DOI: 10.1478/AAPP.971A1

AAPP | Atti della Accademia Peloritana dei Pericolanti
Classe di Scienze Fisiche, Matematiche e Naturali
ISSN 1825-1242

Vol. 97, No. 1, A1 (2019)

SYMMETRIC TRAVELING SALESMAN PROBLEM AND FLOWS IN HYPERGRAPHS: NEW ALGORITHMIC POSSIBILITIES

TIRU S. ARTHANARI *

ABSTRACT. The traveling salesperson problem (TSP) is a very well-known NP-hard combinatorial optimization problem. Many different integer programming formulations are known for the TSP. Some of them are "compact", in the sense of having a polynomially-bounded number of variables and constraints. We focus on one compact formulation for the symmetric TSP, known as the "multistage insertion" formulation [T. S. Arthanari, *Discrete Mathematics* **306**, 1474 (2006)], and show that its linear programming (LP) relaxation can be viewed as a minimum-cost flow problem in a hypergraph. Using some ideas of R. Cambini, G. Gallo and M. G. Scutellà [University of Pisa, TR-1/92 (1992)] on flows in hypergraphs, we propose new algorithms to solve the LP relaxation. We also exploit the Leontief structure of a certain subproblem, to provide additional algorithms for solving the LP relaxation. Some bounds on the running time are also derived.

1. Introduction

Given a complete graph of n-vertices, finding a minimum cost *n-tour* (or tour for short) that starts from the home city and visits every city once and returns back to the home city, with the cost of traveling from city i to city j being same as that of traveling from city j to city i, that is, $c_{ij} = c_{ji}$, is called the Symmetric Traveling Salesman Problem (STSP). This is a typical, difficult combinatorial optimization problem, and has been extensively studied (Lawler *et al.* 1985; Laporte 1992; Jünger *et al.* 1997; Applegate *et al.* 2006; Gutin and Punnen 2007; Punnen 2007).

The standard formulation due to Dantzig *et al.* (1954) renders the asymmetric traveling salesman problem (ATSP), where the cost of traveling from city i to city j can be different from that of city j to city i, as a 0-1 integer program. Analogously the STSP can be formulated as an integer programming problem (see section **2** for notations):

$$\text{minimize} \sum_{e \in E_n} c_e x_e \tag{1}$$

$$\text{subject to} \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V_n, \tag{2}$$

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1, \ \emptyset \neq X \subset V_n, \tag{3}$$

$$x_e \in \{0,1\}, \forall e \in E_n. \tag{4}$$

To arrive at this formulation, Dantzig, Fulkerson and Johnson reason that given any tour the corresponding subgraph satisfies the degree constraints (Eq. 2). But subtours of any k vertices also satisfy these constraints; so to eliminate subtours from consideration in the model, we require that for any nonempty subset X of k-vertices ($k < n$), the number of edges of the tour incident with this subset is strictly less than k. Constraints 3 ensure this, and so they are called subtour elimination constraints. Dantzig, Fulkerson and Johnson also solved a large size TSP to optimality by sequentially adding cutting planes (inequality constraints) to separate the current non-integer solution from the feasible region of the current linear programming problem. This work became seminal for the approaches that followed to solve the STSP. Solving such an optimization problem efficiently is a theoretical challenge (Garey and Johnson 1990).

Polyhedral combinatorics deals with the study of polytopes with the corner points (vertices) corresponding to objects of interest like the tours (Korte and Vygen 2012). A polyhedral approach, like the branch and cut, was used on the largest STSP problem solved to date having 85900 cities using the *Concorde* code (*Concorde TSP Solver* 2018) developed by Applegate *et al.* (2006).

A brief account of polyhedral analysis of different formulations of TSP is provided in subsection **2.1**. Naddef (2007) succinctly summarizes the comparative strengths of the different models for STSP. Among all the known integer linear programming models, five models emerge and attain the same value for their linear relaxations. These are: the *subtour elimination* formulation of Dantzig, Fulkerson and Johnson, which has an exponentially large number of constraints, Wong's multicommodity flow formulation (Wong 1980), Claus's formulation (Claus 1984), the multistage insertion formulation (MI-formulation) (Arthanari 1982), and Carr's *cycle shrink* formulation (Carr 1995). The multistage-insertion is inspired by dynamic programming recursion - building up a tour step by step. The cycle shrink does the opposite: going from a tour to a node. Not surprisingly, these two formulations are equivalent (see the proof in Arthanari and Usha 2001).

A proof that Wong's (1980) and Claus's (1984) formulations give the same bound as DFJ can be found in (Langevin *et al.* 1990). There are formulations that give even stronger lower bounds (see Gouveia and Voß 1995; Öncan *et al.* 2009; Godinho *et al.* 2014). But they were found to be computationally not efficient (see section **2.1**). However, MI-formulation has additional polyhedral properties, which are not shared by other formulations. The integer hull of MI-formulation is the Pedigree polytope. A pedigree is defined as the sequence, $(e_4, \ldots, e_n)$ of $n-3$ edges selected for inserting nodes $4, \ldots, n$ respectively. In other words, any feasible integer solution to MI-formulation corresponds to a pedigree and vice versa.

The convex set, whose vertices are the $0-1$ vectors representing pedigrees, is called the pedigree polytope. With the view to understand better the integer hull of MI-formulation, Pedigree polytope and its properties are studied by Arthanari (2005, 2006, 2008) and Haerian Ardekani and Arthanari (2008).

Haerian Ardekani (2011) as part of her doctoral thesis has compared different formulations of the symmetric traveling salesman problem, including the standard formulation or DFJ formulation with respect to (i) the linear programming (LP) relaxation of the formulation and integrality gap, (ii) number of simplex iterations and (iii) CPU time used to find an optimal continuous solution. It turns out that MI-relaxation has emerged more often than not as the winner as far as the gap is concerned. Also it has shown superiority among those formulations with similar gap by needing fewer number of simplex iterations. Gubb (2003) compared 19 formulations of STSP that model the problem from different perspectives, namely, flows, insertions and subtours (some of these are in the list of references: Miller *et al.* 1960; Gavish and Graves 1978; Fox *et al.* 1980; Wong 1980; Claus 1984; Sherali and Driscoll 2002; Sarin *et al.* 2005). He concludes that even among those formulations that are similar in polytope-wise implications, they vary in computational efficiency. MI-formulation stands out in this experimental comparison as well. Unfortunately, in both (Haerian Ardekani 2011) and (Gubb 2003) the sizes of the instances from TSPLIB (Reinelt 1991) considered are less than 320. The reason for this limitation arises primarily from the capacity of the commercial LP solver software used. MI-formulation has $\frac{n(n-1)}{2}+(n-3)$ constraints and $\tau_n=\sum_{k=4}^{n}\frac{(k-1)(k-2)}{2}$ variables. In order to solve larger size problems one needs to abandon using general purpose LP algorithms to solve the LP instances that are sparse, with $0,\pm1$-matrices, with the non-zero elements occurring in specific positions that can be given by a formula. In this paper we consider the constraint matrix of MI-formulation for further clue to devise special purpose LP algorithms to exploit completely the structure of the problem matrix.

Section **2** introduces the notations and preliminaries required. Brief review of the literature on comparisons of different formulations of TSP is given in subsection **2.1**. Subsection **2.2** introduces concepts from flows on hypergraphs, followed by a brief overview of the Lagrangian relaxation approach and its variants in solving integer programming problems in subsection **2.3**. Subsection **2.4** introduces Leoitief substitution flow problems. Subsections **2.5** and **2.6** introduce MI-formulation and pedigree polytope respectively. Section **3** discusses the Leontief Substitution flow problem ($LSfP$) and the value iteration algorithm of Jeroslow *et al.* (1992) and the new observations made on MI-relaxation problem that relate a sub problem of MI-relaxation to $LSfP$. Section **3.1** applies Lagrangian relaxation to MI-formulation using the results from previous sections. Section **4** shows how MI-relaxation problem can be viewed as a minimum cost hypergraph flow problem and specializes the hypergraph simplex method of Cambini *et al.* (1992) for solving MI-relaxation problem. Section **5** gives an account of the computational experiments planned to evaluate the new algorithms suggested in the paper. The last section is devoted to summarize the findings and to conclude the paper.

2. Notations and preliminaries

Let R denote the set of reals. Similarly Q, Z, N denote the rationals, integers and natural numbers respectively, and B stands for the binary set of $\{0, 1\}$. Let R_+ denote the set of non negative reals. Similarly the subscript $_+$ is understood with rationals. Let R^d denote the set of d -tuples of reals. Similarly the superscript ^{d} is understood with rationals, etc.

Let $K_n = (V_n, E_n)$ be the complete graph of $n \geq 4$ vertices, where $V_n = \{1, \ldots, n\}$ is the set of vertices labeled in some order, and $E_n = \{e = (i, j) \mid i, j \in V_n, i < j\}$ is the set of edges. We denote the elements of E_n by e where $e = (i, j)$. Notice that, unlike the usual practice, an edge is assumed to be written with $i < j$. Let p_k denote $|E_k| = k(k-1)/2$. Let the elements of E_n be labeled as follows: $(i, j) \in E_n$, has the label, $l_{ij} = p_{j-1} + i$.

This means, edges $(1,2), (1,3), (2,3) \in E_3$ are labeled, $1, 2$, and 3 respectively. Once the elements in E_{n-1} are labeled then the elements of $E_n \setminus E_{n-1}$ are labeled in increasing order of the first coordinate, namely i. Let $\tau_n = \sum_{k=4}^{n} p_{k-1}$.

Assume that the edges in E_n are ordered in increasing order of the edge labels. For a subset $F \subset E_n$ we write the *characteristic* vector of F by $x_F \in R^{p_n}$ where

$$x_F(e) = \begin{cases} 1 & \text{if } e \in F, \\ 0 & \text{otherwise.} \end{cases}$$

For a subset $S \subset V_n$ we write

$$E(S) = \{(i, j) \mid (i, j) \in E, i, j \in S\}.$$

Given $u \in R^{p_n}$, $F \subset E_n$,we define,

$$u(F) = \sum_{e \in F} u(e).$$

For any subset S of vertices of V_n, let $\delta(S)$ denote the set of edges in E_n with one end in S and the other in $S^c = V_n \setminus S$. For $S = \{i\}$, we write $\delta(\{i\}) = \delta(i)$.

A subset H of E_n is called a *Hamiltonian cycle* in K_n if it is the edge set of a simple cycle in K_n, of length n. We also call such a Hamiltonian cycle an $n-tour$ in K_n. At times we represent H by the vector $(1 i_2 \ldots i_n 1)$ where $(i_2 \ldots i_n)$ is a permutation of $(2 \ldots n)$, corresponding to H. Let $\mathscr{H}_n$ denotes the set of all *Hamiltonian cycles* (or $n-tours$) in K_n.

Let $\mathbf{E}$ be a finite set, called the ground set. Let $\mathscr{F}$ denote a collection of subsets of $\mathbf{E}$. Let $c : \mathscr{F} \to \mathbf{R}$ denote a cost function. In an abstract way, a combinatorial optimization problem (COP) can be posed as: Find a $X \in \mathscr{F}$ that minimizes $c(X)$.

Let $\{0,1\}^{|E|}$ denote the set of all $0-1$ vectors indexed by $\mathbf{E}$. Since any subset of $\mathbf{E}$ can be given by a $0-1$ vector, called the incidence vector, the collection $\mathscr{F}$ can be equivalently given by a subset F of $\{0,1\}^{|E|}$. And the convex hull of F, denoted by $conv(F)$, is a $0-1$ polytope. And the set of vertices of the polytope can be seen as F. We specify a combinatorial optimization problem by giving $(\mathbf{E}, F, c)$. For example, finding a Hamiltonian cycle that minimizes a linear objective function over the set of all Hamiltonian cycles (or $n-tours$) in K_n is a COP. This problem is also known as the *symmetric traveling salesman problem* (STSP).

Here the ground set $\mathbf{E}$ is the set of edges in a complete graph on n vertices, E_n. F is the set of incidence vectors of $H \in \mathscr{H}_n$. And we are given $c \in R^{p_n}$. Let Q_n denote the polytope $conv(F)$. STSP is a typical COP which is known to be *NP*-hard. In polyhedral

combinatorics, Q_n is studied while solving the STSP (see Lawler *et al.* 1985; Jünger *et al.* 1997)). Next we take a quick survey of comparison of various formulations of TSP, based on computational and polyhedral perspectives.

2.1. A brief review of comparisons of different formulations of STSP. Formulations of any integer liner programming problem, can be compared by using polyhedral information. Suppose two different formulations F_1 and F_2 are stated in the same space of variables $x \in R^p$ and the problem is to find an optimal integer solution that minimizes an objective function. Let $P(F_1)$ and $P(F_2)$ be the polyhedra associated with these formulations. If $P(F_1) \subset P(F_2)$, then F_1 is a better formulation than F_2 since the lower bound obtained by solving the LP relaxation of F_1 is as good as or better than the one obtained by solving the LP relaxation of F_2. In case F_3 is a formulation having the polyhedron

$$P(F_3) = \{(x,y) \in R^p \times R^q : Ax + By \leq b\},$$

where A, B and b have same number of rows, we can use the projection of $P(F_3)$ into the subspace of x variables given by

$$TP(F_3) = \{x \in R^p : \exists\, y \in R^q \ \ni \ (x,y) \in P(F_3)\}$$

to compare it with the formulation F_1. That is, if $TP(F_3) \subset P(F_1)$, then F_3 is a better formulation.

Such polyhedral analyses have been used in the literature to compare a number of existing formulations (Wong 1980; Padberg and Sung 1991; Gouveia and Voß 1995; Arthanari and Usha 2001; Orman and Williams 2007; Öncan *et al.* 2009; Roberti and Toth 2012; Godinho *et al.* 2014). In addition to their comparative study using polytope analysis, Öncan *et al.* (2009) report some results on the empirical quality of the LP bounds obtained with some of the formulations. They tested these formulations on 10 ATSP instances with number of cities ranging from $30-70$, obtained from TSPLIB (Reinelt 1991). They use the relative deviations from the optimal tour lengths, computed as $100(z^*_{IP} - z^*_{LP})/z^*_{IP}$, where z^*_{IP} is the length of an optimal tour and z^*_{LP} is the lower bound obtained by solving the LP relaxation of the models. In their study, the formulations by Sherali *et al.* (2006) and Claus (1984) provide the best bounds. These comparisons are primarily for ATSP formulations.

Gubb (2003) considers the polynomial formulations discussed by Öncan *et al.* (2009) and three new symmetric formulations and presents computational comparisons on these formulations, including MI-formulation. Randomly generated Euclidean TSP instances with $n = 100$ were tested along with 10 instances, with number of cities ranging from 30- 70 from TSPLIB (Reinelt 1991). These results were compared with the implications of polyhedral analysis, and the results show that polyhedral analysis by itself does not necessarily show which formulations are faster in practice. Polyhedral analysis gives one aspect, the strength of the LP relaxation, without any insight on the time taken to solve the LP relaxation. Practical comparisons need both, and to link the relationship between them to fully solving for an optimal integral solution.

Gubb concludes based on his computational results that the best polynomial STSP formulation is Multistage Insertion formulation compared to the formulations studied by Öncan *et al.* (2009). In general, MI-formulation fully solved the instances and is the fastest; as it is not only as strong as DFJ formulation, but also has a small number of variables compared to other polynomial formulations that are equivalent to DFJ-formulation.

Haerian Ardekani (2011) compares the performance of LP relaxations of various TSP formulations with that of MI-formulation. Some STSP and ATSP instances from TSPLIB, and some diamond instances by Papadimitriou and Steiglitz (1977) are used for this purpose. MI-relaxation outperforms other formulations either in terms of solution time and number of iterations, or the quality of the solutions, on STSP instances from TSPLIB . The solution times and number of Cplex iterations for the MI-formulation are significantly lower than those of other formulations with the same LP relaxation value. Despite their lower solution times, the gap with the optimal solution given by the formulations of Fox *et al.* (1980) and Miller *et al.* (1960) are considerably higher than that of MI-formulation. The MI-formulation performs better than those of Carr (1996), Claus (1984), and Flood (1956) in terms of solution time and number of iterations on TSP instances from TSPLIB. Ardekani notes that for the ATSP instances reported by Gouveia and Pires (1999), the gap with the optimal solution for the multi commodity flow (MCF) formulation dominates that of the MI-formulation; however, solution times required by this formulation are notably greater than those of the MI-formulation.

Papadimitriou and Steiglitz designed a series of STSP instances to show how some local search algorithms, in particular the *k*–interchange heuristics, get stuck in local optima and fail to find the optimum. All of these instances have a single global optimum but exponentially many second best local optima that are all appreciably inferior to the global optimum. The instances are made up of some structures called diamonds, and are indexed by the number of diamonds. Ardekani observes that LP relaxation of MI-formulation found the integer solution to all the STSP diamond instances. MI-formulation also outperforms other formulations in terms of LP relaxation value or solution time. Some of the tables from (Haerian Ardekani 2011) are reproduced in Appendix **6**.

Roberti and Toth (2012) show that the branch and cut code *FLT* by Fischetti *et al.* (2003) and the *Concorde* code (*Concorde TSP Solver* 2018) by Applegate *et al.* (2006) were the only two exact methods based on the branch and cut polyhedral approach, that are able to solve, within the imposed time limit, all the 35 instances from TSBLIB and some real world instances to optimality, and are clearly better performing than the other five branch and bound methods considered by them. Largest problem size solved by them is 443. Also, they conclude that *Concorde* is the only one code available to solve very large size instances. *Concorde* and *FLT* work in the space of variables used in DFJ formulation. And such a cutting plane approach needs to ensure the algorithm is able to check whether any subtour elimination constraint is violated and it must be able to solve the LP relaxation efficiently as well. The facet (and adjacency) structure of the tour polytope contained in the subtour elimination polytope plays an important role in the computational difficulty faced by these cutting plane methods. Hence, a new beginning is possible if we choose, instead of working with tours, to work with pedigrees. MI-polytope contains the pedigree polytope and pedigree polytope has nicer adjacency structure (see subsection **2.11** on Pedigrees).

The main aim of this paper is to provide new algorithmic approaches that might appreciably increase the size of problems solved using MI-relaxation. This is the first step towards taking a different polyhedral approach based on Pedigree polytope to solve STSP problem. So we are not at this stage competing with *Concorde*, but suggest methods based on new results on MI-formulation to solve larger instances of MI-relaxation problem. That is, first step in using a polytope that contains Pedigree polytope, to solve STSP.

Next three subsections give a preliminary account of the concepts and tools used to derive new results on MI-relaxation.

2.2. Hypergraph and flows. Here we state some of the definitions and concepts from (Cambini *et al.* 1992) and related works.

Definition 2.1. A *directed hypergraph* is a pair $\mathscr{H} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, and $\mathscr{E} = \{e_1, e_2, \ldots, e_n\}$ is the set of *hyperarcs*. A hyperarc e is a pair (T_e, h_e), where $T_e \subset \mathscr{V}$ is the tail of e and $h_e \in \mathscr{V} \setminus T_e$ is its head. A hyperarc that is headless, $(T_e, \emptyset)$, is called a *sink* and a tailless hyperarc, $(\emptyset, h_e)$, is called a *source*.

Given a hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{E})$, a positive real multiplier $\mu_v(e)$ associated with each $v \in T_e$, a real demand vector b associated with $\mathscr{V}$, and a nonnegative capacity vector w, a flow in $\mathscr{H}$ is a function $f : \mathscr{E} \to R$ which satisfies:

$$\sum_{e \ni v = h_e} f(e) - \sum_{v \in T_e} \mu_v(e) f(e) = b(v), \ \forall v \in \mathscr{V} \qquad \text{(conservation)}, \tag{5}$$

$$0 \le f_e, \ \forall e \in \mathscr{E} \qquad \text{(feasibility)}, \tag{6}$$

$$f_e \le w(e), \ \forall e \in \mathscr{E} \qquad \text{(and capacity)}. \tag{7}$$

Problem **2.1**. [Minimum cost hypergraph flow problem] Let $c(e)$ be the cost associated with the hyperarc $e, \forall e \in \mathscr{E}$. Find f^* such that $\sum_{e \in \mathscr{E}} c(e) f^*(e)$ is a minimum over all f satisfying the flow constraints 5, 6, and 7.

Notice that this problem generalizes not only the minimum cost flow problem in ordinary networks, but also the minimum cost generalized flow problem.

A directed path P_{st} from s to t in $\mathscr{H}$ is a sequence $P_{st} = (v_1 = s, e_1, v_2, e_2, \ldots, e_q, v_{q+1} = t)$, where $s \in T_{e_1}$, $h_{e_q} = t$, and $v_i \in T_{e_i} \cap h_{e_{i-1}}$ for $i = 2, \ldots, q$.

If $s = t$, then P_{st}, is a *directed cycle*. When no directed cycle exists, then $\mathscr{H}$ is called a *cycle-free* hypergraph. A directed hyperpath, Π_{St} from the source set S to the sink t in $\mathscr{H}$ is a minimal cycle-free sub-hypergraph containing both the nodes in S and node t, and such that each node, with the exception of the nodes in S has exactly one entering hyperarc. A hyperarc e' is said to be a *permutation* of a hyperarc e if $T_e \cup \{h_e\} = T_{e'} \cup \{h_{e'}\}$. A hypergraph $\mathscr{H}'$ is a permutation of a hypergraph $\mathscr{H}$ if its hyperarcs are permutations of the hyperarcs of $\mathscr{H}$.

A *directed hypertree* with root set R, and a set of hyperarcs, E_T, called tree arcs, is a hypergraph $\mathscr{T}_R = (R \cup N, E_T)$ such that:

(1) $\mathscr{T}_R$ has no isolated vertices, and does not contain any directed cycle;
(2) $R \cap N = \emptyset$;
(3) Each node $v \in N$ has exactly one entering hyperarc; and
(4) No hyperarc has a vertex of R as its head.

We say that $\mathscr{T}_R$ is a directed hypertree rooted at R and N is called the set of *non-root* vertices. Any non-root vertex not contained in the tail of any tree arc is called a *leaf*. Any permutation of a directed hypertree rooted at R yields an undirected hypertree rooted at R. It can be shown that $\mathscr{T}_R$ is a directed hypertree rooted at R if and only if

- $\mathscr{T}_R$ has no isolated vertices,
- $R \cap N = \emptyset$ and $|N| = |E_T| = q$,

- an ordering $(v_1, v_2, ..., v_q)$ and $(e_1, e_2, ..., e_q)$ exists for the elements of N such that: $h_{e_j} = v_j$, and
- $R \cup \{v_1, v_2, \ldots, v_{j-1}\} \supseteq T_{ej}, \forall e_j \in E_T$.

2.3. Lagrangian relaxation and variants. This section briefly collects relevant results and concepts from Lagrangian relaxation approach to finding lower bounds for difficult combinatorial problems. Lagrangian relaxation approach, has found applications in several different areas of practical problems of scheduling (mixed integer and integer programming formulations of the problems are relaxed using this approach). Guignard (2003) gives many examples of this nature explaining various new developments and practical implementation tips.

The crucial among the research on Lagrangian relaxation for combinatorial optimization problems were the seminal papers for TSP by Held and Karp (1970, 1971). Firstly they showed the Lagrangian relaxation based on 1-tree for TSP gives the same bound as the LP relaxation of the standard formulation of TSP. Subsequently Held and Karp introduced the subgradient optimization to solve Lagrangian Dual. Computational experiments with subgradient optimization are known to be successful. Important in Held and Karp's use of Lagrangian relaxation is the creative identification of the sub problem of solving 1-tree minimization, which breaks down to solving a minimum spanning tree problem, that can be efficiently solved. In addition, the lower bound obtained (by solving LD) using 1-tree relaxation, is a tight lower bound for TSP, which goes by the name HK-bound (Valenzuela and Jones 1997). For solving large-scale convex optimization problems and for providing lower bounds on the optimal value of discrete optimization problems, researchers have effectively used Lagrangian relaxation and duality results. A key approach in providing efficient computational means to obtain near-optimal dual solutions and bounds on the optimal value of the original problem, since the work of Held and Karp (1971), has been *subgradient optimization* or methods based on subgradients. Geoffrion's paper (Geoffrion 1974) is an early work on Lagrangian relaxation and its use in $0-1$ programming problems (several other references can be found in Guignard 2003). But computational complexity of subgradient optimization is not known to be a polynomial; however, Bertsimas and Orlin (1994) showed the existence of fast polynomial algorithms for solving LD of LP problems, in some structured LP instances guaranteeing solution to the primal problem as well. Theoretically, their method is significantly faster than interior point methods and ellipsoid like methods directly applied to the problem.

Next we give the definitions and results for use in later parts of the paper. Let P be the problem $\min\{cx \mid Ax \leq b, Dx \leq d, x \geq 0, x \in \mathscr{X}\}$. Let $F(P)$ denote the set of feasible solutions to a given problem P. Let $F^*(P)$ be the set of optimal solutions to P. Let $z(P)$ be the optimal value of the problem P. Usually $\mathscr{X}$ imposes some integrality restrictions on the solutions of P.

Definition 2.2. A Lagrangian relaxation of (P) is either

$$\min_x\{cx + \lambda(Ax - b) \mid Dx \leq d, x \in \mathscr{X}\}$$

or

$$\min_x\{cx + \lambda(Dx - d) \mid Ax \leq b, x \in \mathscr{X}\}$$

depending on which of the set of constraints is relaxed using non negative λ, called Lagrangian multipliers.

To fix ideas, we generally relax those constraints that make the resulting Lagrangian problem easy to solve, we call the remaining constraints, *kept constraints*. Sometimes, both sets when relaxed may result in relatively easier Lagrangian problems. However, both relaxations provide lower bounds on the optimal value of P, $z(P)$. Let us assume that $Ax \leq b$ are relaxed unless otherwise specified.

The problem of finding supremum of such lower bounds leads us to Lagrangian Dual (LD) problem. Let us call LR_λ Lagrangian relaxation problem under consideration.

Definition 2.3. The LD problem

$$z(\text{LD}) = \max_{\lambda \geq 0} z(LR_\lambda)$$

is called the *Lagrangian Dual* of P, relative to the set of constraints relaxed, $Ax \leq b$.

Let $x(\lambda)$ denote an optimal solution of LR_λ for some $\lambda \geq 0$. If $x(\lambda)$ is feasible for P, then $cx(\lambda) + \lambda(Ax(\lambda) - b) \leq z(P) \leq cx(\lambda)$. If in addition $\lambda(Ax(\lambda) - b) = 0$ then $x(\lambda)$ is an optimal solution to P, as well, *i*.e., $z(P) = cx(\lambda)$. However, if instead of $Ax \leq b$, we had $Ax = b$, set of equality restrictions, then λ is not required to be nonnegative in Lagrangian problem. In that case if, $x(\lambda)$ is optimal for Lagrangian problem and is feasible for P then it is optimal for P as well, $z(P) = cx(\lambda)$.

Lagrangian relaxation bound is never worse than the LP bound. However, if $conv\{x \in \mathscr{X} \mid Dx \leq d\} = \{x \mid Dx \leq d\}$, then $z(\text{LP}) = z(\text{LD}) \leq z(P)$. This is an important result as, we don't hope to improve the bound if the relaxed problem has integral extreme points. So the reason we relax them emanates from the fact that solving the LP is hard either due to the number of constraints (exponentially many) or basis size (too large for the solver). In such cases solving the LD becomes an alternative approach to calculate the LP bound. In general, Lagrangian solutions, which are feasible integral for the kept constraints may violate one or more of the relaxed constraints. In such a situation, Lagrangian solutions could be used as input for some heuristics that achieve feasibility for the problem P. Barahona and Anbil (2000) give an averaging method to recover feasible solutions to the primal problem from the solutions to Lagrangian problems.

Suppose $\mathscr{C} = conv\{x \in \mathscr{X} \mid Dx \leq d\}$ is a polytope, then $\mathscr{C}$ has finitely many extreme points, $x^s, s \in S$, such that

$$z(\lambda) = \min\{cx + \lambda(Ax - b) \mid Dx \leq d, x \in \mathscr{X}\} = \min_{s \in S}\{cx^s + \lambda(Ax^s - b)\}.$$

$z(\lambda)$ is a piecewise linear concave function, so LD seeks a maximum of a concave function, $\max z(\lambda)$, that is not differentiable everywhere. At any breakpoint λ^0 of $z(\lambda)$, we have subgradients $y \ni$

$$z(\lambda) - z(\lambda^0) \leq \langle y, (\lambda - \lambda^0)\rangle,$$

where $\langle y, (\lambda - \lambda^0)\rangle = \sum_i y_i(\lambda_i - \lambda^0)$.

The set of all subgradients of a concave function $z(\lambda)$ at a point (λ^0) is called the subdifferential of the function z at the point λ^0 and is denoted by $\partial z(\lambda^0)$. From convex analysis we know the following facts, which we state without proofs:

Lemma 2.1. The subdifferential $\partial z(\lambda^0)$ is a nonempty, closed, convex and bounded set. And if $\partial z(\lambda^0)$ is a a singleton set then that is the gradient of $z(\lambda)$ at λ^0.

Lemma 2.2. If $z(\lambda)$ is not differentiable at λ^k, then $y^k = (Ax^k - b)^T$ is a subgradient of $z(\lambda)$ at λ^k. y^k is orthogonal to $\{\lambda \mid cx^k + \lambda(Ax^k - b) = \eta^k\}$ where $\eta^k = z(\lambda^k)$, the optimal value for Lagrangian problem corresponding to λ^k.

To solve LD, Held and Karp used subgradient optimization, in which iteratively LR_λ problems are solved for a sequence of λ_k, choosing the next λ_{k+1} moving a step along a subgradient of $z(\lambda)$ at λ_k. The step size is an important issue for practical convergence of the subgradient optimization method, though theoretically Poljak (1967, 1969) has shown that given $t_i \in R, i \in N \ni \sum_{i=1}^{\infty} t_i = \infty$ and $\lim_{i\to\infty} t_i \to 0$; and $\lambda_{i+1} = t_i\lambda_i$, we have $\lim_{\lambda_i\to\infty} z(LR_{\lambda_i}) = z(\mathrm{LD}) = \max_{\lambda\geq 0} z(LR_\lambda)$. The choice

$$\lambda_{k+1} = \lambda_k + \frac{s^k\, \varepsilon_k(\eta^* - \eta^k)}{||s^k||^2}$$

where s^k is the subgradient of $z(\lambda)$ at λ^k; $\varepsilon_k \in (0,2)$, and η^* is an estimate of the optimal value of LD, was suggested by Held and Karp (1971). Other researchers, as Reinelt (1994) and Valenzuela and Jones (1997), have given alternative step size choices. Subgradient methods use different step-size rules to update Lagrangian multipliers to ensure convergence. For solving nondifferentiable problems Poljak (1969) initiated such studies on their convergence properties; under various step-size rules convergence rates have been established by subsequent researchers.

Nedić and Ozdaglar (2009) report on recent great successes of subgradient methods in networking applications. They study methods for generating approximate primal solutions as a byproduct of subgradient methods applied to LD; they also provide theoretical bounds on the approximation. Recently, Lorena and Narciso (2002) and Zamani and Lau (2010) have used surrogate information and learning capability respectively in Lagrangian relaxation to outperform Held and Karp's subgradient approach for solving LD. Using step size depending on the previous iterations, with the capacity of expansion or contraction of the step size Zamani and Lau solve Euclidean TSP instances from TSPLIB (Reinelt 1991) and claim that the procedure was very effective.

To solve LD, in addition to subgradient methods and its variations, there are other approaches like the analytic centre cutting-plane method, Volume algorithm, and Bundle method (Guignard 2003; Lemaréchal 2003). As cited earlier, Barahona and Anbil (2000) give an averaging method called volume algorithm to recover feasible solutions to the primal problem from the solutions to Lagrangian problems.

2.4. Leontief substitution flow problems. We first state some definitions from Veinott Jr. (1968), the classical work on Leontief matrices. Consider $A \in R^{m\times n}, b \in R^m$ and $x \in R^n$.

Definition 2.4 (trivial rows)**.** The ith row (column) of a matrix A is called *trivial* if for every column vector $x \geq 0$ for which $Ax \geq 0$, the ith component of Ax (x) is zero; otherwise the ith row (column) is called *nontrivial*.

Definition 2.5 (pre-Leontief, Leontief matrices)**.** A matrix A is called *pre-Leontief* if each column has at most one positive element. A matrix is called Leontief if each column has exactly one positive element and its rows are nontrivial.

Definition 2.6 (pre-Leontief, Leontief Substitution Systems)**.** The system $Ax = b, x \geq 0$ with $b \geq 0$ is called pre-Leontief or Leontief substitution system according as the given matrix is pre-Leontief or Leontief.

Remarks:

(1) So the incidence matrix of a hypergraph is pre-Leontief.
(2) The hypergraph flow problem is a pre-Leontief substitution system in case the demand vector b is nonnegative. If in addition the rows are nontrivial, we have a corresponding Leontief substitution system.
(3) Conversely it is easy to see that given a pre-Leontief system, we can define a corresponding hypergraph flow problem as defined in subsection **2.2**. (Jeroslow *et al.* 1992, Lemma 1.1).
(4) The term *degenerate* hyperarc is used by Jeroslow *et al.* (1992) if $x(e) = 0$ corresponding to a hyperarc e in every basic feasible solution to the problem. (Trivial column as in Definition **2.4**.)

Definition 2.7 (Leontief substitution flow problem (Jeroslow *et al.* 1992))**.** Given $C \in R^{|\mathscr{E}|}$, and a Leontief matrix A, we call the minimum cost hypergraph flow problem given by minimize $\{Cf \mid Af = b, f \geq 0\}$ a Leontief substitution flow problem, in case $b \geq 0$.

Definition 2.8. (Jeroslow *et al.* 1992). Consider a hypergraph $\mathscr{H}$. Let $v_1, e_l, v_2, e_2, \ldots, e_k, v_{k+1}$ be a directed cycle where $v_{k+1} = v_1$ and $e_i = (T_i, v_{i+1})$, and $v_{i-1} \in T_i, i = 1, ..., k$. The gain of this directed cycle is defined by

$$1/\Pi_{i=1}^{k} \mu_{v_i}(e_i).$$

We call a Leontief flow problem defined on a hypergraph $\mathscr{H}$, gainfree if the gain of every directed cycle in $\mathscr{H}$ is ≤ 1.

Lemma 2.3. If the incidence matrix of a hypergraph $\mathscr{H}$ is integral then a Leontief flow problem defined on the hypergraph $\mathscr{H}$, is gainfree.

Proof: Proof follows from the definition of gain.

Integrality of the basic feasible solutions to Leontief substitution flow problems is known from Veinott's classical result:

Theorem 2.1. *(Veinott Jr. 1968). Let $Ax = b$, $x \geq 0$, be a given system. If A is an integral Leontief matrix, the following are equivalent:*

(1) *B^{-1} is integral for every $B \in \mathscr{B}^*$,*
(2) *$det(B) = 1$ for every $B \in \mathscr{B}^*$,*
(3) *The extreme points of $F(b)$ are integral for every nonnegative integral b,*

where (a) $F(b) = \{x \mid Ax = b, x \geq 0\}$, denotes the solution set of the given system,(b) for a pre-Leontief matrix A, let $\mathscr{B}$ be the set of all square full submatrices of A having pre-Leontief transpose and (c) $\mathscr{B}^*$ is the set of all Leontief matrices contained in $\mathscr{B}$ (for a proof see Veinott Jr. 1968, p. 17).

From the definition of unimodular matrices, we know that a unimodular matrix B is a square integer matrix with determinant +1 or -1. And we know that a matrix A is totally unimodular if every square non-singular submatrix is unimodular, *i*.e., every subdeterminant

of A is either $+1,-1$, or 0. So if A is a totally unimodular matrix then $a_{ij} = 0$, or ± 1. So pre-Leontief matrices do not satisfy this requirement in general. However, we can understand the implication of Veinott's result better by using the definition of totally dual integrality (TDI) of linear programming problems.

Definition 2.9 (Totally Dual Integral (Edmonds and Giles 1977))**.** Given an integral A and rational b the LP $min\{cx \mid Ax = b, x \geq 0\}$ is totally dual integral (TDI) if the dual problem has an integral optimal solution for every integer vector c for which it has an optimal solution.

Jeroslow *et al.* (1992) have shown that every Leontief substitution flow problem with an integral A and rational b is TDI. And using this they prove that if a Leontief substitution flow problem with integral A, b has an optimal solution then it has an integral optimal solution. Also they indicate how this result can be derived from Veinott's Theorem **2.1**.

2.5. The multistage insertion formulation. In this subsection we state the MI-formulation given by Arthanari (1982) and briefly outline some properties and its connection to pedigrees, and STSP (Arthanari 2005, 2006, 2008). The standard formulation of the STSP is due to Dantzig, Fulkerson and Johnson (DFJ) (Dantzig *et al.* 1954). As every tour is a 2 matching, and the converse is not true, as a 2 matching could correspond to a subtour, they give a $0-1$ programming formulation having n equalities and exponentially many inequalities that ensure subtours are eliminated from consideration. Given n, relaxing the integer restriction of the DFJ formulation we obtain the subtour elimination polytope (SEP(n)).

MI-formulation is based on constructing STSP tours by sequentially inserting nodes into the initial tour of three nodes 1, 2 and 3. Given graph K_n, starting with tour $T_3 = [1,2,3,1]$, nodes from 4 to n are inserted sequentially between the nodes of the sub-tour obtained until a complete tour of size n is achieved. For all $1 \leq i < j \leq k-1$ and $4 \leq k \leq n$, the decision variables of MI-formulation are defined as follows:

$$x_{ijk} = \begin{cases} 1, & \text{if node } k \text{ is inserted between nodes } i \text{ and } j, \\ 0, & \text{otherwise.} \end{cases}$$

We also use the equivalent notation $x_k(e)$ for x_{ijk} when $e = (i,j) \in E_{k-1}$.

Let c_{ij} be the cost of an edge $(i,j) \in E_n$. The insertion of some node k between nodes i and j, would replace the edge (i,j) with two new edges of (i,k) and (j,k) in the tour. This will increase the total cost of the tour by $C_{ijk} = c_{ik} + c_{jk} - c_{ij}$. The objective function of MI-formulation is to minimize the total incremental cost. MI-formulation (Arthanari 1982) is:

$$\min \sum_{k=4}^{n} \sum_{(1 \leq i < j \leq k-1)} C_{ijk} x_{ijk}$$

subject to:

$$\sum_{1\le i<j\le k-1} x_{ijk} = 1, \qquad 4 \le k \le n, \tag{8}$$

$$\sum_{k=4}^{n} x_{ijk} \le 1, \qquad 1 \le i < j \le 3, \tag{9}$$

$$-\sum_{r=1}^{i-1} x_{rij} - \sum_{s=i+1}^{j-1} x_{isj} + \sum_{k=j+1}^{n} x_{ijk} \le 0, \qquad 4 \le j \le n-1, 1 \le i < j, \tag{10}$$

$$x_{ijk} \in \{0,1\}, \qquad 4 \le k \le n, 1 \le i < j \le k-1. \tag{11}$$

Constraint (8) of the model guarantees that each node from 4 to n is inserted in some edge. Constraint (9) ensures that at most one node is inserted in each of the edges of T_3. Constraint (10) makes sure that a node is inserted into an edge of the subtour only if that edge has been generated through previous insertions and is available.

By relaxing the integer constraint from MI-formulation and also adding the redundant constraints

$$-\sum_{r=1}^{i-1} x_{rin} - \sum_{s=i+1}^{n-1} x_{isn} \le 0, i = 1, \ldots, n-1, \tag{12}$$

we obtain MI-relaxation problem (Problem 2 in Arthanari and Usha 2000).

Let the slack variables corresponding to MI-relaxation problem be denoted by u_{ij}, corresponding to the inequality for edge (i, j). It is shown by Arthanari and Usha (2000) that for any MI problem instance of size n, if for some $i, j \in V_n$ the slack variable $u_{ij} = 1$, then the edge (i, j) is present in the tour given by the optimal solution of the problem. The polytope corresponding to MI-relaxation problem is called the $P_{MI}(n)$. The affine transformation of $P_{MI}(n)$, projecting out x_{ijk} variables, is denoted by $\mathscr{U}(n)$. Arthanari and Usha (2000) compared $\mathscr{U}(n)$ with the SEP(n), and proved that $\mathscr{U}(n) \subseteq \text{SEP}_n$. Thus MI-formulation is as tight as the standard DFJ formulation and has only polynomially many constraints. Different formulations of TSP varying in size and in the strength of the LP relaxations have been compared in the literature (Langevin *et al.* 1990; Padberg and Sung 1991; Gouveia and Voß 1995; Orman and Williams 2007; Öncan *et al.* 2009; Roberti and Toth 2012; Godinho *et al.* 2014). In the article by Roberti and Toth (2012) we find a diagram that brings out the order relations among the linear relaxations of several polynomial formulations and of DFJ formulation, with respect to strength. Formulations by Dantzig *et al.* (1954), Wong (1980), Claus (1984), and Langevin *et al.* (1990) are equivalent, as they all achieve the same LP optimal value. However, they may not be similar in terms of computational burden as notice in the literature (Gubb 2003; Öncan *et al.* 2009; Haerian Ardekani 2011).

2.6. Properties of pedigree polytope. Earlier we introduced Hamiltonian cycles in a complete graph of n vertices. Now here we introduce the objects called pedigrees. We shall show the $1-1$ correspondence between Hamiltonian cycles and pedigrees.

First we define a generator of an edge.

Definition 2.10. Edge Generators: Given $e = (i, j) \in E_n$, $G(e)$ is called the set of generators of e, and it is defined as follows:

$$G(e) = \begin{cases} \delta(i) \cap E_{j-1}, & \text{if } j \geq 4, \\ E_3 \setminus \{e\}, & \text{otherwise.} \end{cases}$$

Since an edge $e = (i, j), j > 3$ is generated by inserting j in any e' in the set $G(e)$, the name *generator* is used to denote any such edge.

Example 2.1. Consider $n = 5, e = (1,5)$. Here $j \geq 4$, so, we have $G(e) = \delta(i) \cap E_{j-1}$. Since $i = 1, \delta(1) = \{(1,2),(1,3,)(1,4),(1,5)\}$, and $E_{j-1} = E_4 = \{(1,2),(1,3),(2,3),(1,4),(2,4),(3,4)\}$. Therefore, $G(e) = \{(1,2),(1,3,)(1,4)\}$.

Consider $e = (2,3)$. Since $j \leq 3$, we have $G(e) = E_3 \setminus \{e\} = \{(1,2),(1,3)\}$.

The following is one of the possible ways to define our main object of interest, namely, a pedigree. The other equivalent definitions are given in later sections.

Definition 2.11. Given n, consider $W = (e_4, \ldots, e_n)$, where $e_k = (i_k, j_k)$ for $1 \leq i_k < j_k \leq k-1, 4 \leq k \leq n$. W is called a pedigree if and only if

(1) $e_k, 4 \leq k \leq n$, are all distinct,
(2) $e_k \in E_{k-1}, 4 \leq k \leq n$, and
(3) for every $k, 5 \leq k \leq n$, there exists a $e' \in G(e_k)$ such that, $e_q = e'$, where $q = max\{4, j_k\}$.

Let $\mathscr{P}_n$ denote the set of all pedigrees for a given $n > 3$. For any $4 \leq k \leq n$, given an edge $e \in E_{k-1}$, with edge label l, we can associate a $0-1$ vector, $\mathbf{x}(e) \in B^{p_{k-1}}$, such that, $\mathbf{x}(e)$ has a 1 in the lth coordinate, and zeros else where. That is, $\mathbf{x}(e)$ is the indicator of e.

Let $\mathbf{E} = E_3 \times E_4 \ldots \times E_{n-1}$ be the ground set. Let B^{τ_n} denote the set of all binary vectors with τ_n coordinates. That is , here $\{0,1\}^{|E|} = B^{\tau_n}$. Then, we can associate an $X = (\mathbf{x}_4, \ldots, \mathbf{x}_n) \in B^{\tau_n}$, the characteristic vector of the pedigree W, where $(W)_k = e_k$, the $(k-3)$rd component of W, $4 \leq k \leq n$ and $\mathbf{x}_k$ is the indicator of e_k.

Let $P_n = \{X \in B^{\tau_n} \ : X$ is the characteristic vector of W, a pedigree$\}$. Consider the convex hull of P_n. We call this the *pedigree polytope*, denoted by $conv(P_n)$.

Given a cost vector $C \in R^{\tau_n}$, we wish to find a pedigree X^* in P_n that minimizes CX^*. We have a combinatorial optimization problem, called the pedigree optimization problem (POP). As indicated in the introduction, in section **2.5** we shall see that the symmetric traveling salesman problem can be posed as a POP.

Definition 2.12. Let $y(e)$ be the indicator of $e \in E_k$. Given a pedigree, $W = (e_4, \ldots, e_k)$ (with the characteristic vector, $X \in P_k$) and an edge $e \in E_k$, we call $(W, e) = (e_4, \ldots, e_k, e)$ an *extension* of W in case $(X, y(e)) \in P_{k+1}$.

A pedigree $W = (e_4, \ldots, e_n) \in \mathscr{P}_n$ is such that $(e_4, \ldots, e_k)$ is a pedigree in $\mathscr{P}_k$, for $4 \leq k \leq n$. We state this interesting property as a fact below:

Given $n > 3, X = (\mathbf{x}_4, \ldots, \mathbf{x}_n) \in P_n$, let X restricted to the first $k-3$ stage(s), be written as

$$X/k = (\mathbf{x}_4, \ldots, \mathbf{x}_k).$$

Fact: Given $X \in P_n$ and any k such that $4 \leq k \leq n$, X/k is in P_k.
Similarly, $X/(k-1)$ and $X/(k+1)$ are interpreted as restrictions of X.

Example 2.2. Consider W given by

$$W = (e_4 = (1,3), e_5 = (2,3), e_6 = (3,4), e_7 = (2,5)).$$

W is a pedigree because 1] all the edges are distinct 2] $e_k \in E_{k-1}, k = 4,\ldots,7$ and 3] $e_4 = (1,3)$ is a generator of e_5 and $e_6 = (3,4)$. Also $e_5 = (2,3)$ is a generator of $e_7 = (2,5)$.

Let $e = (3,5) \in E_7$. Here $q = 5$. Then (W,e) is a pedigree as 1] e does not appear in W, 2] $G(e) = \{(1,3),(2,3,)(3,4)\}$ and $e_5 = (2,3)$ is in $G(e)$. However, if e were $(2,6)$, we have $q = 6$, and $G(e) = \{(1,2),(2,3),(2,4),(2,5)\}$. But $e_6 = (3,4)$ is not in $G(e)$. So (W,e) is not an extension of W.

Pedigree polytope is the integer hull of MI-relaxation polytope, like the tour polytope is the integer hull of the subtour elimination polytope. Unlike the tour polytope the pedigree polytope has a nicer adjacency structure. Research on pedigree polytopes and their connection to tour polytopes are of interest because in the graph of pedigree polytope, given two pedigrees, whether they are adjacent or not can be checked by a strongly polynomial algorithm as shown by Arthanari (2006). Also pedigree polytope is a combinatorial polytope, in the sense of Naddef and Pulleyblank (Naddef and Pulleyblank 1981). Pedigree polytope and its properties are studied by Arthanari (2005, 2006, 2008, 2013a,b). Arthanari (2013b) establishes that a sufficiency condition for nonadjacency in tour polytope is nonadjacency of the corresponding pedigrees in the pedigree polytope. Also, pedigree polytope as an extension of tour polytope is an extension without hidden vertices (Pashkovich and Weltge 2015). Recently, Makkeh *et al.* (2016), based on the characterization of adjacency in Pedigree graph given by Arthanari, study an *adjacency game*. The authors using this game show that the minimum degree for a vertex in a pedigree graph is asymptotically equal to the number of vertices - *i.e.*, asymptotically the graph is almost complete. These properties of pedigree polytope, along with the computational results on MI-relaxation, favour using pedigree approach to solving STSP. And hence the results obtained in the subsequent sections are important in pushing the boundary of solving MI-relaxation instances.

3. MI-relaxation and Leontief substitution flow problem

A special case of the minimum cost hypergraph flow problem discussed in section **2.1** is the minimum cost gainfree Leontief substitution flow problem. In this section we show that MI-formulation contains a subproblem that is a gainfree Leontief substitution flow problem.

Consider a hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{E})$. We can write the vertex-hyperarc incidence matrix $M = [m_{v,e}]$ corresponding to $\mathscr{H}$ as follows:

$$m_{v,e=(T_e,h_e)} = \begin{cases} 1 & \text{if } v = h_e, \\ -\mu_v(e) & \text{if } v \in T_e, \\ 0 & \text{otherwise.} \end{cases}$$

Matrix $E_{[n]}$ for $n = 6$

$$E_{[6]} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

We can write the minimum cost hypergraph flow problem **2.1** equivalently in matrix form as:

$$\min \sum_{e \in \mathscr{E}} c(e) f(e) \tag{13}$$

$$Mf = b, \tag{14}$$

$$f \geq 0, \text{ and} \tag{15}$$

$$f \leq w. \tag{16}$$

Notice that in M every column corresponding to a hyperarc has at most one positive element, namely, $+1$. Such matrices are studied in economics, inventory management and other fields.

We consider now MI-formulation in matrix form after introducing required matrix notations.

Definition 3.1. In general, let $E_{[n]}$ denote the matrix corresponding to Eq. (8); let $A_{[n]}$ denote the matrix corresponding to the inequalities (9, 10 & 12). Let $\mathbf{1}_r$ denote the row vector of r $1's$. Let I_r denote the identity matrix of size $r \times r$. Then we can write recursively,

$$E_{[n]} = \begin{pmatrix} \mathbf{1}_{p_3} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \mathbf{1}_{p_{n-2}} & 0 \\ 0 & 0 & \dots & \mathbf{1}_{p_{n-1}} \end{pmatrix} = \begin{pmatrix} E_{[n-1]} & 0 \\ \mathbf{0} & \mathbf{1}_{p_{n-1}} \end{pmatrix}.$$

To derive a similar expression for $A_{[n]}$ we first define

$$A^{(n)} = \begin{pmatrix} I_{p_{n-1}} \\ -M_{n-1} \end{pmatrix}$$

where M_i is the $i \times p_i$ *node-edge incidence* matrix. Then

$$A_{[n]} = \begin{pmatrix} A^{(4)} & | & A^{(5)} & | & & | & A^{(n)} \\ & | & & | & \ddots & | & \\ \mathbf{0} & | & \mathbf{0} & | & & | & \end{pmatrix} = \begin{pmatrix} A_{[n-1]} & | & A^{(n)} \\ \mathbf{0} & | & \end{pmatrix}.$$

Observe that $A^{(n)}$ is the submatrix of $A_{[n]}$ corresponding to $\mathbf{x}_n$. The number of rows of $0's$ is decreasing from left to right.

MI-formulation can be recast in matrix notation as:

$$\text{Find } X^* \ni CX^* = \min\{CX | X \ni E_{[n]}X = \mathbf{1}_{n-3}, A_{[n]}X \leq \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix}, X \in \{0,1\}^{\tau_n}\}.$$

Matrix $A_{[n]}$ for $n = 6$

$$A_{[6]} = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1
\end{pmatrix}.$$

Observe that the elements of the constraint matrix of MI-formulation are $0, \pm 1$; the right hand side is a nonnegative integer vector and there are two $+1$'s in each column corresponding to any x_{ijk}, once in $E_{[n]}$ and another time in $A_{[n]}$. And so, unfortunately this problem does not correspond to a gainfree Leontief substitution flow problem.

Definition 3.2. We say X is a *pre-solution* to MI-relaxation if X satisfies the inequalities 9, 10 and 12. In addtion if X is non negative we say X is a *feasible pre-solution*. Equivalently, X is a pre-solution to MI-relaxation problem means

$$A_{[n]}X + I_{P_n}u = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix}.$$

So a feasible pre-solution X, if in addition satisfies Eq. 8, we have a feasible solution to MI-relaxation.

Define

$$\mathscr{X}(n) = \{X \mid E_{[n]}X = \mathbf{1}_{n-3}\} \tag{17}$$

$$\mathscr{U}(n) = \{u \in R^{p_n}, u \geq 0 \mid \exists\, X \in \mathscr{X}(n) \ \ni\ A_{[n]}X + I_{p_n}u = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix},\ X \geq 0\} \tag{18}$$

Remark **3.1**. If X is a pre-solution to MI-relaxation, then pre-multiplying both sides of $A_{[n]}X + I_{p_n}u = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix}$ by c and noting that $cA_{[n]}$ is indeed $-C$, we get $cu = c\begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix} + CX$. So, it is seen that the objective function of MI-formulation is equivalent to cu but for the constant $c_{12} + c_{13} + c_{23}$. So we can equivalently write MI -relaxation problem as Problem **3.1**.

Problem **3.1**.

$$\text{minimise } cu \tag{19}$$

$$\text{subject to } A_{[n]}X + I_{pn}u = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix} \tag{20}$$

$$X \geq 0 \tag{21}$$

$$u \geq 0 \tag{22}$$

$$X \in \mathscr{X}(n). \tag{23}$$

Now notice that the system given by Eqs. 20, 21 and 22 corresponds to a gainfree Leontief substitution system, as (a) the system matrix, $A_{[n]}, I_{p_n}$ is a pre-Leontief matrix, (b) all the rows of this system are nontrivial, (c) elements of the system matrix are ± 1 or 0, (hence gainfree) and (d) the right hand side is a nonnegative integer vector. Therefore a feasible pre-solution X to MI-relaxation with the corresponding u provides us a solution (X,u) to this system. And the basic feasible solutions of this system are integral, by an application of Veinott's Theorem **2.1** and using the directed hypertree corresponding to the basis. Thus we have Theorem **3.1**.

Theorem 3.1. *Consider Problem **3.1** excluding the restriction 23*

$$\textit{minimize } cu, \textit{subject to } A_{[n]}X + I_{p_n}u = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix}, X, u \geq 0.$$

We have a gainfree Leontief substitution flow problem, and every feasible basis is integral.

We shall use this fact subsequently to devise algorithms based on Lagrangian relaxation of the equality constraints 23. In section **2.3** we reviewed the literature on Lagrangian relaxation. We apply some of the approaches and tips to solve Problem **3.1**.

We next propose different algorithms based on Lagrangian relaxation as applied to MI-relaxation problem.

3.1. Lagrangian relaxation of Multiple Choice Constraints. Consider Problem **3.1**. We could rewrite it using Lagrangian multipliers corresponding to the multiple choice constraints given by Eq. (23) as:

Problem **3.2**. [Lagrangian Relaxation problem: LR_λ]

$$\text{minimise } cu + \sum_{k=4}^{n} \lambda_k \Big(\sum_{1 \leq i < j < k} x_{ijk} - 1 \Big) \tag{24}$$

$$\text{subject to } A_{[n]}X + I_{p_n}u = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix} \tag{25}$$

$$X \geq 0 \tag{26}$$

$$u \geq 0 \tag{27}$$

$$\lambda \text{ real.} \tag{28}$$

As observed earlier in Theorem **3.1**, Problem-LR_λ is a gainfree Leontief flow problem and so we can apply the strongly polynomial value iteration algorithm from (Jeroslow *et al.* 1992). Thus Lagrangian subproblem for a given λ is not only easy to solve but also provides

an integer optimal solution. Now to solve LD, we could resort to subgradient optimization or other approaches.

3.2. Value iteration algorithm for Problem 3.2. Consider the dual of the Problem LR_λ for a given λ. We introduce variable π_{ij} for each row of Eq. (25), $\forall\ 1 \le i < j \le n$. For each hyperarc $(\{(i,k),(j,k)\},(i,j))$ corresponding to the variable x_{ijk} we have a dual constraint; for each hyperarc $(\emptyset,(i,j))$ corresponding to the slack variable u_{ij} we have an upper bound constraint for the dual variable, π_{ij} as follows:

$$\pi_{ij} \le \lambda_k + \pi_{ik} + \pi_{jk},\ \forall\ 1 \le i < j < k;\ k = 4,\ldots,n, \tag{29}$$

$$\pi_{ij} \le c_{ij},\ \forall 1 \le i < j \le n. \tag{30}$$

In the paper by Jeroslow *et al.* (1992) we have a strongly polynomial successive approximation scheme. This is based on the Bellman-Ford method (see, for instance, Chapter 7 of Korte and Vygen 2012), which finds either a negative cost cycle or $s-t$ shortest paths for a given $s \in G$, a directed graph with edge costs. We specialize the value iteration algorithm for this specific gainfree Leontief flow problem at hand and get Algorithm 1. Let M be a sufficiently large positive value, which is used to initialize the solution.

Algorithm 1 Algorithm: Value Iteration for LR_λ

INPUT: c_{ij}, cost coefficient for $u_{ij}\ \forall\ 1 \le i < j \le n, \lambda_k$, for $k = 4,\ldots,n$.

OUTPUT: $k^*_{(i,j)}, t[ij]\ \forall\ 1 \le i < j \le n$. Dual solution to LR_λ, that is, $\pi_{ij}^{t[ij]}\ \forall\ 1 \le i < j < k \le n$.

Step 1 $\pi^0_{ij} \leftarrow M; t[ij] \leftarrow 0; \forall (i,j) \in E_n; t \leftarrow 0;$
Step 2 Iterative Step:
while some π^t_{ij} changed; **do**
  $t \leftarrow t+1;$
  for $(i,j) \in E_n$ **do**
    $\bar{k} = argmin\{\lambda_k + \pi^{t-1}_{ik} + \pi^{t-1}_{jk} | 4 \le k \le n\};$
    $\pi^t_{ij} = \min\{\pi^{t-1}_{ij}, \lambda_{\bar{k}} + \pi^{t-1}_{i\bar{k}} + \pi^{t-1}_{j\bar{k}}, c_{ij}\};$
    if $\pi^{t-1}_{ij} > \pi^t_{ij}$ **then**
      $k^*_{(i,j)} \leftarrow \bar{k};$
      $t[ij] \leftarrow t$
    end if
  end for
end while
return $k^*_{(i,j)}, t[ij], \pi_{ij}^{t[ij]}\ \forall\ 1 \le i < j \le n.$

Lemma 3.2 of (Jeroslow *et al.* 1992) asserts that, given a Leontief flow problem for a Leontief directed hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{E})$, if there is a basic feasible flow $\bar{f}$ having acyclic support with a hyperarc directed into every nontrivial vertex, and a complementary dual solution feasible for the dual, then for all nontrivial vertex in $\mathscr{V}$, their value iteration algorithm finds this dual solution after at most $|\mathscr{V}|$ iterations. So we have a strongly

polynomial method for solving the problem LR_λ. We can recover the primal solution for the problem LR_λ using the specialized version (Algorithm 2) of the primal retrieval steps given by Jeroslow *et al.* (1992). Let us denote x_{ijk} by $x[k:(i,j)]$ for the hypergraph flow variable corresponding to the hyperarc $(\{(i,k),(j,k)\},(i,j))$. This way the head of the hyperarc is clearly seen as (i,j) and the tail set $\{(i,k),(j,k)\}$ corresponds to the insertion of k in (i,j). We wish to retrieve these values for a given dual solution obtained from the value iteration Algorithm 1.

Algorithm 2 Algorithm: Primal Retrieval for LR_λ

INPUT: b_{ij}, demand at node (i,j); $k^*_{(i,j)}, t[ij]$ from value iteration algorithm, $\forall\ 1 \le i < j \le n$.
OUTPUT: Primal solution to LR_λ, that is, $x[k:(i,j)]\ \forall\ 1 \le i < j < k \le n$.

Step 1: Set $x[k:(i,j)] \leftarrow 0\ \forall\ (k:(i,j) \in \mathscr{E}$; create active vertex list $\tilde{V} \leftarrow E_n$; set vertex flows $f_{ij} \leftarrow b_{ij} \ge 0\ \forall\ (i,j) \in \tilde{V}$.
Step 2 Iterative Step:
while $\tilde{V} \neq \emptyset$, **do**
  Let $t[(\bar{i},\bar{j})] = \max\{t[(i,j)] \mid (i,j) \in \tilde{V}\}$.
  Choose the hyperarc with head $(\bar{i},\bar{j})$ and tail set $\{(\bar{i},k^*_{\bar{i}\bar{j}}),(\bar{j},k^*_{\bar{i}\bar{j}})\}$ and
  update $x[k^*_{\bar{i}\bar{j}}:(\bar{i},\bar{j})] \leftarrow f_{\bar{i}\bar{j}}$;
  $f_{\bar{i},k^*_{\bar{i}\bar{j}}} \leftarrow f_{\bar{i},k^*_{\bar{i}\bar{j}}} + x[k^*_{\bar{i}\bar{j}}:(\bar{i},\bar{j})]$;
  $f_{\bar{j},k^*_{\bar{i}\bar{j}}} \leftarrow f_{\bar{j},k^*_{\bar{i}\bar{j}}} + x[k^*_{\bar{i}\bar{j}}:(\bar{i},\bar{j})]$;
  $\tilde{V} \leftarrow \tilde{V} \setminus \{(\bar{i},\bar{j})\}$.
end while
return $x[k:(i,j)]\ \forall\ 1 \le i < j < k \le n$.

Thus we have a way to recover the primal solution from a solution to Problem **3.2** without further increasing the complexity of solving the Problem **3.2**. The time complexity of solving this problem using value iteration algorithm and primal retrieval is $O(n^2)$ as the number of rows in the gainfree Leontief flow problem (Problem **3.2**) is $p_n + (n-3)$.

3.2.1. *Lagrangian Dual of Problem 3.1: LD.* As discussed in subsection **2.3** to find the best possible Lagrangian bound we solve the Langrangian Dual problem. Here we consider LD given by

Problem **3.3**. [Lagrangian Dual: LD] Find λ^* real $\ni z(\text{LD}) = z(LR_{\lambda^*}) = \max_\lambda\{z(LR_\lambda)\}$.

As mentioned earlier subgradient optimization is applied in practice to solve this problem, though we do not have any complexity bounds on using subgradient optimization. Theoretically, we can guarantee a polynomial time algorithm to find the optimal value of the problem **3.1** using Vaidya's sliding objective algorithm (Vaidya 1990), as discussed by Bertsimas and Orlin (1994). Bertsimas and Orlin propose techniques for the solution of the LP relaxation and LD in combinatorial optimization problems. One can use simplified and strengthened version as in (Anstreicher 1997) in place of Vaidya's algorithm. One of the structured LP problems they discuss is $\min\{\sum_{r=1}^N c_r x_r \mid \sum_{r=1}^N A_r x_r = b, x_r \in \mathscr{S}_r, r = 1,\ldots,N\}$. Notice

that Problem **2.3** can be written in this form using $A^{(k)}$ matrices defined earlier (Definition **3.1**) as:

$$\min\{\sum_{1\le i<j\le n} c_{ij}w_{ij} \mid \sum_{k=4}^{n} A^{(k)}x_k + I_{p_n}w = \begin{bmatrix} \mathbf{1}_3 \\ 0 \end{bmatrix}, \mathbf{1}_{p_n} \ge w \ge 0, \\ x_k \in \mathscr{S}_k, r = 4,\ldots,n\},$$

with

$$\mathscr{S}_k = \{x_k = (x_{12k},\ldots,x_{k-2k-1k}) \ge 0 \mid \sum_{1\le i<j<k} x_{ijk} = 1\}.$$

Let $\{x_k^l, l \in J_k\}$ be the set of extreme pints of $\mathscr{S}_k$. Let $y, \mu \in R^{p_n}$, and $\sigma_k \in R, k = 4,\ldots,n$. We can write the dual of the above problem as:

Problem **3.4**.

$$z(D) = \max yb + \sum_{1\le i<j\le n} \mu_{ij} + \sum_{k=4}^{n} \sigma_k \tag{31}$$

$$\text{subject to } yA^{(k)}x_k^l + \sigma_k \le 0, \ \ \forall\, l \in J_k \text{ and } k \in \{4,\ldots,n\}, \tag{32}$$

$$y + \mu \le c. \tag{33}$$

As suggested by Bertsimas and Orlin's approach we solve the problem **3.4** using Vaidya's algorithm; which in turn solves a separation problem for all k, given a solution y, μ, σ to this problem. Each separation problem turns out to be a problem of finding minimum over a multiple choice constrained set having p_k variables. So it can be done in time $O(p_k)$. So we could apply Theorem 2 of (Bertsimas and Orlin 1994) to obtain a bound on the complexity of solving this problem given by

Theorem 3.2. *The optimal solution value of problem **3.1** and the optimal dual variables can be found in $O([\sum_{k=4}^{n} p_{k-1}](p_n+(n-3))L + M[p_n+(n-3)](p_n+(n-3))L)$ arithmetic operations, where L is the input size of the instance, $M[s]$ is the number of arithmetic operations to multiply two matrices of size $s \times s$.*

Following Theorem 1 of (Bertsimas and Orlin 1994), the authors discuss finding a solution to the problem **3.1** without changing the overall complexity of the method. However, we often want to find the optimal solution value rather than the solution of the LP or Lagrangian relaxation, since the solution value can be later used in a branch and bound algorithm. Bertsimas and Orlin (1994) apply their approach to solve LD discussed in general. So we could apply the same to problem **3.3**. The separation problem in this case turns out to be a Leontief substitution flow problem and so we can solve it in time $O(p_n)$, as observed earlier. Thus, using Theorem 4 in (Bertsimas and Orlin 1994) we have a polynomial algorithm of complexity $O(p_n(p_n+n-3)L + M(p_n+n-3)(p_n+n-3)L)$ to solve LD. Notice that this algorithm, complexity wise is superior to the one mentioned above in Theorem **3.2**.

However, in practice subgradient optimization and its variants are observed to perform well (Lorena and Narciso 2002; Zamani and Lau 2010) for solving LD. There are other competing methods like Volume method and Bundle method, which could be compared with subgradient approach to solve LD.

4. Hypergraph flow and MI-relaxation

A simple trick converts MI-relaxation problem into a minimum cost hypergraph flow problem. We noticed earlier that the system of equations is not a pre-Leontief system and since there are two $+1$'s in each column; it also does not correspond to a hypergraph flow problem. But multiplying both sides of the multiple choice Eqs. (23) by -1, with the rest of the constraints intact, we get a minimum cost hypergraph flow problem corresponding to MI-relaxation. But this destroys the Leontief substitution flow structure as the righthand side elements are not nonnegative anymore. Here we study the special minimum cost hypergraph flow problem at hand bringing out the changes required in the procedures used in the hypergraph simplex algorithm of Cambini *et al.* (1992).

We consider the following hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{E})$ corresponding to MI-formulation. We have, $\mathscr{V} = \{4, \ldots, n\} \cup \{(i,j) \mid 1 \le i < j \le n\}$ with $\mathscr{E} = \{(\emptyset, (i,j)) \mid (i,j) \in \mathscr{V}\} \cup_{k=4}^{n} \{(k : (i,j)) \mid 1 \le i < j < k\}$ where $(k : (i,j))$denotes the hyperarc$(\{(i,k),(j,k),k\},(i,j))$, for $1 \le i < j < k, k \in \mathscr{V}$.

Theorem 4.1. *The hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{E})$ corresponding to MI-formulation is cycle-free.*

Proof: Vertices in $S = \{4, \ldots, n\}$ have no hyperarcs entering any of $k \in S$. So any directed path starting from k cannot end in k. So there are no cycles involving $k \in S$. Consider any $(i,j) \in \mathscr{V}$, for any $1 \le i < j \le n$.

Case 1: $(i,j) \in \mathscr{V}, 1 \le i < j \le 3$. Since none of these vertices is in the tail set of any hyperarc, a directed cycle involving such an (i,j) is not possible. Case 2: $(i,j) \in \mathscr{V}, 1 \le i < j, 4 \le j \le n$. Suppose for some (i_0, j_0) there is a directed cycle, then (i_0, j_0) is the head for an hyperarc e and is in the tail set of another hyperarc, e'. Therefore, e' has to be an arc $(j_0 : (u,v))$ for some $u < v < j_0$ with u or $v = i_0$ and e is either $(\emptyset, (i_0, j_0))$ or $(r : (i_0, j_0))$ for some $n \ge r > j_0$.

First we show that $e = (\emptyset, (i_0, j_0))$ is not possible. In any directed path, if e appears as e_i for some $1 \le i \le q$ then the vertex v_i is required to belong to $\{h_{e_{i-1}}\} \cup T_{e_i}$. But $T_{e_i} = T_e = \emptyset$ implies e cannot appear in any such directed path.

So $e = (r : (i_0, j_0))$ for some $n \ge r > j_0$. Now we have in the directed path,

$$\ldots, (r : (i_0, j_0)), (i_0, j_0), (j_0 : (u,v)), \ldots$$

with $u < v < j_0$ and one of u or $v = i_0$.

Thus any directed path $P_{st} = (v_1 = s = (i,j), e_1, v_2, \ldots, e_q, v_{q+1} = t)$ in $\mathscr{H}$ is such that $h_{e_q} = t$ and $t = (a,b)$ with $max\{a,b\} < j$. Therefore $t = s$ is not possible, and hence $\mathscr{H}$ is cycle-free.□

Theorem 4.2. *Given any pedigree P, we have a spanning hypertree of $\mathscr{H} = (\mathscr{V}, \mathscr{E})$ given by $(\mathscr{V}, E_T)$ with*

$$E_T = \{(k : (i,j) \mid (i,j) \in E(P)\} \cup \{(\emptyset, (i,j) \mid (i,j) \in E_{n-1} \setminus E(P)\},$$

where, $E(P) = \{(i_k, j_k) \mid 4 \le k \le n, \ni P = ((i_4, j_4), \ldots, (i_n, j_n))$ is the given pedigree $\}$.

Proof: Since $E_T \subset \mathscr{E}$ and $(\mathscr{V}, E_T)$ is a subhypergraph of $\mathscr{H}$, $(\mathscr{V}, E_T)$ is cycle-free. Let $R = \{k \mid 4 \le k \le n\}$, then $N = \mathscr{V} \setminus R$. So $(\mathscr{V}, E_T)$ satisfies the requirement $R \cap N = \emptyset$. No hyperarc in E_T has a vertex in R as its head. Every vertex $v = (i,j) \in N$ either has a unique hyperarc $(k : (i,j))$ entering (i,j) if $(i,j) \in E(P)$ or has a unique hyperarc $(\emptyset, (i,j))$

entering (i,j) if $(i,j) \notin E(P)$. Thus $(\mathscr{V}, E_T)$ is indeed a hypertree. Notice that it is a spanning hypertree as well, as all vertices in $\mathscr{V}$ are spanned. Since we have $m = |\mathscr{V}|$ is the number of vertices and we have only $|E_T|$ hyperarcs, we need to add $n-3$ other hyperarcs such that the incidence matrix corresponding to the spanning hypertree is extended to a basis of size $m \times m$. The $n-3$ hyperarcs external to the hyperarcs in E_T are such that they have at least one non-zero element among the last E_n elements of the column. Let $\mathscr{T}_R$ be a traverse of $(\mathscr{V}, E_T)$ given by

$$(R,(\emptyset,(i_n,n)),(i_n,n),(\emptyset,(j_n,n)),(j_n,n),(n:i_n,j_n),\ldots,(4:i_4,j_4),(\emptyset,(i,j)),(i,j),$$
$$\forall (i,j) \notin E(P)).$$

This introduces an order among the vertices and hyperarcs and this order is used in rearranging the incidence matrix of the spanning hypertree.

Remark **4.1**.

(1) Notice that $\forall e \in \mathscr{E} \setminus E_T, T_e \cup \{h_e\}$ is not a subset of R as every hyperarc not in E_T has $h_e \in E_n$ and so not in R.
(2) In general hypergraph flow problems, we may not always have an initial primal feasible solution to the hypergraph flow problem. But for MI-hypergraph flow problem, we have a feasible basis given by the spanning hypertree corresponding to any pedigree, so we can start the hypergraph simplex algorithms without phase I using artificial variables.
(3) The set of linearly independent columns corresponding to the hyperarcs in E_T, needs to be expanded to a basis of size $m \times m$.

Example 4.1. Consider $n = 6$ and the pedigree in $\mathscr{P}_6$ given by $P = ((1,3),(1,4),(2,3))$. The corresponding spanning tree is given in Figure 1. The corresponding traverse $\mathscr{T}_R$ of the spanning hypertree with root $R = \{4,5,6\}$ and $E_T = \{(4:(1,3)),(5:(1,4)),(6:(2,3))\} \cup \{(\emptyset,(i,j)) \mid (i,j) \in E_5 \text{ and } (i,j) \notin \text{ the pedigree } P\}$ is:

$$\mathscr{T}_R = (R,(\emptyset,(2,6)),(2,6),(\emptyset,(3,6)),(3,6),(6:(2,3)),(2,3),(\emptyset,(1,5)),(1,5),(\emptyset,(4,5)),$$
$$(4,5),(5:(1,4)),(1,4),(\emptyset,(3,4)),(3,4),(4:(1,3)),(1,3),(\emptyset,(1,2)),(1,2),(\emptyset,(2,4)),$$
$$(2,4),(\emptyset,(2,5)),(2,5),(\emptyset,(3,5)),(3,5),(\emptyset,(1,6)),(1,6),(\emptyset,(4,6)),(4,6),(\emptyset,(5,6)),(5,6)).$$

We expand this to a basis by adding 3 $(=6-3)$ more external hyperarcs $(4:(1,2),(5:(3,4)$ and $(6:(3,5))$ that are linearly independent of the set of columns corresponding to the 15 hyperarcs in E_T. This initial basis is shown in Table 1. Notice that it is an upper triangular matrix.

We briefly reproduce the main steps of the Hypergraph Simplex from (Cambini *et al.* 1992; Scutellà *et al.* 1996). Since we have an initial basis we proceed to discuss the hypergraph simplex method for solving the minimum cost hypergraph flow problem. Each basis M corresponds to a pair $(\mathscr{T}_R, E_X)$ where $\mathscr{T}_R$ is a spanning hypertree of the sub-hypergraph corresponding to M. E_X is the set of the external hyperarcs, that is, the basic hyperarcs other than the hyperarcs in the spanning hypertree. The number of external arcs is same as the number of root vertices of $\mathscr{T}_R$. The root matrix M_R is given by rows corresponding to the root vertices and columns corresponding to the external hyperarcs. M

TABLE 1. Initial basis corresponding to the spanning hypertree for Example **4.1** (zeros are suppressed).

hyperarc →			06			05		04								04	05	06
vertex	26	36	23	15	45	14	34	13	12	24	25	35	16	46	56	12	34	35
4								−1								−1		
5						−1											−1	
6			−1															−1
26	1		−1															
36		1	−1															−1
23			1															
15				1		−1												
45					1	−1											−1	
14						1		−1								−1		
34							1	−1									1	
13								1										
12									1							1		
24										1						−1		
25											1							
35												1					−1	1
16													1					
46														1				
56															1			−1

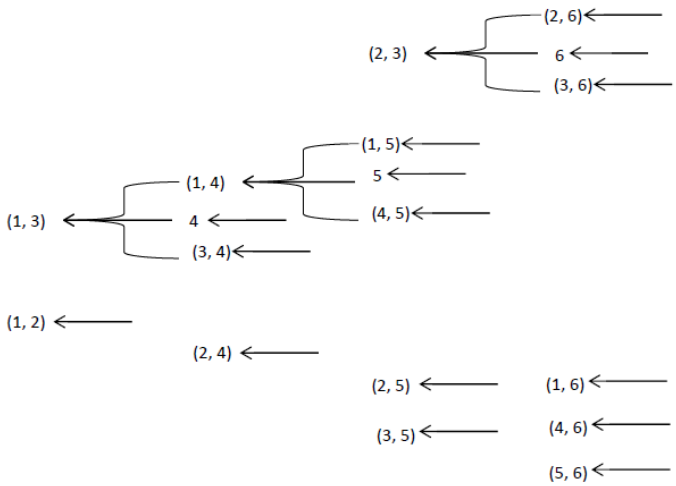

FIGURE 1. Spanning tree for Example **4.1**

corresponds to a basis if and only if the matrix M_R is non-singular. M_R can be completely characterized in terms of hypergraph concepts like flows and potentials. Let $\bar{b} = (\bar{b}(R), \bar{b}(N))$ be the demand vector induced on the vertices by the flows on the non-basic hyperarcs. Then the system $Mf = \bar{b}$ can be solved by the procedure PRIMAL, and $\pi M = \bar{c}$ can be solved by the DUAL procedure given by Cambini *et al.* (1992). The optimality of the current basis is checked using reduced costs thus computed. The optimality conditions are, as usual, based on the reduced costs: non- basic hyperarcs must have reduced costs ≥ 0 if their flow

is zero, and reduced costs ≤ 0 if their flow is at the upper bound. If these conditions are satisfied, M is optimal and the algorithm terminates. The steps of the simplex method are similarly inherited in hypergraph simplex, like choosing the hyperarc to enter and choosing the hyperarc to leave the basis; but these are done on the hypergraph, using hypergraph concepts.

Since the reported performance of the hypergraph simplex compared to that of generic LP solvers is superior, it is worth specializing the procedures FLOW[1], POTENTIAL, and PRIMAL, DUAL of hypergraph simplex algorithm to the particular special hypergraph corresponding to MI-MIrelaxation. Once these are implemented in a computer, the new algorithms based on the value iteration algorithm for Leontief substitution flow problem and the special minimum cost hypergraph flow algorithm could be compared with commercially available/opensource LP solvers.

A four phase approach to experimentally compare the efficacy of the suggested new algorithms is planned. Phase 1 is generating the computer codes for the specialized subroutines and algorithms and testing them. Phase 2 is the implementation of a prototype from the algorithms in phase 1. Phase 3 consists of optimizing the prototype from phase 2. Phase 4 consists of computational experiments. First phase in this plan is recently completed and appears in (Arthanari and Qian 2018).

5. Planned experiments to compare the proposed algorithms

Though some theoretical computational complexity results are given on the algorithms proposed in the preceding sections, the importance of conducting comparative computational experiments on TSBLIB test problems can not be undermined. For this reason experiments are planned to test Lagrangian based algorithms and minimum cost hypergraph flow simplex algorithm on two sets of problems: [1] STSP problems in (Reinelt 1991) and [2] randomly generated Euclidean STSP problem instances with varying sizes. The integrality gap, CPU time, number of iterations and other performance statistics on procedures like the value iteration, flows and potentials used in the algorithms would be collected. The main aim of the experiments is to estimate the compression in computational time achieved by the new algorithms compared to solving these LP instances using commercial (as well as other open source) generic LP solvers. Based on the reported advantages of using Lagrangian approach in STSP and other difficult combinatorial problems, and the superior performance of Hypergraph Simplex algorithm of Cambini *et al.* (1992), we are encouraged to conduct this computational comparisons. Also the special structure of MI-relaxation is expected to show additional computational advantage with further specialized versions of value iteration and hypergraph flow algorithms, fully exploiting MI-relaxations recursive structure.

6. Conclusions and future research directions

MI-formulation given by Arthanari (1982) is a compact alternative formulation for STSP, which has fewer constraints than the standard formulation by Dantzig *et al.* (1954). Computational experiments comparing MI-relaxation with other formulations have shown the

[1]The specialized version of the procedure FLOW (Cambini *et al.* 1992) is included in the Appendices (see Appendix 2).

superiority of MI-formulation with respect to integrality gap and other measures. However, in those, the size of the instances considered are less than 320 (Gubb 2003; Haerian Ardekani 2011), and the relaxed LP problems are solved using commercially available general purpose LP solvers. So in this paper the possibility of devising special purpose algorithms that completely exploit the structure of MI-relaxation problem is addressed. New algorithms based on the Leontief substitution flow problem and the minimum cost hypergraph flow problem are given. Computational complexity of some of the algorithms suggested are discussed. The plan for a computational experiment to compare the new algorithms with commercially available LP solvers is outlined. From the positive experience reported on hypergraph simplex method of Cambini *et al.* (1992) and Scutellà *et al.* (1996), and Lagrangian approach in general for combinatorial optimization and the strongly polynomial value iteration method for the Leontief substitution flow problem given by Jeroslow *et al.* (1992) we expect as a result of the algorithms proposed, in future we would be able to solve larger size instances of MI-relaxation. Also this paper advances the theoretical possibilities of understanding MI-formulation from the hypergraph flow perspective.

Acknowledgements

The author thanks anonymous referees for their valuable suggestions for improved clarity of the presentation. The author thankfully acknowledges Dr. David Carfi and Prof. Maria Grazia Scutellà for access to a copy of a paper on the experiments with Hypergraph Simplex method.

Appendices

Appendix 1: Excerpts from Haerian Ardekani (2011)

In Table 6.1 of Haerian Ardekani (2011) we have the computational comparisons of DFJ relaxation and MI-relaxation on STSP instances from (Reinelt 1991). Here, column 1 gives problem name. The value of the 2-matching relaxation and its gap are shown in columns LP1 Value, and LP1 Gap respectively. The number of violated subtour elimination constraints found using two different algorithms are shown in column SEC1 and SEC2, respectively. The solutions given by MI-formulation are given in the table as well as the CPU seconds for the MI and the DFJ formulations. The results for the MI-relaxation formulation are given separately in Table 6.2 of Haerian Ardekani (2011) in more detail. Table 6.2 includes the optimal solution of the instances and the optimal solution of the MI-relaxation. The solutions to the LP relaxation of the MI-formulation are equal to the results of the LP relaxation of the DFJ formulation. Problems larger than 318 for MI-relaxation could not be solved as it exceeded Cplex memory limit.

Ardekani solved diamond instances (Papadimitriou and Steiglitz 1977) from size 16 to 104 using LP relaxations of various TSP formulations . We did not solve instances larger than 11-diamonds by the formulation of Wong (1980), or instances larger than 12-diamonds by that of Claus (1984) or instances larger than 13-diamonds by the formulation of Carr (1996) as it would exceed Cplex memory. The results of this experiment are presented in Table 6.15 of Haerian Ardekani (2011).

Table 6.1: Results for the DFJ Formulation Compared with the MI-relaxation

	Optimal	MI	LP_1	MI	LP_1			CPU Seconds	
Problem	Value	Value	Value	gap	gap	SEC_1	SEC_2	DFJ	MI
gr17	2085	2085.0	1684.0	**0.00**%	19.23%	11	0	1.45	**0.02**
gr21	2707	2707.0	2707.0	0.00%	0.00%	0	0	0.14	**0.02**
gr24	1272	1272.0	1224.5	**0.00**%	3.73%	2	2	0.73	**0.03**
fri26	937	937.0	880.0	**0.00**%	6.08%	14	2	2.16	**0.02**
bayg29	1610	1608.0	1546.0	**0.12**%	3.98%	8	2	1.40	**0.03**
bays29	2020	2013.5	1944.0	**0.32**%	3.76%	3	1	0.63	**0.03**
dantzig42	699	697.0	641.0	**0.29**%	8.30%	5	2	1.24	**0.14**
swiss42	1273	1272.0	1214.5	**0.08**%	4.60%	3	8	2.03	**0.16**
att48	10628	10604.0	10041.5	**0.23**%	5.52%	27	7	6.10	**0.58**
gr48	5046	4959.0	4769.0	**1.72**%	5.49%	5	4	1.80	**0.69**
hk48	11461	11444.5	11197.0	**0.14**%	2.30%	14	5	3.30	**0.52**
eil51	426	422.5	416.5	**0.82**%	2.23%	4	0	**0.74**	0.88
berlin52	7542	7542.0	7163.0	**0.00**%	5.03%	6	0	1.05	**0.97**
brazil58	25395	25345.5	20896.0	**0.19**%	17.72%	9	12	4.73	**1.42**
st70	675	671.0	623.5	**0.59**%	7.63%	19	16	10.08	**3.39**
eil76	538	537.0	534.0	**0.19**%	0.74%	4	0	**0.87**	9.92
pr76	108159	105120.0	98994.5	**2.81**%	8.47%	5	5	3.16	**2.80**
rat99	1211	1206.0	1198.0	**0.41**%	1.07%	9	11	**12.13**	14.25
kroA100	21282	20936.5	19378.5	**1.62**%	8.94%	18	23	**22.79**	29.89
kroB100	22141	21834.0	20339.5	**1.39**%	8.14%	14	25	**23.09**	25.25
kroC100	20749	20472.5	19705.0	**1.33**%	5.03%	27	18	23.20	**21.88**
kroD100	21294	21141.5	19952.5	**0.72**%	6.30%	23	25	27.87	**27.78**
kroE100	22068	21799.5	20622.0	**1.22**%	6.55%	33	18	**24.50**	33.61
rd100	7904	7896.5	7334.0	**0.09**%	7.21%	28	27	**31.29**	41.14
eil101	629	627.5	619.0	**0.24**%	1.59%	11	13	**14.45**	65.25
lin105	14379	14370.5	13213.5	**0.06**%	8.11%	40	29	44.46	**29.69**
pr107	44303	44303.0	43381.0	**0.00**%	2.08%	168	35	173.08	**10.50**
pr124	59030	58067.5	50200.0	**1.63**%	14.96%	76	23	88.97	**47.34**
bier127	118282	117431.0	112278.5	**0.72**%	5.08%	40	28	**62.66**	146.80
ch130	6110	6075.5	5597.0	**0.56**%	8.40%	50	35	**85.77**	143.49
pr144	58537	58189.3	32863.0	**0.59**%	43.86%	59	69	234.70	**192.84**
ch150	6528	6490.1	6295.0	**0.58**%	3.57%	17	32	**87.24**	477.22
kroA150	26524	26299.0	24848.8	**0.85**%	6.32%	30	47	**127.02**	733.44
kroB150	26130	25732.5	24698.0	**1.52**%	5.48%	53	30	**116.62**	358.89
u159	42080	41925.0	40685.0	**0.37**%	3.32%	21	29	**85.87**	1183.81
si175	21407	21374.5	21140.0	**0.15**%	1.25%	84	40	**287.76**	799.61
brg180	1950	1950.0	1800.0	**0.00**%	7.69%	29	17	**113.69**	2671.59

Table 6.1 from Haerian Ardekani (2011).

Table 6.2: Results for the LP Relaxation of MI Formulation for Some TSPLIB Instances

Problem	Size	Optimal Value	MI relaxation	MI Gap	Cplex Iterations	Solution Seconds
gr17	17	2085	2085.0	0.00%	133	0.02
gr21	21	2707	2707.0	0.00%	298	0.02
gr24	24	1272	1272.0	0.00%	288	0.03
fri26	26	937	937.0	0.00%	224	0.02
bayg29	29	1610	1608.0	0.12%	375	0.03
bays29	29	2020	2013.5	0.32%	435	0.03
swiss42	42	1273	1272.0	0.08%	960	0.14
dantzig42	42	699	697.0	0.29%	935	0.16
att48	48	10628	10604.0	0.23%	2028	0.58
gr48	48	5046	4959.0	1.72%	2363	0.69
hk48	48	11461	11444.5	0.14%	1820	0.52
eil51	51	426	422.5	0.82%	2217	0.88
berlin52	52	7542	7542.0	0.00%	2794	0.97
brazil58	58	25395	25345.5	0.19%	3596	1.42
st70	70	675	671.0	0.59%	4838	3.39
eil76	76	538	537.0	0.19%	7983	9.92
pr76	77	108159	105120.0	2.81%	3380	2.80
rat99	99	1211	1206.0	0.41%	7994	14.25
kroA100	100	21282	20936.5	1.62%	11708	29.89
kroB100	100	22141	21834.0	1.39%	10732	25.25
kroC100	100	20749	20472.5	1.33%	9386	21.88
kroD100	100	21294	21141.5	0.72%	11058	27.78
kroE100	100	22068	21799.5	1.22%	12338	33.61
rd100	100	7904	7896.5	0.09%	12603	41.14
eil101	101	629	627.5	0.24%	17586	65.25
lin105	105	14379	14371.0	0.06%	7358	29.69
pr107	107	44303	44303.0	0.00%	5988	10.50
gr120	120	6942	6911.3	0.44%	27553	151.81
pr124	124	59030	58067.5	1.63%	9792	47.34
bier127	127	118282	117431.0	0.72%	22753	146.80
ch130	130	6110	6075.5	0.56%	24542	143.49
pr136	136	96772	95934.5	0.87%	31818	216.08
pr144	144	58537	58189.3	0.59%	19418	192.84
ch150	150	6528	6490.1	0.58%	43906	477.22
kroA150	150	26524	26299.0	0.85%	49837	733.44
kroB150	150	26130	25732.5	1.52%	35071	358.89
pr152	152	73682	73208.5	0.64%	24364	224.98
u159	159	42080	41925.0	0.37%	80621	1183.81
si175	175	21407	21374.5	0.15%	47553	799.61
brg180	180	1950	1950.0	0.00%	135584	2671.59
rat195	195	2323	2299.3	1.02%	52534	706.19
d198	198	15780	15722.0	0.37%	69002	958.75
kroA200	200	29368	29065.0	1.03%	93262	3154.19
kroB200	200	29437	28865.0	1.94%	99720	3043.31

Continued on next page

Table 6.2 from Haerian Ardekani (2011).

Problem	Size	Optimal Value	MI relaxation	MI Gap	Cplex Iterations	Solution Seconds
pr226	226	80369	80092.0	0.34%	141296	6776.33
pr264	264	49135	49020.5	0.23%	90859	2022.39
a280	280	2579	2566.0	0.50%	412138	34954.51
pr299	299	48191	47423.0	1.59%	439825	35243.33
lin318	318	42029	41888.8	0.33%	289438	23196.38
linhp318	318	41345	41245.8	0.24%	289438	22386.91

Continuation of Table 6.2 from Haerian Ardekani (2011).

Table 6.15: Cplex Iterations for Solving Directed Diamond with ATSP Formulations

k	Si□e	Carr	Claus	FGG	Flood	MI	MTZ
2	12	146	1230	10	9	7	10
3	18	1351	5636	16	105	38	16
4	24	3000	16510	24	206	88	22
5	30	5261	34980	30	384	243	28
6	36	12422	66997	37	587	464	35
7	42	24594	120050	52	400	339	42
8	48	43944	191265	58	794	751	46
9	54	58468	296440	69	974	1048	52
10	60	93089	422779	77	1565	1270	58
11	66	133179	633209	78	2256	1599	64
12	72	178982	949096	95	3276	2162	70

Table 6.15 from Haerian Ardekani (2011).

Appendix 2

Here we give the specialized version of the FLOW procedure from Cambini *et al.* (1992). This version uses the fact that the tail of a hyperarc in MI-relaxation problem is either $\emptyset$ or $\{(i,k),(j,k),k\}$, when the head is (i,j).

Algorithm 3 Procedure flow:

INPUT: Hypergraph $\mathscr{H} = (\mathscr{V}, \mathscr{E})$, a spanning tree given by $\mathscr{T}_R = (R \cup N, E_T)$; a traverse of the spanning tree $\mathscr{T}_R = (R, e_1, v_1, \ldots, e_q, v_q)$; demands at nonroot vertices $d(N)$, flows on external hyperarcs; external hyperarcs $E_X = \mathscr{E} \setminus E_T$.
OUTPUT: demands at root vertices $d(R)$, and flows on tree hyperarcs $f(T)$;

for $v \in R$ **do**
  $d(v) = 0$;
end for
for $e = (k : (i,j)) \in E_X$ **do**
  $d(i,k) \leftarrow d(i,k) + f(e)$;
  $d(j,k) \leftarrow d(j,k) + f(e)$;
  $d(k) \leftarrow d(k) + f(e)$;
  $d(i,j) \leftarrow d(i,j) - f(e)$;
end for
for $e = (\emptyset, (i,j)) \in E_X$ **do**
  $d(i,j) \leftarrow d(i,j) - f(e)$;
end for
for $v \in \mathscr{V}$ **do**
  $unvisited(v) \leftarrow$ number of hyperarcs incident into v;
end for
$Queue \leftarrow \{v \mid v$ is a leaf of $\mathscr{T}_R\}$;
while $Queue \neq \emptyset$ **do**
  Select a $v \in Queue$;
  $Queue \leftarrow Queue \setminus v$;
  let $e_v = (k' : (i', j'))$;
  $f(e_v) \leftarrow d(v)$ *if* $v = i'j', -d(v)$ otherwise;
  for $w \in \{i'j', i'k', j'k', k'\} \setminus \{v\}$ **do**
    $d(w) \leftarrow d(w) - f(e_v)$ if $w = i'j', d(w) + f(e_v),$ otherwise;
    $unvisited(w) \leftarrow unvisited(w) - 1$;
    if $unvisited(w) = 1$ **and** $w \notin R$ **then**
      $Queue \leftarrow Queue \cup \{w\}$;
    end if
  end for
end while
for $v \in R$ **do**
  $d(v) \leftarrow -d(v)$;
end for
return demands at root vertices: $d(R)$, and flows on tree hyperarcs: $f(T)$;

References

Anstreicher, K. M. (1997). "On Vaidya's volumetric cutting plane method for convex programming". *Mathematics of Operations Research* **22**(1), 63–89. URL: https://www.jstor.org/stable/3690140.

Applegate, D. L., Bixby, R. E., Chvatál, V., and Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press. URL: https://www.jstor.org/stable/j.ctt7s8xg.

Arthanari, T. S. (1982). "On the traveling salesman problem". Paper presented at the XI International Symposium on Mathematical Programming (23-27 August 1982; Bonn. Germany).

Arthanari, T. S. (2005). "Pedigree polytope is a combinatorial polytope". In: *Operations Research with Economic and Industrial Applications: Emerging Trends*. Ed. by S. R. Mohan and S. K. Neogy. Anamaya Publishers.

Arthanari, T. S. (2006). "On pedigree polytopes and Hamiltonian cycles". *Discrete Mathematics* **306**(14), 1474–1492. DOI: 10.1016/j.disc.2005.11.030.

Arthanari, T. S. (2008). "On the membership problem of pedigree polytope". In: *Mathematical Programming and Game Theory for Decision Making*. Ed. by S. K. Neogy, R. B. Bapat, A. K. Das, and T. Parthasarathy. World Scientific. DOI: 10.1142/9789812813220_0006.

Arthanari, T. S. (2013a). "On pedigree polytope and its properties". *Atti della Accademia Peloritana dei Pericolanti. Classe di Scienze Fisiche, Matematiche e Naturali* **91**(S2), 91S2A3 [24 pages]. DOI: 10.1478/AAPP.91S2A3.

Arthanari, T. S. (2013b). "Study of the pedigree polytope and a sufficiency condition for nonadjacency in the tour polytope". *Discrete Optimization* **10**(3), 224–232. DOI: 10.1016/j.disopt.2013.07.001.

Arthanari, T. S. and Qian, K. (2018). "Symmetric Travelling Salesman Problem". In: *Mathematical Programming and Game Theory*. Ed. by S. K. Neogy, R. B. Bapat, and D. Dubey. Springer, pp. 87–114. DOI: 10.1007/978-981-13-3059-9_5.

Arthanari, T. S. and Usha, M. (2001). "On the equivalence of the multistage-insertion and cycle shrink formulations of the symmetric traveling salesman problem". *Operations Research Letters* **29**(3), 129–139. DOI: 10.1016/S0167-6377(01)00088-8.

Arthanari, T. S. and Usha, M. (2000). "An alternate formulation of the symmetric traveling salesman problem and its properties". *Discrete Applied Mathematics* **98**(3), 173–190. DOI: 10.1016/S0166-218X(99)00154-7.

Barahona, F. and Anbil, R. (2000). "The volume algorithm: producing primal solutions with a subgradient method". *Mathematical Programming* **87**, 385–399. DOI: 10.1007/s101070050002.

Bertsimas, D. and Orlin, J. B. (1994). "A technique for speeding up the solution of the Lagrangean dual". *Mathematical Programming* **63**(1-3), 23–45. DOI: 10.1007/BF01582057.

Cambini, R., Gallo, G., and Scutellà, M. G. (1992). *Minimum cost flows on hypergraphs*. Tech. rep. TR-1/92. Università degli Studi di Pisa.

Carr, R. (1995). "Polynomial separation procedures and facet determination for inequalities of the traveling salesman polytope". PhD thesis. Carnegie Mellon University.

Carr, R. (1996). "Separating over classes of TSP inequalities defined by 0 node-lifting in polynomial time". In: *Integer Programming and Combinatorial Optimization. IPCO 1996*. Ed. by W. H. Cunningham, S. T. McCormick, and Q. M. Vol. 1084. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 460–474. DOI: 10.1007/3-540-61310-2_34.

Claus, A. (1984). "A new formulation for the travelling salesman problem". *SIAM Journal on Algebraic and Discrete Methods* **5**(1), 21–25. DOI: 10.1137/0605004.

Concorde TSP Solver (2018). URL: http://www.math.uwaterloo.ca/tsp/concorde/index.html (visited on 12/14/2018).

Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M. (1954). "Solution of a large-scale traveling-saesman problem". *Operations Research* **2**(4), 393–410. DOI: 10.1287/opre.2.4.393.

Edmonds, J. and Giles, R. (1977). "A min-max relation for submodular functions on graphs". In: *Studies in Integer Programming*. Ed. by P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser. Vol. 1. Annals of Discrete Mathematics. Elsevier, pp. 185–204. DOI: 10.1016/S0167-5060(08)70734-9.

Fischetti, M., Lodi, A., and Toth, P. (2003). "Solving Real-World ATSP Instances by Branch-and-Cut". In: *Combinatorial Optimization — Eureka, You Shrink!: Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*. Ed. by M. Jünger, G. Reinelt, and G. Rinaldi. Berlin, Heidelberg: Springer, pp. 64–77. DOI: 10.1007/3-540-36478-1_8.

Flood, M. M. (1956). "The traveling-salesman problem". *Operations Research* **4**(1), 61–75. DOI: 10.1287/opre.4.1.61.

Fox, K. R., Gavish, B., and Graves, S. C. (1980). "Technical note – An *n*-constraint formulation of the (time-dependent) travelling salesman problem". *Operations Research* **28**(4), 1018–1021. DOI: 10.1287/opre.28.4.1018.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co. URL: https://dl.acm.org/citation.cfm?id=574848.

Gavish, B. and Graves, S. C. (1978). *The travelling salesman problem and related problems*. Tech. rep. OR-078-78. Massachusetts Institute of Technology, Operations Research Center. URL: https://dspace.mit.edu/handle/1721.1/5363.

Geoffrion, A. M. (1974). "Lagrangean relaxation for integer programming". In: *Approaches to Integer Programming*. Ed. by M. Balinski. Vol. 2. Mathematical Programming Studies. Berlin, Heidelberg: Springer, pp. 82–114. DOI: 10.1007/BFb0120690.

Godinho, M. T., Gouveia, L., and Pesneau, P. (2014). "Natural and extended formulations for the time-dependent traveling salesman problem". *Discrete Applied Mathematics* **164**, 138–153. DOI: 10.1016/j.dam.2011.11.019.

Gouveia, L. and Pires, J. M. (1999). "The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints". *European Journal of Operational Research* **112**(1), 134–146. DOI: 10.1016/S0377-2217(97)00358-5.

Gouveia, L. and Voß, S. (1995). "A classification of formulations for the (time-dependent) traveling salesman problem". *European Journal of Operational Research* **83**(1), 69–82. DOI: 10.1016/0377-2217(93)E0238-S.

Gubb, M. (2003). *Flows, Insertions and Subtours Modelling the Travelling Salesman*. Tech. rep. Part IV Project. Engineering Science & Biomedical Engineering, University of Auckland, New Zealand.

Guignard, M. (2003). "Lagrangean relaxation". *Top* **11**(2), 151–200. DOI: 10.1007/BF02579036.

Gutin, G. and Punnen, A., eds. (2007). *The Traveling Salesman Problem and Its Variations*. Vol. 12. Combinatorial Optimization. Boston, MA: Springer. DOI: 10.1007/b101971.

Haerian Ardekani, L. (2011). "New Insights on the Multistage Insertion Formulation of the Traveling SalesmanProblem – Polytopes, Experiments, and Algorithm". PhD thesis. University of Auckland, New Zealand. URL: https://researchspace.auckland.ac.nz/handle/2292/6981.

Haerian Ardekani, L. and Arthanari, T. S. (2008). "Traveling salesman problem and membership in pedigree polytope - A numerical illustration". In: *Modelling, Computation and Optimization in Information Systems and Management Science. MCO 2008*. Ed. by H. A. Le Thi, P. Bouvry, and Pham Dinh, T. Vol. 14. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, pp. 145–154. DOI: 10.1007/978-3-540-87477-5_16.

Held, M. and Karp, R. M. (1970). "The travelling salesman problem and minimum spanning trees". *Operations Research* **18**(6), 1138–1162. DOI: 10.1287/opre.18.6.1138.

Held, M. and Karp, R. M. (1971). "The travelling salesman problem and minimum spanning trees: Part II". *Mathematical Programming* **1**(1), 6–25. DOI: 10.1007/BF01584070.

Jeroslow, R. G., Martin, R. K., Rardin, R. L., and Wang, J. (1992). "Gainfree Leontief substitution flow problems". *Mathematical Programming* **57**(1-3), 375–414. DOI: 10.1007/BF01581090.

Jünger, M., Reinelt, G., and Rinaldi, G. (1997). "The traveling salesman problem". In: *Annotated Bibliographies in Combinatorial Optimization*. Ed. by M. Dell'Amico, F. Maffioli, and S. Martello. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons, pp. 199–221.

Korte, B. and Vygen, J. (2012). *Combinatorial Optimization. Theory and Algorithms*. 5th ed. Vol. 21. Algorithms and Combinatorics. Berlin, Heidelberg: Springer. DOI: 10.1007/3-540-29297-7.

Langevin, A., Soumis, F., and Desrosiers, J. (1990). "Classification of travelling salesman problem formulations". *Operation Research Letters* **9**(2), 127–132. DOI: 10.1016/0167-6377(90)90052-7.

Laporte, G. (1992). "The traveling salesman problem: An overview of exact and approximate algorithms". *European Journal of Operational Research* **59**(2), 231–247. DOI: 10.1016/0377-2217(92)90138-Y.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H., and Shmoys, D. B. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons.

Lemaréchal, C. (2003). "The omnipresence of Lagrange". *4OR - A Quarterly Journal of Operations Research* **1**(1), 7–25. DOI: 10.1007/s10288-002-0003-1.

Lorena, L. A. N. and Narciso, M. G. (2002). "Using logical surrogate information in Lagrangean relaxation: An application to symmetric traveling salesman problems". *European Journal of Operational Research* **138**(3), 473–483. DOI: 10.1016/S0377-2217(01)00159-X.

Makkeh, A., Pourmoradnasseri, M., and Theis, D. O. (2016). "On the graph of the pedigree polytope". arXiv: 1611.08431.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). "Integer programming formulations of traveling salesman problems". *Journal of the Association for Computing Machinery* **7**(4), 326–329. DOI: 10.1145/321043.321046.

Naddef, D. (2007). "Polyhedral theory and branch-and-cut algorithms for the symmetric TSP." In: *The Traveling Salesman Problem and its Variations*. Ed. by G. Gutin and A. P. Punnen. Vol. 12. Combinatorial Optimization. Boston, MA: Springer. DOI: 10.1007/0-306-48213-4_2.

Naddef, D. and Pulleyblank, W. R. (1981). "Hamiltonicity and combinatorial polyhedra". *Journal of Combinatorial Theory* **B 31**(3), 297–312. DOI: 10.1016/0095-8956(81)90032-0.

Nedić, A. and Ozdaglar, A. (2009). "Approximate Primal Solutions and Rate Analysis for Dual Subgradient Methods". *SIAM Journal on Optimization* **19**(4), 1757–1780. DOI: 10.1137/070708111.

Öncan, T., Altınel, İ. K., and Laporte, G. (2009). "A comparative analysis of several asymmetric traveling salesman problem formulations". *Computers & Operations Research* **36**(3), 637–654. DOI: 10.1016/j.cor.2007.11.008.

Orman, A. J. and Williams, H. P. (2007). "A survey of different integer programming formulations of the travelling salesman problem". In: *Optimization, Econometric and Financial Analysis*. Ed. by E. J. Kontoghiorghes and C. Gatu. Vol. 9. Advances in Computational Management Science. Berlin, Heidelberg: Springer, pp. 91–104. DOI: 10.1007/3-540-36626-1_5.

Padberg, M. and Sung, T.-Y. (1991). "An analytical comparison of different formulations of the traveling salesman problem". *Mathematical Programming* **52**(1-3), 315–357. DOI: 10.1007/BF01582894.

Papadimitriou, C. H. and Steiglitz, K. (1977). "On the complexity of local search for the traveling salesman problem". *SIAM Journal on Computing* **6**(1), 76–83. DOI: 10.1137/0206005.

Pashkovich, K. and Weltge, S. (2015). "Hidden vertices in extensions of polytopes". *Operations Research Letters* **43**(2), 161–164. DOI: 10.1016/j.orl.2015.01.004.

Poljak, B. T. (1967). "A general method for solving extremal problems". *Doklady Akademii Nauk SSSR* **174**(1), 33–36. URL: http://www.mathnet.ru/eng/dan33049.

Poljak, B. T. (1969). "Minimization of unsmooth functionals". *USSR Computational Mathematics and Mathematical Physics* **9**(3), 14–29. DOI: 10.1016/0041-5553(69)90061-5.

Punnen, A. P. (2007). "The traveling salesman problem: Applications, formulations and variations". In: *The Traveling Salesman Problem and Its Variations*. Ed. by G. Gutin and A. P. Punnen. Vol. 12. Combinatorial Optimization. Boston, MA: Springer, pp. 1–28. DOI: 10.1007/0-306-48213-4_1.

Reinelt, G. (1991). "TSPLIB – A traveling salesman problem library". *ORSA Journal on Computing* **3**(4), 276–384. DOI: 10.1287/ijoc.3.4.376.

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Vol. 840. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. DOI: 10.1007/3-540-48661-5.

Roberti, R. and Toth, P. (2012). "Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison". *EURO Journal on Transportation and Logistics* **1**(1-2), 113–133. DOI: 10.1007/s13676-012-0010-0.

Sarin, S. C., Sherali, H. D., and Bhootra, A. (2005). "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints". *Operations Research Letters* **33**(1), 62–70. DOI: 10.1016/j.orl.2004.03.007.

Scutellà, M. G., Gallo, G., and Licheri, F. (1996). "The hypergraph simplex approach: Some experimental results". *Ricerca Operativa* **26**(78), 21–54.

Sherali, H. D. and Driscoll, P. J. (2002). "On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems". *Operations Research* **50**(4), 656–669. DOI: 10.1287/opre.50.4.656.2865.

Sherali, H. D., Sarin, S. C., and Tsai, P.-F. (2006). "A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints". *Discrete Optimization* **3**(1), 20–32. DOI: 10.1016/j.disopt.2005.10.004.

Vaidya, P. M. (1990). "An algorithm for linear programming which requires $O((m+n)n^2+(m+n)^{1.5}nL)$ arithmetic operations". *Mathematical Programming* **47**(1-3), 175–201. DOI: 10.1007/BF01580859.

Valenzuela, C. L. and Jones, A. J. (1997). "Estimating the Held-Karp lower bound for the geometric TSP". *European Journal of Operational Research* **102**(1), 157–175. DOI: 10.1016/S0377-2217(96)00214-7.

Veinott Jr., A. F. (1968). "Extreme points of Leontief substitution systems". *Linear Algebra and Its Applications* **1**(2), 184–194. DOI: 10.1016/0024-3795(68)90002-5.

Wong, R. T. (1980). "Integer programming formulations of the traveling salesman problem". In: *Proceedings of the IEEE International Conference on Circuits and Computers*, pp. 149–152.

Zamani, R. and Lau, S. K. (2010). "Embedding learning capability in Lagrangean relaxation: An application to the travelling salesman problem". *European Journal of Operational Research* **201**(1), 82–88. DOI: 10.1016/j.ejor.2009.02.008.

* University of Auckland
Auckland, New Zealand

Email: t.arthanari@auckland.ac.nz

Manuscript received 19 September 2017; published online 10 January 2019