue as the **0**-vector is always a feasible solution.

This experimental setup has been coded in the C++ programming language and was compiled with the GNU Compiler Collection (GCC) 4.6.3. The IP-formulations and the LP-relaxation of Formulation 2 were implemented and solved with IBM Cplex 12.5, using the network simplex algorithm. The LP-formulation of Formulation 1 was implemented as a minimum-cost flow problem, and was implemented and solved with the network simplex algorithm in LEMON 1.3 (**L**ibrary for **E**fficient **M**odeling and **O**ptimization in **N**etworks) from the COIN-OR initiative. The approximation

Table 2
Characteristics of the generated test instances.

Parameter	Value
PS=$\|S\|+\|D\|$	$\{50,100,150,200,250,300,350,400\}$
PR=$\frac{\|R\|}{\|S\|}$	$\{0,0.1,0.2,0.3,0.4,0.5\}$
DEN=$\frac{\|E\|}{\|S\|\times\|D\|}$	$\{0.25,0.5,0.75,1.0\}$
S_{max}	50
C_{max}	20

algorithms MAX-RB, TP + R and Iterated TP + R also make use of a minimum-cost flow problem implementation in LEMON 1.3 for solving the transportation problem on the respective (sub) graphs.

All tests were done on a workstation computer equipped with two Intel Xeon 2670 2.6 GHz processors and 128 GB of main memory (RAM), which was running a Linux-based operating system. The MIP solver was configured to use only one processing thread, so this system was used to solve up to 16 instances in parallel (limiting the MIP solver to 8 GB of memory for each instance).

5.2. Results

5.2.1. Formulation 1 vs. Formulation 2

The effect of increasing the problem size (PS) from 50 to 400 nodes is summarized in Table 3 for Red–Blue TP. For each setting of PS, the results are averaged over all feasible instances of 240 instances; 10 instances for each of the parameter settings of PR and DEN. It is clear that the computation time of both models increases as the problem size grows, and that the difference in computation time between the two formulations also increases, indicating that Formulation 2 scales worse than Formulation 1. This is expected, as Formulation 2 has a quadratic number of y variables for each source/destination pair, whereas Formulation 1 only has a y variable for each destination node. Furthermore, the exclusionary constraints are also quadratic in number for Formulation 2, whereas they are linear in number for Formulation 1.

We can clearly see that for both formulations, the LP-relaxation becomes tighter as the problem size increases; i.e. the gap, defined as $\frac{V_{IP}-V_{LP}}{V_{IP}}$, decreases. Of course, then the difference between the LP-relaxations of both formulations also becomes smaller as the problem size grows. A possible explanation for this observation is that the impact of a single color constraint decreases when more nodes (or more edges) are present.

The proportion of red vs. blue nodes (PR) has a clear impact on the performance of both formulations, shown in Table 3. For each setting of PS, the results are averaged over all feasible instances of 320 instances; 10 instances for each of the parameter settings of PS and DEN. It shows that for Red–Blue TP the LP relaxation of Formulation 2 is tighter than Formulation 1, and the difference in tightness between the two formulations increases as the percentage of red nodes grows to 50%. However, as this percentage increases, the computation time for both formulations also grows, and faster for Formulation 2 than for Formulation 1. At 0% red nodes, both models reduce to the Transportation Problem, and the linear relaxation equals the integer optimum.

Finally, the effect of varying the density of the underlying graph (DEN), from 25% to 100% is also summarized in Table 3. For each setting of DEN, the results are averaged over all feasible instances of 480 instances; 10 instances for each of the parameter settings of PS and PR. It is clear that as the density grows to 100%, the gap between the integer optimum and the LP-relaxation of Formulation 1 and 2 decreases. Notice that computation times for Formulation 1 seem to be highest for a 50% density. For Formulation 2, the density for which computation times are highest, is larger, which can be explained by the fact that the number of constraints is directly dependent on the number of edges. Furthermore, the difference in gap between Formulation 1 and Formulation 2 becomes smaller as the density increases. Thus, the benefit of the stronger LP-relaxation of Formulation 2 again reduces as the density grows to 100%.

It is clear that although Formulation 2 has a tighter LP-relaxation than Formulation 1, this difference decreases as all factors considered (problem size, percentage red nodes, density) grow. Furthermore, in all cases the computation time increases much faster for Formulation 2 than for Formulation 1. This results in a large amount of computation time when trying to find an integer optimum using Formulation 2. Table 3 reports the number of instances that timed out after 3600 s. It shows that for PS > 200 nodes increasingly more instances cannot be solved in less than 1 h, and in fact (not reported in the table) some instances cannot be solved to optimality in less than 24 h. It is also clear that this is highly related to the ratio red/blue nodes. Therefore, we advocate using Formulation 1 over Formulation 2 in all cases.

Table 3
Comparison of the gap from the integer optimum, and the computation time for Formulation 1 and Formulation 2. Results shown w.r.t. the influence of the size (PS) of the graph, the proportion of red nodes (PR) and the density (DEN) of the graph, averaged over 10 runs of all values of the respective other parameters.

Param.	Value	# Feasible instances	IP Formulation 1	LP Formulation 1		IP Formulation 2	LP Formulation 2	
			T (s)	Gap (%) $\frac{V_{IP}-V_{LP1}}{V_{IP}}$	T (s)	T (s) (**# timeout after 3600 s**)	Gap (%) $\frac{V_{IP}-V_{LP2}}{V_{IP}}$	T (s)
PS (#)	50	187	0.1	9.5	< 0.1	0.7 (**0**)	4.2	< 0.1
	100	236	0.4	8.6	< 0.1	5.7 (**0**)	3.4	0.1
	150	240	1.7	7.4	< 0.1	47.9 (**0**)	2.8	0.2
	200	240	5.4	6.6	< 0.1	300.5 (**0**)	2.4	0.9
	250	240	12.0	5.8	< 0.1	949.7 (**18**)	2.1	2.7
	300	240	19.3	4.7	< 0.1	1393.3 (**42**)	1.6	6.3
	350	240	30.7	4.0	< 0.1	1817.6 (**70**)	1.3	12.2
	400	240	48.6	3.5	< 0.1	1993.7 (**82**)	1.1	22.0
PR (%)	0	315	0.2	0.0	< 0.1	0.2 (**0**)	0.0	0.1
	10	309	6.0	4.6	< 0.1	392.8 (**5**)	2.3	4.1
	20	307	12.3	6.6	< 0.1	918.5 (**27**)	2.6	6.0
	30	312	17.8	7.9	< 0.1	1132.3 (**51**)	2.9	7.2
	40	310	25.8	8.8	< 0.1	1273.8 (**60**)	3.0	8.4
	50	310	29.4	9.0	< 0.1	1324.3 (**69**)	3.0	8.7
DEN (%)	25	425	15.9	8.4	< 0.1	655.5 (**42**)	3.2	0.6
	50	478	24.3	7.2	< 0.1	945.8 (**81**)	2.7	3.1
	75	480	14.1	5.2	< 0.1	1007.2 (**69**)	1.9	7.0
	100	480	6.7	4.1	< 0.1	725.1 (**20**)	1.5	11.7

Table 4
Comparison on the gap from the integer optimum and the computation time for the TP + R, ITP + R and MAX-RB heuristics, averaged over 10 runs of all values of the respective other parameters. Results shown w.r.t. the influence of the problem size (PS), the percentage of red nodes (PR) and the density (DEN) of the graph.

Param.	Value	MAX-RB		TP + R		ITP + R		IP (Formulation 1)
		Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)	T (s)	T (s)
PS (#)	50	19.9	< 0.1	2.4	< 0.1	0.4	<0.1	0.1
	100	22.2	< 0.1	2.7	< 0.1	0.2	0.1	0.6
	150	22.9	< 0.1	2.6	< 0.1	0.2	0.2	2.2
	200	23.8	< 0.1	2.6	< 0.1	0.1	0.5	5.1
	250	23.8	< 0.1	2.3	< 0.1	0.1	1.0	10.0
	300	23.9	< 0.1	2.3	< 0.1	0.1	1.6	16.8
	350	24.1	< 0.1	2.1	< 0.1	0.1	2.6	28.9
	400	24.5	< 0.1	2.3	< 0.1	0.1	3.9	53.3
PR (%)	0	0.0	< 0.1	0.0	< 0.1	0.0	<0.1	0.1
	10	8.4	< 0.1	2.2	< 0.1	0.2	1.6	5.9
	20	18.3	< 0.1	3.2	< 0.1	0.2	1.5	12.1
	30	28.1	< 0.1	3.3	< 0.1	0.2	1.5	18.4
	40	38.1	< 0.1	3.0	< 0.1	0.2	1.4	27.1
	50	46.0	< 0.1	2.9	< 0.1	0.2	1.4	24.2
DEN (%)	25	21.7	< 0.1	2.0	< 0.1	0.3	0.8	17.2
	50	23.0	< 0.1	2.4	< 0.1	0.2	1.2	23.3
	75	23.6	< 0.1	2.6	< 0.1	0.1	1.4	11.9
	100	24.2	< 0.1	2.6	< 0.1	0.1	1.6	6.0

We can also report the same effects concerning the problem size, proportion of red nodes, and the density for the maximization version of Red–Blue TP. Given that the results are basically the same, we have omitted the corresponding table from this paper.

5.2.2. The approximation algorithms

The computational results for the approximation algorithms are summarized in Table 4, showing the relative gap between the heuristic result and the integer optimum, as well as the computation time, while the problem size, the percentage of red nodes, and the density of the network increase.

Overall, the gap with the integer optimum is considerably smaller for the ITP + R heuristic (again, with an exception for the special case where all supply nodes have the same color) than both MAX-RB and TP + R. MAX-RB, obviously being the most naive algorithm, performs worst of all and almost reaches its worst case behavior on instances where the percentage of red nodes is equal to 50%. It must be noted, however, that none of these heuristics dominates any of the other heuristics. Indeed, for each of the heuristics, instances were found for which it outperforms the others.

The fact that with ITP + R, the number of transportation problems that need to be solved increases with the number of demand nodes explains that the computation time increases faster than with TP + R, where only three transportation problems need to be solved, irrespective of the problem size.

6. Conclusion

In this paper, we discussed Red–Blue TP, which is a very natural generalization of the transportation problem, namely where supply nodes receive one of two colors, and demand nodes cannot receive flow from supply nodes with different colors. We settled the complexity status, showing that Red–Blue TP is NP-hard, and a constant-factor approximation is not likely to exist, even in a number of special cases. We discussed two IP formulations: although we can show that one formulation is strictly stronger than the other, experimental results show that the stronger formulation requires increasingly more computation time than the weaker formulation, as the problem size, the percentage of red nodes, or the density of the graph increase.

We also considered a maximization variant of Red–Blue TP, which is interesting with respect to approximation. We developed three approximation algorithms, each of which guarantee an approximation ratio of $\frac{1}{2}$. Computational experiments show that Iterated TP + R achieves the best approximation on most instances, at the expense of considerable computation times. Finally, we discussed a number of generalizations of Max-Red–Blue TP, including a variant with K colors.

Acknowledgements

This research is funded by a Ph.D. grant of the agency for Innovation by Science and Technology (IWT). This work is further supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office, and by FWO grant G.0729.13.

Appendix A. Instance generation procedure

The procedure for generating instances is described in Algorithm 4.

Algorithm 4. Red–Blue TP instance generation procedure

```
Require: $|S|, |D|, PR, DEN, S_{\max}, C_{\max}, seed$
 1: RAND $\leftarrow$ $seed$
 2: $R \leftarrow \{1,\ldots,PR\cdot|S|\}, B \leftarrow \{PR\cdot|S|+\ 1,\ldots,|S|\}$
 3: $S \leftarrow R \cup B, D \leftarrow \{1,\ldots,|D|\}$
 4: $S_{\text{total}} \leftarrow 0$
 5: for $i \leftarrow 1,\ldots,|S|$ do
 6:     $a_i \leftarrow \text{RAND}(1, S_{\max})$
 7:     $S_{\text{total}} \leftarrow S_{\text{total}} + a_i$
 8: end for
 9: $T \leftarrow \text{UNIQUERAND}(1, S_{\text{total}}-1, |D|-1)$
10: $SORT(T)$
11: $b_1 = t_1 - 0, b_2 = t_2 - t_1, \ldots, b_j = t_j - t_{j-1}, \ldots,$
    $b_{|D|} = S_{\text{total}} - t_{|D|-1}$
12: for $(i,j) \in S \times D$ do
13:     $c_{ij} \leftarrow \text{RAND}(0, C_{\max})$
14: end for
15: for $k \in (1-DEN)\cdot|S|\times|D|$
16:     $(i,j) \leftarrow \text{UNIQUERAND}(S \times D)$
17:     $c_{ij} \leftarrow -1$
18: end for
19: return Red-Blue TP $(S, D, (a_i), (b_j), (c_{ij}))$
```

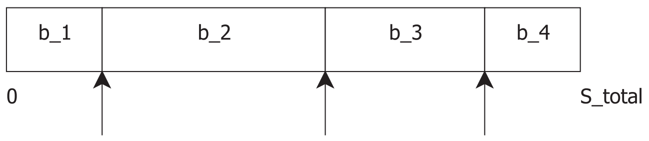

Fig. A.5. Procedure for randomly generating demand, matching supply. The procedure generates $|D| - 1$ random, unique numbers strictly between 0 and S_{total}. The numbers then divide the total supply over n demand nodes.

The procedure generates Red–Blue TP instances according to the following parameters:

- $|S|, |D|$: the number of supply and demand nodes,
- *PR*: the percentage of red supply nodes in the graph,
- *DEN*: the density of the graph,
- $S_{\max}$: the maximum supply for any give supply node,
- $C_{\max}$: the maximum cost (or profit, for Max-Red–Blue TP) for any given edge in the bipartite graph,
- *seed*: a seed value for the pseudo-random number generator.

Supply nodes are generated with supply a_i randomly selected in $[1, S_{\max}]$. The procedure ensures that total supply meets total demand by randomly dividing total supply over $|D|$ demand nodes (see Fig. A.5). First, $|D| - 1$ unique numbers between 0 and S_{total} (the total supply) are generated. Next, these numbers are sorted and the demand nodes are then generated as the pairwise difference between these numbers. Finally, to reduce the density of the graph, edges of the graph are randomly selected and removed (indicated by setting $c_{ij} = -1$).

References

Bai, G., Mao, J., & Lu, G. (2004). Grey transportation problem. *Kybernetes, 33*, 219–224.

Cao, B. (1992). Transportation problem with nonlinear side constraints a branch and bound approach. *Mathematical Methods of Operations Research (ZOR), 36*, 185–197.

Cao, B., & Uebe, G. (1995). Solving transportation problems with nonlinear side constraints with tabu search. *Computers & Operations Research, 22*, 593–603.

Ceschia, S., & Schaerf, A. (2011). Local search and lower bounds for the patient admission scheduling problem. *Computers & Operations Research, 38*, 1452–1463.

Demeester, P., Souffriau, W., De Causmaecker, P., & Vanden Berghe, G. (2010). A hybrid tabu search algorithm for automatically assigning patients to beds. *Artificial Intelligence in Medicine, 48*, 61–70.

Ferguson, R. O., & Dantzig, G. R. (1955). The problem of routing aircraft. *Aeronautical Engineering Review, 14*, 51–55.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A guide to the theory of NP-completeness*. WH Freeman & Co.

Goossens, D., & Spieksma, F. C. R. (2009). The transportation problem with exclusionary side constraints. *4OR, A Quarterly Journal of Operations Research, 7*, 51–60.

Hirsch, W. M., & Dantzig, G. B. (1968). The fixed charge problem. *Naval Research Logistics Quarterly, 15*, 413–424.

MacKinnon, J. (1975). An algorithm for the generalized transportation problem. *Regional Science and Urban Economics, 5*, 445–464.

Orden, A. (1956). The transhipment problem. *Management Science, 2*, 276–285.

Pferschy, U., & Schauer, J. (2013). The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization, 26*, 109–119.

Schreuder, J. A. M. (1992). Combinatorial aspects of construction of competition dutch professional football leagues. *Discrete Applied Mathematics, 35*, 301–312.

Sun, M. (2002). The transportation problem with exclusionary side constraints and two branch-and-bound algorithms. *European Journal of Operational Research, 140*, 629–647.

Vancroonenburg, W. (2013). Red–Blue transportation problem: Problem instances. <http://allserv.kahosl.be/∼wimvc/rbtp/rbtp-instances.zip> (Last accessed 11.04.13).