On the effectiveness of primal and dual heuristics for the transportation problem

JONAS SCHWINN*, AND RALF WERNER†
University of Augsburg,
Universitätsstraße 14, 86159 Augsburg, Germany

August 31, 2017

Abstract

The transportation problem is one of the most popular problems in linear programming. Over the course of time a multitude of exact solution methods and heuristics have been proposed. Due to substantial progress of exact solvers since the mid of the last century, the interest in heuristics for the transportation problem over the last few decades has been reduced to their potential as starting methods for exact algorithms. In the context of ever increasing problem dimensions, this paper contributes an in-depth study of heuristics with respect to their performance in terms of computation time and objective value. Furthermore, we consider – to the best of our knowledge for the first time – some simple efficient dual heuristics to obtain performance certificates based on weak duality without the need to solve the problem exactly. We test these heuristics in conjunction with state-of-the-art solvers on a variety of test problems. Thus we especially close the gap to almost outdated comparative studies from the last century, as for example Srinivasan & Thompson (1973); Glover *et al.* (1974); Gass (1990). For one class of random test problems, we extend previous approaches to provide a consistent and flexible problem generator with known solution. Based on our numerical results, it can be concluded that primal and dual heuristics are able to rapidly generate good approximations for specific randomly generated problem instances but – as expected – are not able to yield satisfactory performance in most cases.

*Email: jonas.schwinn@math.uni-augsburg.de
†Corresponding author. Email: ralf.werner@math.uni-augsburg.de

Keywords: transportation problem; primal heuristic; dual heuristic; problem generator for transportation problems

1 Introduction and motivation

1.1 Motivation

In this paper we present a detailed numerical study on the effectiveness of primal and dual heuristics for the transportation problem. Since the first statement of the transportation problem, in its continuous form due to Monge (1781), it has been the subject of numerous mathematical publications. The problem is known under several terms, among which are the general, classical or Hitchcock transportation problem as well as the distribution problem. We will consider the problem in its discrete, dense and uncapacitated form, without so called "inadmissible" cells, and refer to this form, see (2.1), as the transportation problem for the remainder of this paper. Studies of this problem date back to the 1930s and the work of A.N. Tolstoi (see Schrijver, 2002) and were followed by the well known publications of Hitchcock (1941), Kantorovich (1942) (continuous formulation), Koopmans (1948) and Dantzig (1963) who adapted his famous Simplex algorithm to the transportation problem – which we will call the (Primal) Transportation Simplex[1]. In addition to exact solvers, a number of primal heuristics were proposed. Most of these heuristics like the North-West Corner Rule or Minimum Cost Allocation methods implement straightforward ideas and can be found in virtually every textbook concerning the transportation problem. More sophisticated heuristics were for example proposed by Houthakker (1955), Reinfeld & Vogel (1958) and Russel (1969). Concerning exact solution methods, the groundbreaking work of Glover *et al.* (1974) and Srinivasan & Thompson (1973) established the Transportation Simplex as state-of-the-art solver thanks to new methods for base representation and pivot operations. Formerly, Out-of-Kilter methods had been the method of choice for the transportation problem, see for instance Gass (1990). Recent publications indicate nowadays increasing competitiveness of modern primal-dual or dual solvers from the combinatorial optimization field, cf. Kovács (2015). From this point on, the interest in heuristics shifted to their capability as starting methods for the Transportation Simplex. Numerical studies, cf. Srinivasan & Thompson (1973); Glover *et al.* (1974); Gass (1990); Gottschlich & Schumacher (2014); Schrieber *et al.* (2017), examine the total computation time of the heuristic and the time it takes an exact solver to compute an optimal solution from the heuristic

[1] This method is also known as the MODI method – Modified Distribution method or the Row-Column Sum method.

solution.

1.2 Our contribution

In contrast to this rather narrow point of view on heuristics, we provide in the following a novel analysis of the *quality of heuristic solutions* in terms of the gap between primal and dual heuristics. Our analysis provides a novel aspect insofar, as, up to now, primal-dual information has not been used to judge the quality of heuristics for the transportation problem. To the best of our knowledge, it also constitutes the first broad study on heuristics for the transportation problem in the last 40 years, see below. Until today, related numerical studies were usually focussed on exact solvers and not on heuristics, and only consider special cases of the transportation problem, see for instance Schrieber *et al.* (2017), or the solution of the more general minimum cost flow problem, cf. Ahrens & Finke (1980) or Kovács (2015). En passant, we also close the gap to almost outdated comparative studies in the last century of Srinivasan & Thompson (1973); Glover *et al.* (1974); Gass (1990) by investigating the performance of state-of-the-art Transportation Simplex codes on our problem instances.

Furthermore, as we are not aware of any efficient *dual heuristics* for the pure transportation problem[2], which runs in at most $\mathcal{O}(m^2)$, where m is the number of sources and sinks in the problem, we also add to the existing literature by providing novel efficient dual heuristics running in $\mathcal{O}(m^2)$.

Subsequently, we especially provide detailed insight on the computation time and on the solution quality of primal & dual heuristics on three different problem classes, two classes of random instances and one class from real world application (imaging sciences). Here, the generation of the random instances, see especially Appendix A, extends previous approaches to allow for a more flexible and consistent choice of the instances.

1.3 Main findings

Not unexpectedly, the quality of heuristic solutions cannot be assured across all problem classes. However, for some randomly generated problem instances, primal and dual heuristics quickly compute and verify solutions with function values within a few percent of the optimal value. Since primal heuristics are usually run as part of an exact solver whenever the Transportation Simplex is used, they come at almost no costs. Further, the cost for the validation of a heuristic primal solution via an efficient dual heuristic is also negligible compared to running an exact solver.

[2]The authors are aware of publications concerning the dual of the capacitated problem, cf. Glover & Klingman (1972b,a).

Furthermore, we find that the performance of heuristics and exact solvers differs significantly across problem classes. For specially structured problems general heuristics might fail to deliver good results and one has to resort to exact solvers or specifically tailored heuristics (see, e.g., Gottschlich & Schumacher, 2014). In summary, our study greatly supports current academic and industrial practice to rely on exact solvers for the transportation problem instead of using heuristics, especially when quite accurate solutions are asked for.

2 The transportation problem

In order to provide an intuitive understanding of the problem and the methods described in the remainder of this paper, let us start with the classical economical interpretation of the transportation problem. To this end, consider the transportation of a certain good from $i = 1,\dots,m$ origins to $j = 1,\dots,n$ destinations. Each of the origins has a supply a_i of the good whereas each of the destinations has a demand b_j for the good. The costs for shipping one unit of the good from i to j are given by c_{ij}. These cost coefficients are composed in the cost matrix $C := (c_{ij})_{i,j=1}^{m,n}$. A transportation plan (transportation matrix) $X := (x_{ij})_{i,j=1}^{m,n}$ is given by the amount x_{ij} of transported units between origin i and destination j. It is called feasible, if all supplies a_i are used (the row constraints are satisfied) and all demands b_j are saturated (the column constraints are satisfied). The aim of the transportation problem is then to find a feasible transportation plan at minimum cost. Mathematically, the transportation problem is given by the linear program:

$$\begin{array}{lrcll} \min\limits_{x} & \sum_{i=1}^{m}\sum_{j=1}^{n} & c_{ij}x_{ij} & & \\ \text{s.t.} & \sum_{j=1}^{n} & x_{ij} & = a_i & \forall i = 1,\dots,m \\ & \sum_{i=1}^{m} & x_{ij} & = b_j & \forall j = 1,\dots,n \\ & & x_{ij} & \geq 0 & \forall i,j \end{array} \tag{2.1}$$

where we have $c_{ij} \geq 0$, $a_i > 0$ and $b_j > 0$ without loss of generality. Furthermore we make no assumption on the number of c_{ij} equal to zero, that is, we assume the problem to be dense in the general case. We ensure feasibility of (2.1) by assuming the transportation problem to be *balanced*, that is $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. By quick inspection of the constraints, we also have that (2.1) is bounded and thus allows for an optimal solution. Since the roles of the a_i and b_j are completely interchangeable, we assume $m \geq n$ without loss of generality for the remainder of this paper and further call a problem *symmetric*, whenever $m \leq 10 \cdot n$ and *asymmetric* if the opposite holds.

2.1 Properties

Setting $c := (c_{11}, c_{12}, \ldots, c_{1n}, c_{21}, \ldots c_{mn})^\top$ and $x := (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots x_{mn})^\top$ we can write (2.1) in the form

$$\begin{array}{lrcl} \min\limits_{x \in \mathbb{R}^{mn}} & c^\top x & & \\ \text{s.t.} & Ax & = & e \\ & x & \geq & 0 \end{array}$$

for $e := (a,b)^\top$ and corresponding A. It is well known that in the case of the transportation problem we have $\operatorname{rank}(A) = m+n-1$. Following the standard definitions of linear optimization, we call a set $B \subsetneq \{(i,j) \mid i = 1,\ldots,m; j = 1,\ldots,n\}$ with $|B| = m+n-1$ a *basis* of (2.1), whenever the columns A_{ij} corresponding to the index tupels in B are linearly independent. Further, we call a solution x satisfying $Ax = e$ a *basic solution* with respect to a base B whenever $x_{ij} > 0$ implies (i,j) in B, and a *feasible basic solution* respectively, whenever additionally $x \geq 0$ is satisfied. A basic solution is called *degenerate* whenever we have $x_{ij} = 0$ for a least one (i,j) in B.

Given the special structure of A, one is also able to provide a very intuitive characterization of bases: a given set $B \subsetneq \{(i,j) \mid i = 1,\ldots,m; j = 1,\ldots,n\}$ with $|B| = m+n-1$ is a base of (2.1) if and only if no subset of its entries form a cycle in the transportation matrix. For a detailed introduction into characteristics of bases in minimum cost flow problems (e.g. basis triangularity and graph theoretical interpretations) let us refer for instance to Ahuja *et al.* (1993). Accordingly, the dual of the transportation problem is given as:

$$\begin{array}{lll} \max\limits_{\substack{u \in \mathbb{R}^m \\ v \in \mathbb{R}^m}} & u^\top a + v^\top b & \\ \text{s.t.} & u_i + v_j \leq c_{ij} & \forall i,j. \end{array} \tag{2.2}$$

As the transportation problem is a feasible and bounded linear program, we have strong duality and complementary conditions for (2.1) and (2.2). We follow the common notation and denote the dual feasibility of a given dual solution (u,v) by the reduced costs

$$c_{ij}^{uv} := c_{ij} - u_i - v_j.$$

This allows a compact notation of the weak complementary slackness conditions

$$\begin{array}{l} x_{ij} > 0 \Rightarrow c_{ij}^{uv} = 0 \\ c_{ij}^{uv} > 0 \Rightarrow x_{ij} = 0 \end{array} \tag{2.3}$$

which are equivalent to the optimality of x and (u,v) for feasible primal and dual solutions x and (u,v).

3 Primal and dual heuristics

In the course of modern operations research a multitude of primal heuristics for the transportation problem has been proposed. Taking into account the development of exact solvers, only a few of these concepts remain computationally competitive today. For the sake of conciseness of our study and expressiveness of our results, we made a preselection by the following criteria.

- From the start, we excluded any candidate where computing the solution of the heuristic took a disproportionate amount of time compared to the time needed to produce exact solutions. Among thereby excluded heuristics were methods like Russel's Method (Russel, 1969), Houthakker's Method of Mutually Preferred Flows (Houthakker, 1955) and Vogel's Approximation Method (Reinfeld & Vogel, 1958) as well as some more recent adaptations of it. One might argue to exclude the Matrix Minimum Rule and its variants for the same reason, since it is similarly reported to be too time-consuming (see, e.g., Srinivasan & Thompson, 1973; Glover *et al.*, 1974). However, in accordance with Gottschlich & Schumacher (2014), we found that one can successively speed up this heuristic by sorting the cost coefficients in advance and iterating once through the sorted list to obtain an equivalent solution. We assume this concept is to some extent equivalent to the Matrix Minimum by Rank Method proposed by Lee[3]. In contrast to the vast amount of publications towards the transportation problem, this rather obvious approach has been neglected in former comparative studies. The same strategy was applied to ensure the competitiveness of related methods like the Modified Russel's Method and the Large Amount Least Cost Rule which was also proposed by Lee and mentioned in Srinivasan & Thompson (1973); Glover *et al.* (1974).

- In a second step, we excluded some methods that are sufficiently represented by other heuristics, in the sense that they employ rather similar concepts and produce comparable or worse results in terms of computation time and solution quality. Thus, we excluded methods like the Two Smallest in a Row Rule (Srinivasan & Thompson, 1973) as well as the (Alternating) Row Column Rule (Srinivasan & Thompson, 1973) which are represented by the Row Minimum Rule and the Modified Row Minimum Rule (Srinivasan & Thompson, 1973) as well as their column counterparts and the Tree Minimum Rule.

[3]This work constitutes a Master's thesis submitted at the Graduate School of Business at the University of California in 1968 (see, e.g., Srinivasan & Thompson, 1973). Unfortunately, the authors were not able to gain access to the original work.

- Finally, this study's focus is on "classical" heuristics for the transportation problem and thereby we do not cover heuristics that consist of originally exact solvers producing inexact solutions by concepts like considering a relaxation of the problem or terminating before optimality is reached (e.g. the Shortlist Method in Gottschlich & Schumacher (2014)).

From a complexity point of view and assuming $m \geq n$, the fastest known strongly polynomial algorithms are due to Kleinschmidt & Schannath (1995) and Brenner (2008), with complexities $\mathcal{O}(m^2 n \log m)$ and $\mathcal{O}\left(mn^2(\log m + \log n)\right)$ respectively. For this reason, we focus on heuristics running in $\mathcal{O}(m^2)$ or better and do not consider other heuristics any further.

Since we are not aware of any competitive non-basic heuristics, all primal heuristics presented in Section 3.1 are *primal basic heuristics*, that is they produce feasible basic solutions for (2.1) and could thus serve as starting procedure for the Transportation Simplex.

In the case of dual heuristics we are not aware of any efficient dual heuristic running in $\mathcal{O}(m^2)$ or faster. Therefore, we implemented a variety of straightforward ideas and present these in Section 3.2.

3.1 Primal heuristics

The primal heuristics can be separated in two main classes. The first class, which we will denote as the *elimination heuristics*, consists of iterative methods that consider in each step a cell, row or column of the cost matrix and subsequently eliminate a row or column from further consideration by the selection of a new basis variable(s). The second class constitutes *greedy heuristics* which sort the (modified) cost matrix once in advance and then iterate through the list of sorted cost coefficients. In the degenerate case, a basic solution is hereby ensured by running an additional subroutine after the greedy algorithm.

3.1.1 Elimination heuristics

In order to give a compact description of the elimination heuristics, let us consider the Prototype Elimination Algorithm. All following heuristics implement this prototype heuristic and differ only in the implementation of the selection rule in Step (1). Note further that in Step (3) either i or j are deleted from their respective sets but never both. This ensures (including the degenerate case), that after $m+n-1$ iterations, B constitutes a feasible basis for (2.1).

North-west Corner Rule (NWCR) The North-West Corner Rule is a very simple heuristic that rapidly yields a basic solution, but generally realizes a rather poor

Algorithm 1 Prototype Elimination Algorithm

(0) Initiate lists of the indices $I := \{1, \ldots, m\}$ (rows) and $J := \{1, ..., n\}$ (columns).
Set $B := \emptyset$, $k := 1$.

In iteration k:

(1) Choose a new base element (i, j) with $i \in I$ and $j \in J$ by a selection rule.

(2) Set $x_{ij} := \min(a_i, b_j)$, $a_i := a_i - x_{ij}$ and $b_j := b_j - x_{ij}$.

(3) If $a_i = 0$ delete i from I (eliminate row i);
Otherwise delete j from J (eliminate column j).

(4) Add (i, j) to B.

(5) Set $k := k + 1$. If $k = m + n$ stop; otherwise goto (1).

objective value. It begins by taking $i = j = 1$. Subsequently, whenever i or j is deleted from I or J in Step (3), either $(i+1, j)$ or $(i, j+1)$ is chosen in the next visit of Step (1). Graphically speaking, the heuristics moves in a stair-like pattern from the north-west corner of the matrix C to the south-east corner of C. Since the selection rule can be applied in constant time, this leads to a complexity of $\mathcal{O}(m)$.

Row Minimum Rule (RMR) This rule cycles once through the indices i in I. Starting with $i = 1$ it (repeatedly) selects minimum entries $j = \arg\min_{j \in J} c_{ij}$ of the i-th row of C and adds (i, j) to B. This procedure is repeated until $a_i = 0$ at which point i is deleted from I and the algorithm advances to the next element in I. This results in a complexity of $\mathcal{O}((m+n) \cdot n) = \mathcal{O}(mn)$.

Modified Row Minimum Rule (MRMR) This rule is a modification of (RMR), cycling (possibly multiple times) through the remaining indices in I where for each row i only one minimum element j is selected before the algorithm advances to the next element in I. Whenever i is the last element in I, by convention the next element is considered to be the first element in I. This gives the same complexity $\mathcal{O}((m+n) \cdot n) = \mathcal{O}(mn)$ as for (RMR).

Column Minimum Rule (CMR) and Modified Column Minimum Rule (MCMR) Exchanging the roles of the columns (I) and rows (J), we obtain an equivalent Col-

umn Minimum Rule for any Row Minimum Rule above. The complexity of the Column Minimum Rules is $\mathcal{O}(m^2)$.

Tree Minimum Rule (TMR) This rule starts by selecting (i, j) corresponding to the minimal entry c_{ij} of the cost matrix. Subsequently, this rule alternately scans rows and columns of C. Suppose, w.l.o.g., after the first allocation, j was deleted from J. The rule proceeds by (repeatedly) selecting minimum entries $j = \arg\min_{j \in J} c_{ij}$ of the i-th row of C and adding (i, j) to B, as in one step of (RMR). Subsequently, it scans the column j of the element (i, j) selected last in Step (2) and chooses minimum entries $i = \arg\min_{i \in I} c_{ij}$ of this column until j is deleted from J, as in one step of (CMR). Accordingly, the algorithm proceeds by scanning the row i corresponding to the last minimum entry of the column j. The algorithm proceeds in the same fashion, alternating between rows and columns until $m+n-1$ assignments have been made. An advantage of this procedure is that the elements of the corresponding base are considered in a depth-search sequence with respect to the spanning tree corresponding to the basis. This is for example advantageous for the computation of basis representations in the Transportation Simplex (see, e.g., Ahuja *et al.*, 1993). The resulting complexity is $\mathcal{O}(m^2)$.

3.1.2 Greedy heuristics

As mentioned above the traditional Matrix Minimum Rule (and related methods) can be successively speed up by applying a greedy methodology. We will give four methods of this type in this section and further methods using the greedy methodology in Section 3.4.

Matrix Minimum Rule (MMR) This heuristics implements a greedy approach with respect to the cost coefficients. In the beginning one sets $B := \emptyset$ and all $x_{ij} := 0$ and computes a sorting of C, that is, a mapping

$$\rho : \{1, \ldots, mn\} \to \{(i, j) | i = 1, \ldots, m \text{ and } j = 1, \ldots, n\}$$

such that

$$c_{\rho(k)} \leq c_{\rho(k+1)} \text{ for all } k = 1, \ldots, mn-1.$$

Then, in each step $k = 1, \ldots, mn$, one has $(i, j) := \rho(k)$ and sets $x_{ij} := \min(a_i, b_j)$, $a_i := a_i - x_{ij}$ and $b_j := b_j - x_{ij}$. In the case that $x_{ij} > 0$, (i, j) is added to B. In the non-degenerate case, the method computes a feasible basic solution with respect to a basis B for the transportation problem. In the degenerate case, a subroutine is run afterwards to complete the basis B. Empirical investigation showed that the time needed by the subroutine is negligible compared to the overall time

of the heuristic. Its worst-case complexity is $\mathcal{O}(m+n)$. Since the sorting of C can be done in $\mathcal{O}(mn \cdot \log m)$, the overall worst-case complexity of the method is $\mathcal{O}(mn \cdot \log(m) + mn) = \mathcal{O}(mn \cdot \log m)$.

Modified Russel's Method (MRUM) This adaption of Russel's Method proposed by Gottschlich & Schumacher (2014) implements the Matrix Minimum Rule on a modified version of C. In the beginning of the algorithm each entry c_{ij} is set to $c_{ij} - u_i - v_j$ where $u_i = \max_{j \in J} c_{ij}$ and $v_j = \max_{i \in I} c_{ij}$. The intention is to approximate the reduced costs in the Transportation Simplex by estimating dual multipliers u_i and v_j. In contrast to the original method proposed by Russel (1969) this modification is done once at the beginning of the algorithm for the full lists I and J instead of updating with respect to I and J in every iteration. Thereby, the greedy approach is applicable and we have a complexity of $\mathcal{O}(mn \cdot \log m)$.

Large Amount Least Cost (LALC) In contrast to the two methods above, this heuristic, also proposed in the Master's thesis of Lee (see above), implements a greedy heuristic with respect to the remaining supply and demand. That is, it iteratively chooses the maximum entry of the updated a and b and selects the cell in the corresponding row or column with the lowest costs. Using suitable data structures (e.g. Fibonacci heaps) the maximum entry can be found in $\mathcal{O}(\log m)$ in each step. Thereby, one obtains an overall complexity of $\mathcal{O}(m \log m)$.

3.2 Dual heuristics

In the following, we present a list of simple efficient dual heuristics. Here, we only consider methods producing spanning solutions (see, e.g., Glover & Klingman, 1972a) for (2.2). A dual feasible solution (u,v) is called *spanning*, if for any i there exists a j such that $c^{uv}_{ij} = 0$ and for any j' there exists an i' such that $c^{uv}_{i'j'} = 0$. This obviously constitutes a necessary condition for (u,v) to be optimal.

Row First Method (RFM) In this rule, the dual variables are determined by firstly choosing $u_i := \min_{j=1,\ldots,n} c_{ij}$ and subsequently setting $v_j := \min_{i=1,\ldots,m} c_{ij} - u_i$. This can be done with complexity $\mathcal{O}(mn)$.

Column First Method (CFM) Employs the same methodology as (RFM) with the roles of u (rows) and v (columns) exchanged. This results in the same complexity $\mathcal{O}(mn)$.

Dual Greedy Method (DGM) Here, u and v are initially set to zero and the objective function $(a,b)^\top$ is sorted in descending order. The variables u_i or v_j are then considered in order of the corresponding entries a_i and b_j in the sorted objective function. For any variable u_i or v_j, we then set $u_i := \min_{j=1,\ldots,n} c_{ij} - v_j$ or $v_j := \min_{i=1,\ldots,m} c_{ij} - u_i$. This leads to a complexity of $\mathcal{O}((m+n)\log(m+n) + (m+n)\cdot m) = \mathcal{O}(m^2)$.

Maximal Gain Method (MGM) This method provides a variation of the dual greedy approach. The same steps as in (DGM) are executed but with respect to an altered cost function, where a_i is set to $a_i \cdot \left(\min_{j=1,\ldots,n} c_{ij}\right)$ and b_j is set to $b_j \cdot \left(\min_{i=1,\ldots,m} c_{ij}\right)$. Thus, the dual variables are considered in a different order than in (DGM). The complexity is again $\mathcal{O}(m^2)$.

Pull Push Method (PPM) One major weakness of the rules above is that negative values in a dual solution cannot be realized. This fact is addressed here by extending the dual greedy approach presented above. Thus, u and v are initially set to zero and the objective function $(a,b)^\top$ is sorted in descending order. Then, we make negative assignments for the last $\lfloor (m+n)/2 \rfloor$ elements. Suppose therefore that u_i corresponds to the $k-th$ element of the sorted objective function and $k \geq m+n-\lfloor (m+n)/2 \rfloor$. We then set $u_i := -M\frac{k}{m+n}$ where $M := \min_{ij} c_{ij}$. Subsequently the same assignments as in (DGM) are made which leads to an overall complexity of $\mathcal{O}(m^2)$.

3.3 Overview of worst-case complexities

Recapitulating, let us give an overview of the described methods and their worst-case complexities:

Primal Methods	**Complexity**
North-West Corner Rule	$\mathcal{O}(m)$
Row Minimum Rules	$\mathcal{O}(mn)$
Column Minimum Rules	$\mathcal{O}(m^2)$
Tree Minimum Rule	$\mathcal{O}(m^2)$
Matrix Minimum Rule	$\mathcal{O}(mn\log m)$
Modified Russel's Method	$\mathcal{O}(mn\log m)$
Large Amount Least Cost Rule	$\mathcal{O}(m\log m)$
Dual Methods	**Complexity**
Row First Method	$\mathcal{O}(mn)$
Column First Method	$\mathcal{O}(mn)$
Dual Greedy Method	$\mathcal{O}(m^2)$
Pull Push Method	$\mathcal{O}(m^2)$
Maximal Gain Method	$\mathcal{O}(m^2)$

Table 1: Overview of the worst-case complexities of all presented heuristics.

3.4 Constructing primal solutions from dual solutions

In this section let us provide a simple scheme to construct a feasible primal solution x from a given feasible dual solution (u,v). In this context one can find various ways to construct primal solutions from dual solutions in the literature (see for instance Russel's Method, primal-dual methods or the Simplex Method), however, in order to guarantee low computation time (and thereby justify the use of such a heuristic before running an exact solver), we omit any strategy requiring more involved subprocedures like maximum flow or transportation algorithms. Instead our approach is based on the greedy version of the Minimum Matrix Rule.

Our goal is to construct, for a given feasible dual solution (u,v), a feasible primal solution x such that the complementary slackness conditions (2.3) are somewhat approximated. For this reason, we apply the greedy version of the Matrix Minimum Rule on a dual feasible spanning solution (u,v) provided by one of the dual heuristics. This strategy is justified in the following: Assume a feasible spanning solution (u,v) and the reduced cost matrix $C^{uv} := (c_{ij}^{uv})_{i,j=1}^{m,n}$. Clearly, we have $C^{uv} \geq 0$ and $c_{ij}^{uv} = 0$ for at least m entries in C^{uv}. Thus, any minimum cost allocation method applied to C^{uv} will focus on these entries of C^{uv} and small entries c_{ij}^{uv} in general and thereby approximate the complementary slackness conditions (2.3) by keeping $\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}c_{ij}^{uv}$ small. The complexities of these methods are the combined complexities of the applied dual heuristic and the Matrix Minimum Rule.

Note that one could argue to include (the modified) Russel's Method in this class of methods, but as argued above we like to focus at this point on methods using a *feasible* dual solution to produce a feasible primal solution.

3.5 Constructing dual solutions from primal solutions

In addition to the previous section, we also like to present a simple scheme to produce feasible dual solutions from feasible primal solutions. Again, we aim to rapidly approximate the complementary slackness conditions (2.3). That is, given a feasible primal solution x, we want to minimize c_{ij}^{uv} whenever $x_{ij} > 0$. Therefore, let us employ a strategy used in the Network Simplex Method. Here, one makes use of the tree representation (depth search order) of basic solutions and set $c_{ij}^{uv} = 0$ whenever $x_{ij} > 0$ (see Ahuja *et al.*, 1993). Apart from the optimal case, this leads to an infeasible dual solution. Therefore, we use a second step where we employ the Row First Method on the cost coefficients c_{ij}^{uv} and alter the dual variables achieved in the first step to ensure a feasible spanning solution. As one needs a depth-search sequence of the primal variables to efficiently carry out the first step, we use the Tree Minimum Rule to avoid additional computations. As for the dual method, we resort to the Row First Method because of its low computation time, since the results of other (more sophisticated) methods did not lead to significantly better solutions in our numerical experiments.

3.6 Exact solvers

In this section we briefly mention the two exact solvers we used as a reference point to evaluate the computation time and solution quality of the heuristics.

3.6.1 Transportation Simplex

In our implementation of the Transportation Simplex we follow the numerical evaluations of Kovács (2015) for the Network Simplex. The transportation problem constitutes a minimum cost flow problem on a complete bipartite graph. Thus, in contrast to general network flow problems, there is no need to store any information on which nodes are connected by arcs. More precisely, there is no need to store any other information than the parameters C, a and b. When specializing their algorithm to the transportation problem the only concession to be made is in finding the entering variable in the simplex step. All other operations and data structures remain the same as in the Network Simplex, especially all operations performed on the basis. Thus, we follow the recommendation of Kovács (2015)

to "substantially improve the performance" of the algorithm and use the XTI (eXtended Threaded Index) labeling procedure proposed by Glover *et al.* (1979) over the commonly used ATI (Augmented Threaded Index) procedure for basis representation (see Ahuja *et al.*, 1993). We used the Tree Minimum Rule to produce initial solutions for the Transportation Simplex, since it allows for an efficient initialization of the XTI representation.

3.6.2 Network Simplex

Furthermore, we include in our study the Network Simplex of IBM's ILOG CPLEX 12.6.2 for MATLAB since our numerical investigations showed that it clearly outperforms other LP solvers provided by ILOG CPLEX or GUROBI on the transportation problem. These findings are compatible with traditional and also more recent studies (see, e.g., Glover *et al.*, 1974; Schrieber *et al.*, 2017). Moreover, Kovács (2015) showed that the Network Simplex is competitive with the fastest combinatorial algorithms on a variety of general minimum cost flow problems.

3.6.3 Performance of exact solvers

The performance of both solvers is comparable across all problem classes, with small advantages for CPLEX on asymmetric problems and on the DOTMARK instances. For more details let us refer to Figures 2 and 4, as well as Table 3.

3.6.4 Further implementation details

Although all methods were implemented to the best of our knowledge, for the Large Amount Least Cost Method we only achieve a complexity of $\mathcal{O}(m^2 \log m)$ instead of $\mathcal{O}(m \log m)$. The reason for this efficiency loss is due to MATLAB not allowing for a proper efficient implementation of priority lists like for instance Fibonacci Heaps.

4 Numerical experiments

4.1 Problem classes

In the following we present three different problem classes of the transportation problem. The first class constitutes the DOTMARK benchmark considered in Gottschlich & Schumacher (2014), the other two problem classes represent randomly generated problem instances. For problem class 2, we follow common strategies to produce

problem instances by drawing parameters (C,a,b) from suitable uniform distributions. Finally, for problem class 3, we employ a problem generator, which randomly generates optimal solutions and corresponding problem instances for given problem dimensions (m,n). Observe, that in this study, we restrict ourselves to dense problems, that is, the cost matrix C is not sparse in the general case. Moreover, to ensure comparability of these classes, we will force $\sum_i a_i = \sum_j b_j = 1$ and $0 \leq C \leq 1$ for all problem instances.

In the case of randomly generated problem instances, the parameters (C,a,b) are usually drawn from the uniform distribution (see, e.g., Srinivasan & Thompson, 1973; Glover *et al.*, 1974; Ahrens & Finke, 1980; Sandi, 1986). Moreover, reports on the use of different probability distributions, cf. Ross *et al.* (1975), suggested that the uniform distribution produced the hardest instances. Thereby, all our cost coefficients c_{ij} are generated by sampling from a uniform distribution. When sampling for a and b, one additionally has to make sure to satisfy $\sum_i a_i = \sum_j b_j = 1$. Thus, in order to produce statistical sound values for a and b, we uniformly generate points on the standard $(m-1)$-simplex and the standard $(n-1)$-simplex by sampling from a Dirichlet distribution.

DOTMARK[4] This benchmark was made publicly available by Schrieber *et al.* (2017) for the evaluation of discrete transportation algorithms computing the Wasserstein distance between images. It provides 10 different classes of black-and-white images where each class contains 10 different images in resolutions from 32×32 to 512×512. Each image is defined by the gray values of its pixels ranging from zero (white) to 255 (black). For any two images P1 and P2, a transportation problem can be generated by traversing the pixels e.g. in a row-wise order and setting a_i equal to the gray value of the i-th visited pixel in P1 and b_j equal to the gray value of the j-th visited pixel in P2. The costs c_{ij} are then chosen as the quadratic euclidean distance of the pixel positions of pixel i and j. In order to compare our study to Schrieber *et al.* (2017) we generated transportation problems for all pairs of two different images of a given class and fixed resolution 32×32. This amounts to $\binom{10}{2} = 45$ problem instances per class. For a detailed description of these problem instances let us refer to Schrieber *et al.* (2017). Since (C,a,b) are, by construction, non-negative integer values, we finally normalize

$$a_i := \frac{a_i}{||a||_1},\ b_j := \frac{b_j}{||b||_1} \text{ and } C := \frac{C}{||C||_\infty}$$

for the sake of comparability.

[4] http://www.stochastik.math.uni-goettingen.de/

UNIFORM As mentioned above we follow the common approach (see, e.g., Srinivasan & Thompson, 1973; Glover *et al.*, 1974; Ahrens & Finke, 1980; Sandi, 1986) for randomly generating general transportation problems. Thus, we draw the cost coefficients c_{ij} from a uniform distribution on $[0,1]$ where afterwards we divide by $||C||_\infty$ to normalize the problem [5]. In contrast to former studies we prefer to sample a and b by means of a Dirichlet distribution to ensure uniformly distributed a_i and b_j.

SOLGEN As mentioned above uniform distributed costs provide harder problem instances, therefore we will include a third problem class constituting easier-to-solve problems. This problem class comes with the advantage that problem instances are produced together with a corresponding optimal solution. To achieve this, we apply similar concepts as presented in Jeffrey & James (1994) for capacitated minimum cost flow problems. We improve their approach for the generation of uncapacitated transportation problem by ensuring a uniformly distributed topological structure of the corresponding optimal solution. In short, the approach consists of two steps: For given problem dimensions m and n, one begins by randomly constructing basic solutions $x \in \mathbb{R}^{mn}$, $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ together with the corresponding base B. In a second step, one obtains parameters C, a and b such that x and (u,v) are a primal-dual optimal solution pair satisfying the complementary slackness conditions (2.3). For a more detailed description of this problem class, see Appendix A. In terms of difficulty Jeffrey & James (1994) indicate that the problems generated by their algorithm "are of comparable difficulty" to problems generated by NETGEN, a commonly used minimum cost flow problem generator.

Overview Concluding, we give the expected value and the variation of the cost coefficients for the three problem classes in Table 2.

4.2 Results

Let us start with problem classes SOLGEN and UNIFORM on symmetric problems with equally sized vectors a and b, i.e. $m = n$.

- In terms of computation time, all presented heuristics are at least 10 times faster than the exact solvers, as can be seen in Figure 2. This justifies the use of each of these heuristics over exact solvers if one is mainly interested in an approximate solution.

[5] Note that this last operation has almost no effect, since even for low dimension, the probability that one entry of C is close to 1 is large. For example for $m = n = 25$, the chance for C to have one entry larger than 0.99 is 99.8%.

Generators	$\mathbb{E}(C)$	$\sigma(C)$	empirical $\mathbb{E}(C)$	empirical $\sigma(C)$
DOTMARK	-	-	0.145 – 0.177	0.118 – 0.148
UNIFORM	0.5	$\frac{1}{\sqrt{12}}$	0.500	0.289
SOLGEN	-	-	0.593	0.166

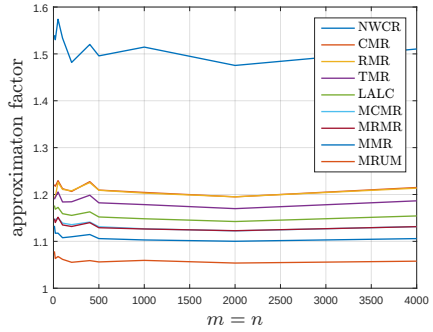
Table 2: Analytical and empirical distribution parameters for the costs of the three problem classes. In case of the DOTMARK instances we give the interval of the values over all classes. In case of the UNIFORM and SOLGEN instances we give the values for $m = n = 2000$.

- In Figure 1 we illustrate approximation factors for all heuristics, obtained by dividing the function value of the respective solutions by the optimal function value.
  - As depicted in Figures 1 and 2, the heuristics work quite well for the SOLGEN instances where the primal-dual gap can be reduced to less than 7% of the optimal value in 10% of the time an exact solver needs to find the optimal solution.
  - Especially, we find that the primal heuristics using dual information (including Russel's Method) produce very good upper bounds (primal gap of 5%) and the more sophisticated dual methods involving sorting schemes produce lower bounds which are within 2% percent of the optimal value.
  - These results reduce to a primal gap of 50% and a dual gap of 25% in the case of the UNIFORM instances and 14% of the time needed by an exact solver. This is due to the fact that the more time-consuming Large Amount Least Cost Method [6] is needed to compute the upper bound.
  - The Pull Push Method gives the best lower bound.

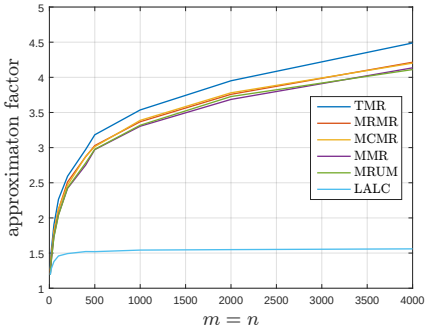
To analyze the performance on asymmetric problems we keep $n = 50$ fixed and only increase m.

- It can be observed in Figures 3 and 4 that, by and large, problems are better approximated by the heuristics and solved easier by the exact solvers as the ratio of m/n grows.

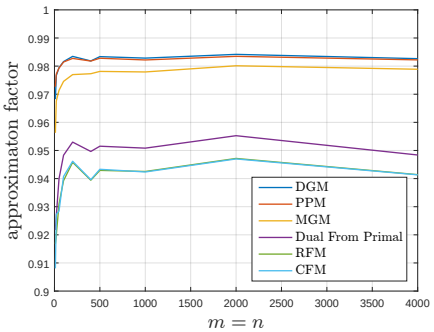
[6] As mentioned above the computation time of the LALC Method could be further improved by the use of suitable data structures.



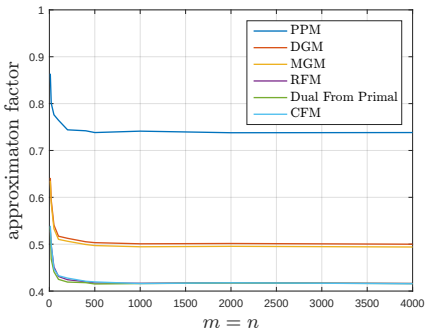
(a) Primal heuristics on SOLGEN.



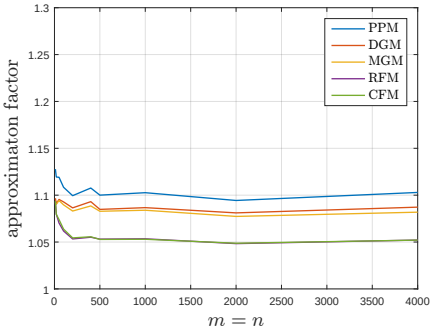
(b) Primal heuristics on UNIFORM.



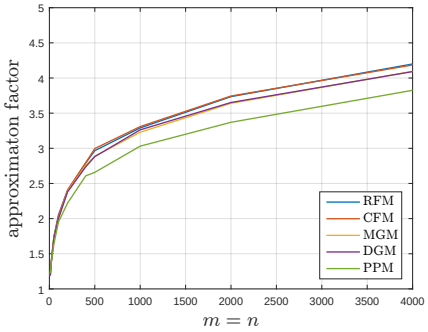
(c) Dual heuristics on SOLGEN.



(d) Dual heuristics on UNIFORM.

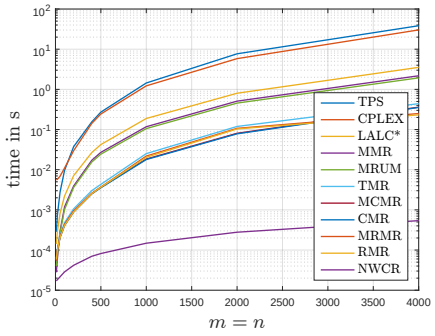


(e) Primal from dual heuristics on SOLGEN for different dual methods (see Section 3.4).

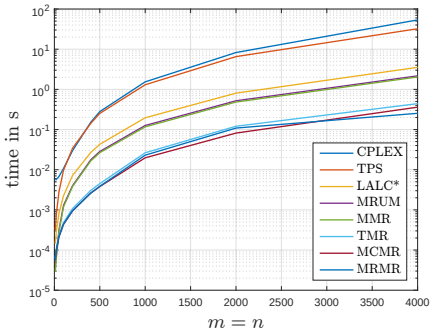


(f) Primal from dual heuristics on UNIFORM for different dual methods (see Section 3.4).

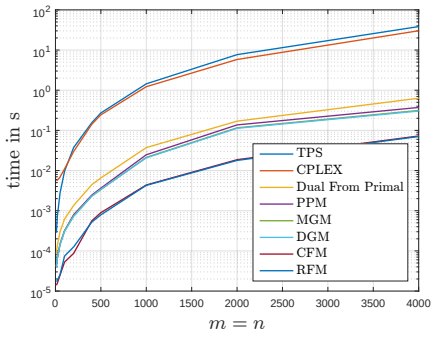
Figure 1: Approximation factors for the problem classes SOLGEN and UNIFORM on **symmetric** problems, averaged over 100 problem instances. The approximation factor is given by $\frac{val}{opt}$ where *val* is the function value of the solution of the heuristic and *opt* is the optimal function value for the problem. The positions of the heuristics in the legend corresponds to the position of the respective graph in the figure. Note that we omitted certain heuristics such as (NWCR) and (MRR) in some figures when their approximation factors where far worse than the rest, see the legends for details. It can be observed that the approximation factors are significantly better on the SOLGEN instances than on the UNIFORM instances.



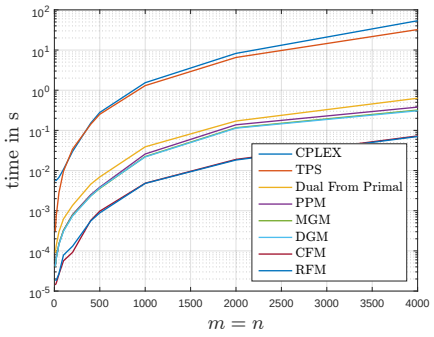
(a) Primal heuristics on SOLGEN.



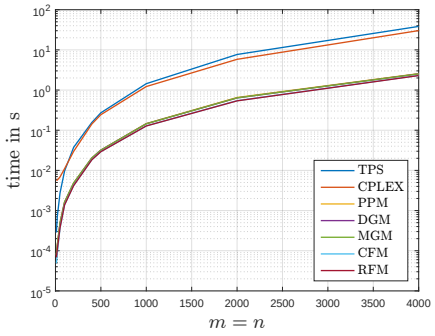
(b) Primal heuristics on UNIFORM.



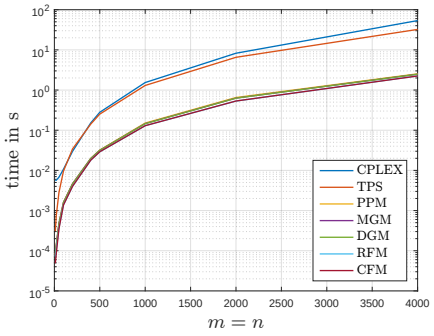
(c) Dual heuristics on SOLGEN.



(d) Dual heuristics on UNIFORM.

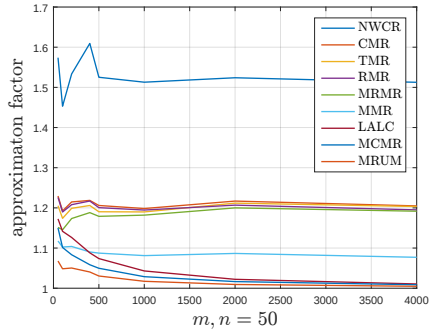


(e) Primal from dual heuristics on SOLGEN for different dual methods (see Section 3.4).

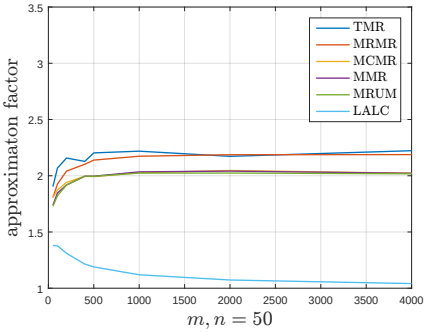


(f) Primal from dual heuristics on UNIFORM for different dual methods (see Section 3.4).

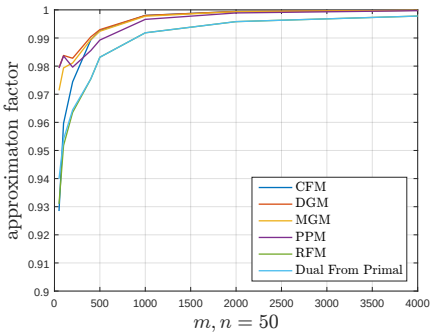
Figure 2: Average computation time for the problem classes SOLGEN and UNIFORM on **symmetric** problems, averaged over 100 problem instances. The positions of the heuristics in the legend corresponds to the positions of the respective graph in the figure. Observe that all heuristics are at least ten times fast than the exact solvers. LALC* denotes the suboptimal implementation of LALC. An optimal implementation should realize a computation time close to MMR.



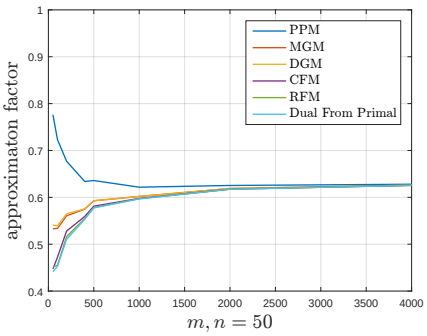
(a) Primal heuristics on SOLGEN.



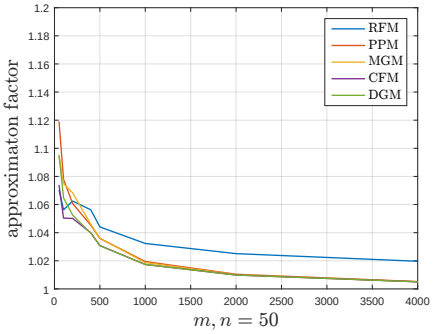
(b) Primal heuristics on UNIFORM.



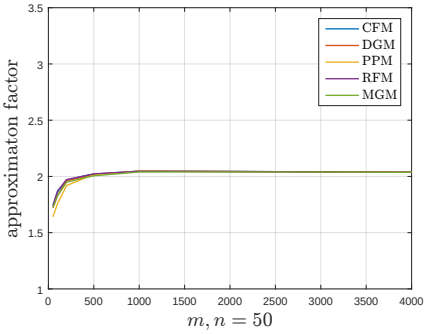
(c) Dual heuristics on SOLGEN.



(d) Dual heuristics on UNIFORM.

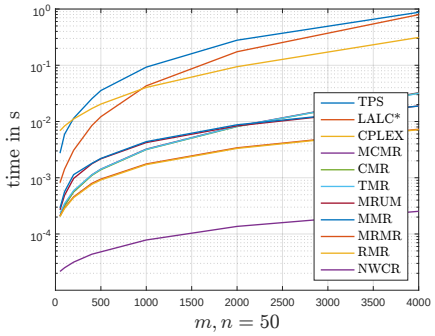


(e) Primal from dual heuristics on SOLGEN for different dual methods (see Section 3.4).

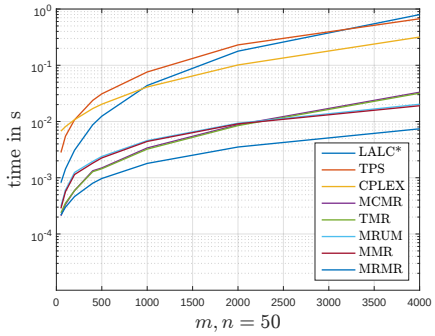


(f) Primal from dual heuristics on UNIFORM for different dual methods (see Section 3.4).

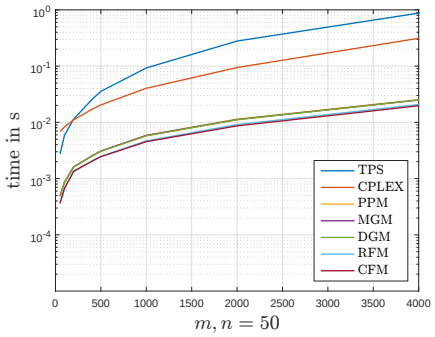
Figure 3: Approximation factors for the problem classes SOLGEN and UNIFORM on **asymmetric** problems. For fixed $n = 50$ and increasing $m \geq 50$ the evaluation points were averaged over 100 problem instances. The approximation factor is given by $\frac{val}{opt}$ where *val* is the function value of the solution of the heuristic and opt is the optimal function value for the problem. The positions of the heuristics in the legend corresponds to the position of the respective graph in the figure. Note that we omitted certain heuristics such as the NWCR and MRR in some figures when their approximation factors where far worse than the rest, see the legends for details. It can be observed again that the approximation factors are significantly better on the SOLGEN instances than on the UNIFORM instances.



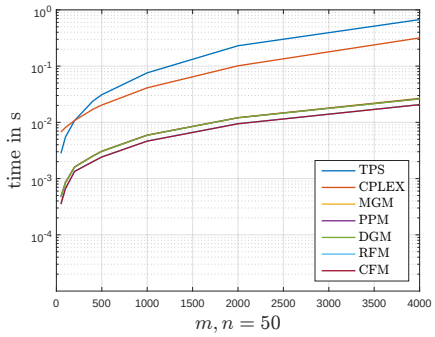
(a) Primal heuristics SOLGEN



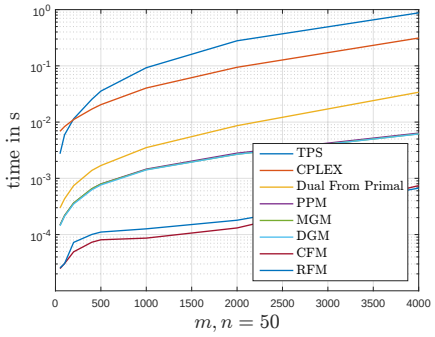
(b) Primal heuristics UNIFORM



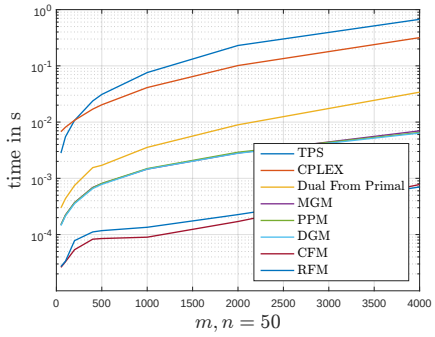
(c) Dual to primal heuristics SOLGEN



(d) Dual to primal heuristics UNIFORM



(e) Dual heuristics SOLGEN



(f) Dual heuristics UNIFORM

Figure 4: Average computation time for the problem classes SOLGEN and UNIFORM on **asymmetric** problems. For fixed $n = 50$ and increasing $m \geq 50$ the evaluation points were averaged over 100 problem instances. The positions of the heuristics in the legend corresponds to the positions of the respective graph in the figure. Observe that all heuristics are at least ten times faster than the exact solvers. LALC* denotes the suboptimal implementation of LALC. An optimal implementation should realize a computation time close to MMR.

- Otherwise, results generally resemble the symmetric case, with the exception that we observe better approximation quality of the Modified Column Minimum Rule and the Column First Method compared to their row equivalents on the SOLGEN instances.

- Finally, the approximation factor of the Pull Push Method drops for asymmetric problems, whereas the other dual heuristics better approximate the optimal value.

In summary, in accordance to the findings of Ross *et al.* (1975), we conclude that for randomly generated problems smaller standard deviation in the cost coefficients results in problems that are solved faster by exact solvers. Further, we can observe that these problems are also better approximated by the heuristics. This matches the intuitive notion that due to less variation in the cost coefficients the number of different optimal (and close to optimal) solutions increases. In turn, the chance to heuristically produce good solutions increases and reduces thus the time to find one of the optimal solutions for the exact solver.

The DOTMARK instances demonstrate that good performance on randomly generated problems does not generally transfer to specifically structured problems. For this problem class, the classic heuristics fail to produce relevant approximation factors and one should therefore resolve to the specifically tailored methods (see, e.g., Schrieber *et al.*, 2017). We highlight this in Table 3, where we present results for a selection of heuristics including the best and worst performing ones. As for the dual methods, the bad performance is easily explained, since by construction the cost matrix of the DOTMARK problems is zero on the diagonal which contradicts the concept of iteratively increasing the dual variables to compute spanning solutions consisting of non-negative dual variables only. This shortcoming was addressed in the Pull Push Method, where negative dual variables are possible. Although the method actually did not provide better results on the DOTMARK instances, it performed surprisingly well on other problem classes. Finally, as a negative result, the idea to use primal information to produce good dual solutions, as introduced in Section 3.5, fails and does not yield a satisfactory performance.

5 Conclusion

In this paper we have analyzed the effectiveness of heuristics for the transportation problem. For this purpose we have considered a selection of the most efficient (in terms of computation time) commonly used primal heuristics, together with some novel suggestions for similarly time efficient dual heuristics. For the specific case of Minimum Rules, it was observed in accordance with Gottschlich & Schumacher

Time in s	**CPLEX**	**TPS**	**NWCR**	**MRMR**	**MRUM**	**LALC**	**RFM**
WhiteNoise	1,69	2,13	1,4 E-04	0,03	0,15	0,20	4,8 E-03
GRFrough	1,68	3,17	1,4 E-04	0,03	0,15	0,19	4,8 E-03
GRFmoderate	1,66	4,64	1,4 E-04	0,03	0,15	0,19	4,8 E-03
GRFsmooth	1,65	4,94	1,4 E-04	0,03	0,15	0,18	4,8 E-03
LogGRF	1,71	5,57	1,4 E-04	0,03	0,15	0,18	4,8 E-03
LogitGRF	1,71	6,48	1,4 E-04	0,03	0,15	0,19	4,8 E-03
CauchyDensity	1,63	3,69	1,4 E-04	0,03	0,15	0,17	4,8 E-03
Shapes	0,96	0,36	8,5 E-05	5,0 E-03	0,04	0,04	1,1 E-03
ClassicImages	1,67	3,70	1,4 E-04	0,03	0,15	0,19	4,8 E-03
MicroscopyImages	1,24	1,34	1,1 E-04	0,01	0,09	0,12	2,8 E-03
Approximation factor	**CPLEX**	**TPS**	**NWCR**	**MRMR**	**MRUM**	**LALC**	**RFM**
WhiteNoise	1,00	1,00	185,57	8,82	4,79	8,59	0,00
GRFrough	1,00	1,00	132,11	10,66	3,56	9,72	0,00
GRFmoderate	1,00	1,00	28,07	5,75	1,76	5,63	0,00
GRFsmooth	1,00	1,00	11,12	3,98	1,42	3,65	0,00
LogGRF	1,00	1,00	4,11	2,24	1,26	2,11	0,00
LogitGRF	1,00	1,00	12,29	3,76	1,57	3,39	0,00
CauchyDensity	1,00	1,00	5,98	2,98	1,30	2,83	0,00
Shapes	1,00	1,00	8,91	3,87	1,67	3,68	0,47
ClassicImages	1,00	1,00	43,80	6,37	2,01	6,03	0,00
MicroscopyImages	1,00	1,00	6,38	2,75	1,38	2,55	0,01

Table 3: Computation time and approximation quality for a selection of heuristics on the DOTMARK instances. The Modified Russel's Method delivered the best overall approximation factor. The primal North-West Corner Rule and the dual Row First Method performed worst in terms of approximation quality.

(2014) that these can be significantly speed up by sorting the cost coefficients in advance. It was demonstrated that all primal and dual heuristics can be implemented to run in at most 10% of the time it takes an exact solver to find an optimal solution. As main advantage of simultaneously running primal and dual heuristics, a certificate of the quality of both heuristic solutions can be obtained by weak duality.

In summary, heuristics fail to deliver satisfactory results on most problem classes. Still, low computation time might justify the use of these heuristics before looking for optimal solutions, especially when the primal heuristic can be used as starting point for an LP solver anyway. Only in the case of specifically randomly generated problems the use of primal and dual heuristics delivers solutions with function values within a few percent of the optimal value. Our results highlight again the importance of testing transportation heuristics especially on problem instances originating from the specific application they are intended for.

Furthermore, we investigated the use of using dual information in primal heuristics and vice versa. Here, it was observed that the latter concept failed to deliver satisfactory results, whereas the former concept delivered acceptable primal solutions at least for the randomly generated problem instances.

In summary it is very hard to suggest a one-heuristic-fits-all method. Based on our study it seems reasonable to suggest using one of the primal heuristics using dual information, together with the Large Amount Least Cost Method to produce upper bounds. This should be complemented by one of the dual heuristics using sorting schemes for corresponding lower bounds, before exact solvers are invoked.

For our numerical experiments, we also improved upon the work of Jeffrey & James (1994) and suggest a more consistent problem generator called SOLGEN. This generator can produce problem instances for the transportation problem together with the corresponding optimal solution. We ensure that the topological structure of the solution is indeed uniformly distributed by employing random walks on complete bipartite graphs. Further, uniform distributions are obtained by sampling from Dirichlet distributions instead of inconsistently scaling uniform random variates.

Acknowledgements

We would like to thank Jörn Schrieber for directing us to the DOTMARK benchmark problems.

References

AHRENS, J. H. & FINKE, G. (1980) Primal transportation and transshipment algorithms. *Zeitschrift für Operations Research*, **24**, 1–32.

AHUJA, R. K., MAGNANTI, T. L. & ORLIN, J. B. (1993) *Network Flows – Theory, Algorithms, and Applications*. New Jersey: Prentice Hall.

BRENNER, U. (2008) A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. *Oper. Res. Lett.*, **36**, 408–413.

BRODER, A. (1989) Generating random spanning trees. *Foundations of Computer Science*, **30th Annual Symposium on Foundations of Computer Science**.

DANTZIG, G. B. (1963) *Linear Programming and Extensions*. Princeton: Princeton University Press.

GASS, S. I. (1990) On solving the transportation problem. *The Journal of the Operational Research Society*, **41**, 291–297.

GLOVER, F., KARNEY, D., KLINGMAN, D. & NAPIER, D. (1974) A computation study on start procedures, basis change criteria, and solution algorithms for transportation problems. *INFOR*, **20**, 793 – 813.

GLOVER, F., KLINGMAN, D. & BARR, R. (1979) Enhancements of spanning tree labelling procedures for network optimization. *Management Science*, **17**, 16–34.

GLOVER, F. & KLINGMAN, D. (1972a) Double-pricing dual and feasible start algorithms for the capacitated transportation (distribution) problem. *Applied Mathematics for Management*, **AMM-20**.

GLOVER, F. & KLINGMAN, D. (1972b) Higher order tree dual approximation methods for the distribution problem. *Management Science*, **18**, 574 – 583.

GOTTSCHLICH, C. & SCHUMACHER, D. (2014) The shortlist method for fast computation of the earth mover's distance and finding optimal solutions to transportation problems. *PLoS ONE*, **9**.

HITCHCOCK, F. L. (1941) The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, **20**, 224–230.

HOUTHAKKER, H. S. (1955) On the numerical solution of the transportation problem. *Operations Research*, **3**, 210–214.

JEFFREY, L. A. & JAMES, O. F. J. (1994) An algorithm for generating minimum cost network flow problems with specific structure and known optimal solutions. *Networks − An International Journal*, **24**, 445–454.

KANTOROVICH, L. W. (1942) On the translocation of masses. *Comptes Rendus (Doklady) de l'Académie des Sciences de l'U.R.S.S.*, **37**, 199–201.

KLEINSCHMIDT, P. & SCHANNATH, H. (1995) A strongly polynomial algorithm for the transportation problem. *Mathematical Programming*, **68**, 1–13.

KOOPMANS, T. C. (1948) Optimum utilization of the transportation system. *Econometrica*, **17**, 136–146. Supplement: Report of the Washington Meeting.

KOVÁCS, P. (2015) Minimum-cost flow algorithms: An experimental evaluation. *Optimization Methods and Software*, **30**, 94–127.

MONGE, G. (1781) Mémoire sur la théorie des déblais et des remblais. *De l'Imprimerie Royale*.

REINFELD, N. V. & VOGEL, W. R. (1958) *Mathematical Programming*. New Jersey: Prentice Hall, Inc.

ROSS, G. T., KLINGMAN, D. & NAPIER, A. (1975) A computational study of the effects of problem dimensions on solution times for transportation problems. *J. ACM*, **22**, 413–424.

RUSSEL, E. J. (1969) Letters to the editor − extension of Dantzig's algorithm to finding an initial near-optimal basis for the transportation problem. *Operations Research*, **17**, 187–191.

SANDI, C. (1986) *On a nonbasic dual method for the transportation problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 65–82.

SCHRIEBER, J., SCHUHMACHER, D. & GOTTSCHLICH, C. (2017) Dotmark – 2013; a benchmark for discrete optimal transport. *IEEE Access*, **5**, 271–282.

SCHRIJVER, A. (2002) On the history of the transportation and maximum flow problems. *Mathematical Programming*, **91**, 437–445.

SRINIVASAN, V. & THOMPSON, G. (1973) Benefit-cost analysis of coding techniques for the primal transportation algorithm. *Journal of the ACM*, **20**.

Appendix A

SOLGEN

Main idea Our approach resembles the heuristic for constructing minimum cost flow problems presented in Jeffrey & James (1994). The essential idea is as follows: One begins by randomly generating a base B for given dimensions m and n of a transportation problem. Subsequently, a basic primal solution x corresponding to B together with a dual solution (u,v) and the problem parameters C, a and b are chosen such that (C,a,b) constitutes a balanced (feasible) transportation problem and x and (u,v) are primal and dual feasible and satisfy the complementary slackness conditions (2.3).

In the special case of the transportation problem, there holds a one-to-one correspondence between bases (of basic solutions) and spanning trees on bipartite graphs (see, e.g., Ahuja *et al.*, 1993). In this sense, the base B consists of all edges in the spanning tree connecting the two disjoint sets of nodes in the bipartite graph. Thus B defines a topological structure of the optimal solution from which the problem is later constructed. Jeffrey & James (1994) make use of this equivalence and propose to compute bases B by a heuristic that randomly generates spanning trees. However, they do not prove any statistical properties of their approach. We propose to generate these spanning trees by employing random walks on bipartite graphs (see, e.g., Broder, 1989). This modification adds statistical value to the algorithm by ensuring uniformly distributed spanning trees and thereby uniformly distributed bases for the optimal solution of the transportation problem.

Problem generation The rest of our algorithm is similar to the work of Jeffrey & James (1994) with the exception that we give specific probability distributions

for all randomly generated values:

$$u_i := \mathscr{U}([-0.5, 0.5]) \text{ for all } i = 1, \ldots, m \tag{.1}$$

$$v_j := \mathscr{U}([-0.5, 0.5]) \text{ for all } j = 1, \ldots, n \tag{.2}$$

$$c_{ij} := \begin{cases} u(i) + v(j) & \text{for } (i,j) \in B \\ u(i) + v(j) + \mathscr{U}([0,1]) & \text{otherwise} \end{cases} \tag{.3}$$

$$x_{ij} := \begin{cases} p_k & \text{for the } k\text{-th entry } (i_k, j_k) \text{ in } B \\ 0 & \text{otherwise} \end{cases} \tag{.4}$$

$$a_i := \sum_{i:(i,j)\in B} x_{ij} \tag{.5}$$

$$b_j := \sum_{j:(i,j)\in B} x_{ij} \tag{.6}$$

where $\mathscr{U}([a,b])$ denotes the uniform distribution on the interval $[a,b]$. Furthermore, as for the UNIFORM instances, we ensure $\sum_i a_i = \sum_j b_j = 1$ by uniformly generating a vector $p = (p_1, \ldots, p_{m+n-1})^\top$ on the $(m+n)$-simplex by means of a Dirichlet distribution.

Then, by construction, x and (u,v) are a primal-dual optimal pair for the generated problem since they are primal and dual feasible and satisfy the complementary slackness conditions (2.3). Since the entries of the cost matrix are generated by the sum of three uniform distributions (i.e. an Irwin-Hall distribution) their standard deviation is lower compared to the UNIFORM problems, which generally results in easier-to-solve problem instances (see Section 4). For the sake of comparability we employ a final step were we shift the cost matrix C, i.e. we set $c_{ij} - \min_{ij} c_{ij}$ in the case that there exist negative values $c_{ij} < 0$ and finally scale the cost matrix by dividing by $\max_{ij} c_{ij}$.

Primal and dual degeneracy in SOLGEN One can easily adjust the SOLGEN generator in a way that the user can specify the primal and dual degeneracy of the optimal solution. Thus, primal degeneracy can be achieved by only setting a portion of the basic variables to non-zero values in (A.4) whereas dual degeneracy can be achieved by setting $c_{ij} := u_i + v_j$ for a specified number of non-basic dual constraints.

Hardware and software choice

All tests were performed on a standard personal computer (processor: Intel Core i5-4090, 3.30 GHz, RAM: 16GB). The implementation of all methods was carried

out in MATLAB 2015B and we used the Network Simplex of IBM's ILOG CPLEX 12.6.2 for MATLAB. All codes are available upon request to the authors.