Transformations of Generalized ATSP into ATSP

D. Ben-Arieh
Department of Industrial and Manufacturing Systems Engineering, Kansas State University, USA
G. Gutin*
Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
M. Penn
Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel
A. Yeo, A. Zverovitch
Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK

Abstract

The Generalized Traveling Salesman Problem (GTSP) is stated as follows. Given a weighted complete digraph K_n^* and a partition $V_1, \dots, V_k$ of its vertices, find a minimum weight cycle containing exactly one vertex from each set V_i, $i = 1, \dots, k$. We study transformations from GTSP to TSP. The 'exact' Noon-Bean transformation is investigated in computational experiments. We study the 'non-exact' Fischetti-Salazar-Toth (FST) transformation and its two modifications in computational experiments and theoretically using domination analysis. One of our conclusions is that one of the modifications of the FST transformation is better than the original FST transformation in the worst case in terms of domination analysis.

Keywords: TSP, Generalized TSP, domination analysis, heuristics.

1 Introduction

The *Generalized Traveling Salesman Problem* (GTSP) is stated as follows. Given a weighted complete digraph K_n^* and a partition $V_1, \dots, V_k$ of its vertices, find a minimum weight cycle containing exactly one (at least one) vertex from each set V_i, $i = 1, \dots, k$. We assume that every arc uv of K_n^* is assigned a weight $c(u, v)$; the weight of a cycle is the sum of the weights of its arcs. The sets V_i will be called *clusters*. The GTSP is sometimes called the International TSP [7].

In this paper we consider only the 'exactly one' variant of the GTSP (i.e., every *tour* of GTSP has exactly one vertex from each cluster V_i) and with no restrictions imposed on the weights of the complete digraph. We call the Asymmetric TSP simply the TSP.

Since the GTSP is an extension of the TSP, it is natural to try to solve the GTSP by reducing it to the TSP. This way to tackle the GTSP is practically motivated by the fact that there exist a large variety of exact and heuristic algorithms for the TSP, see, e.g., [10, 16]. In this paper, we show, by computational experiments, that the Noon-Bean transformation from the GTSP to the TSP, can be used to solve to optimality small to moderate instances of the GTSP. A similar conclusion for the Laporte-Semet transformation from the Symmetric GTSP to the Symmetric TSP was derived in [15]. We used the Noon-Bean transformation to solve GTSP instances obtained from an industrial application in [2].

Fischetti, Salazar and Toth [5] (see also, [7]) suggested a 'non-exact' transformation from the GTSP to TSP that allows to obtain heuristic solutions to the GTSP using exact and heuristic algorithms for the TSP. In this paper we provide a computational evaluation of this approach as well as of its two modifications. The experiments indicate that the original Fischetti-Salazar-Toth (FST) transformation, which we call the minimum FST transformation, and one of our modifications, which we call the average FST transformation, often produce results of similar quality and, in general, outperform our other modification of the FST transformation.

*Corresponding author, G.Gutin@rhul.ac.uk

We prove, using a recent theoretical approach introduced by Glover and Punnen [9] and called domination analysis (see, e.g., [11, 21] and references therein), that the minimum FST transformation is inferior to the average FST transformation, in the worst case in the measure of domination analysis. Thus, we conclude that the average FST transformation may be recommended, on its own or together with the minimum FST transformation, as a good way to obtain approximate solutions for the GTSP.

The GTSP has many applications, see, e.g., [2, 18, 19, 12]. The GTSP was studied in [13, 19]; its symmetric weight version was investigated in [5, 6, 14, 24, 25]; an informative account on the symmetric GTSP is in [7]. Transformations from the GTSP to the TSP were provided in [19, 17, 3, 15].

2 Testbed for Experiments

The testbed comprises all asymmetric instances from TSPLIB [23] converted to GTSP instances using the procedure introduced by Fischetti, Salazar and Toth [6]. The vertex clustering have been done to simulate geographical regions. The number of clusters is $k = \lceil n/5 \rceil$. The k centers of clusters are chosen according to the following procedure. Let the distance from a vertex x_j to a set S of vertices be the minimum of $c(x_j, u),\ u \in S$. The first center w_1 is the vertex furthest to the vertex x_1. The ith center w_i is the vertex furthest to the set $\{w_1, \dots, w_{i-1}\}$. If several vertices have the same (smallest) distance from the current set of centers, the vertex with the smallest index is chosen as the next center.

A vertex x_j belongs to the cluster with the center w_i closest to x_j. If there are several centers equidistant from x_j, the center w_i of the smallest index is chosen.

We have made available instances of GTSP that we generated from TSPLIB instances by using this method. These instances are available for download from www.cs.rhul.ac.uk/home/zvero/GTSPLIB/ The instances are named in the form *Xname*, where X is the number of clusters and *name* is the name of the original TSPLIB instance. All algorithms were coded in C++ and the experiments were done on an Intel Celeron 550MHz computer with 384 megabytes of RAM.

3 Noon-Bean Transformation

To describe the Noon-Bean transformation [20] we will use the same notation as in Section 1: K_n^* denotes a complete digraph; every arc uv is assigned a weight $c(u, v)$; the vertices of K_n^* are partitioned into clusters $V_1, \dots, V_k$. The transformation introduces a new weight function c' in K_n^* such that the minimum c'-weight Hamilton cycle in K_n^* is of the same weight as the minimum c-weight tour in K_n^*. (Recall that a GTSP tour contains exactly one vertex from every cluster.)

A GTSP instance is converted to an instance of the TSP. The number of vertices stays the same. Weights are modified in such a way that guarantees that an optimal tour visits all vertices that belong to the same cluster in the original problem before moving on to the next cluster. This is achieved by adding a large constant M to the weight of each inter-cluster arc. If the constant is large enough (it suffices that M is greater than the sum of n heaviest arcs in K_n^*), an optimal tour would contain exactly k such heavy arcs, thus ensuring that no cluster is visited more than once.

Vertices within each cluster are visited in a prescribed "circular" order. Vertices of a cluster are first numbered in an arbitrary way, $v_1, ..., v_r$. Weights of arcs that connect consecutive vertices ($v_i v_{i+1}$, including $v_r v_1$) are set to 0. Weight of all other arcs within the cluster is set to $2M$. This ensures that, when the cluster is entered at vertex v_s, the remaining vertices are traversed in the following order: $v_{s+1} v_{s+2} \dots v_r v_1 v_2 \dots v_{s-1}$. Having entered the cluster at a vertex v_s, the optimal tour always traverses all other vertices of the cluster and then exits the cluster at the previous vertex, v_{s-1}. The optimal tour has zero weight inside the cluster.

There is one last change in weights that is required to ensure that each 'good' tour on the transformed instance has the same weight modulo kM as the corresponding GTSP tour on the original instance. (By a 'good' tour here we mean a tour that uses exactly k arcs of weight M and no $2M$ arcs.) This change is reassigning the weight of each inter-cluster arc $u_i v_j$ to $c(u_{i+1}, v_j)$.

The Noon-Bean transformation allows one to solve GTSP instances if one has an (Asymmetric) TSP solver. Unfortunately, we did not have access to such a solver. We had access to `Concorde`, a well-known Symmetric TSP solver described in [1]. To transform an Asymmetric TSP instance to an 'equivalent' Symmetric one, we used the following standard reduction: Replace every vertex x_i in K_n^*

Instance	Optimal solution	Solution time (s)
4br17	31	0.11
7ftv33	476	3.89
8ftv35	525	2.52
8ftv38	511	11.23
9p43	5563	39.52
9ftv44	510	75.91
10ftv47	569	38.14
10ry48p	6284	214.81
11ft53	2648	82.58
12ftv55	689	218.19
13ftv64	708	107.81
14ft70	7707	11.38
15ftv70	594	8.12
20kro124p	11203	809.42
35ftv170	1205	820.69
65rbg323	471	27.86
72rbg358	693	31.25
81rbg403	1170	45.28
89rbg443	632	48.06

Table 1: Solving GTSP instances using the Noon-Bean transformation

by three vertices x_i^-, x_i^0, x_i^+ (in a complete undirected graph U on $3n$ vertices); the weights of edges of the form $x_i^- x_i^0$ and $x_i^0 x_i^+$ are set to 0; the weight of every edge of the form x_i^+, x_j^- is set to $c'(x_i, x_j)$; every other edge in U gets weight $3M$.

The results of our experiment with the Noon-Bean transformation are given in Table 1.

Table 1 shows that the use of the Noon-Bean transformation allows one to solve small to moderate instances of GTSP with the running time depending heavily on the weight function, and not only on the number of vertices, n. Nevertheless, the running time grows very quickly when n increases, on average, and the transformation is likely to be impractical for many large instances of GTSP.

A good branch-and-cut specialized Asymmetric TSP solver, like the one described in [4], may well bring down the running time due to both the smaller size of the transformed Asymmetric TSP instances (three times smaller than the Symmetric ones) and the fact that the Asymmetric instances are less 'exotic' (because of more restricted use of large constants).

4 Fischetti-Salazar-Toth Transformation and Its Modifications

Recall that the vertex set of K_n^* is partitioned into clusters $V_1, \dots, V_k$. If $W_i \subseteq V_i$, $i = 1, \dots, k$, and $i_1, i_2, \dots, i_k$ is a permutation of $1, 2, \dots, k$, then $C(W_{i_1}, W_{i_2}, \dots, W_{i_k})$ is the subdigraph of K_n^* with vertex set $\cup_{i=1}^k W_i$ and arc set $\cup_{j=1}^k \{xy : \ x \in W_{i_j},\ y \in W_{i_{j+1}}\}$, where $i_{k+1} = i_1$. The Fischetti-Salazar-Toth (FST) transformation is the following generic algorithm:

1. For every pair $i, j;\ 1 \le i \ne j \le k$ compute $w(i,j) = \min\{c(v_i, v_j) :\ v_i \in V_i, v_j \in V_j\}$. Construct a weighted complete digraph K_k^* with vertices $\{1, 2, \dots, k\}$ and weights $w(i,j)$.

2. Using a TSP algorithm or heuristic find a Hamilton cycle $i_1 i_2 \dots i_k i_1$ in K_k^* (optimal or otherwise).

3. Find a lightest k-cycle in $C(V_{i_1}, V_{i_2}, \dots, V_{i_k})$ as follows. Let $V_1 = \{x_1, \dots, x_s\}$. For each $j = 1, \dots, s$, compute a minimum weight k-cycle Z_j in $C(\{x_j\}, V_{i_2}, V_{i_3}, \dots, V_{i_k})$ as follows. Delete x_j from $C(\{x_j\}, V_{i_2}, V_{i_3}, \dots, V_{i_k})$ and add, to the remaining digraph, a pair u_j, v_j of vertices together with arcs $\{u_j v :\ v \in V_{i_2}\} \cup \{v v_j :\ v \in V_{i_k}\}$ with weights $c(u_j, y) = c(x_j, y)$, $c(z, v_j) = c(z, x_j)$, where $y \in V_{i_2}$, $z \in V_{i_k}$. In the obtained digraph, find a minimum weight path P_j from u_j to v_j

and transform P_j into Z_j by replacing u_j, v_j with x_j. Find the minimum weight cycle among $Z_1, Z_2, \ldots, Z_s$.

We also consider the following modification of the FST transformation. Let $c(V_i, V_j)$ be the total weight of arcs from V_i to V_j. In Step 1 compute $w(i, j)$ as follows: For every pair i, j; $1 \leq i \neq j \leq k$ compute $w(i, j) = c(V_i, V_j)/(n_i n_j)$, the average weight of an arc from V_i to V_j. To distinguish between the original FST transformation and its modification we call them the *minimum* FST transformation and *average* FST transformation, respectively. Also, we will consider the *10-average* FST transformation, where $w(i, j)$ is computed as the average of the ten smallest weights of arcs between V_i and V_j (if $|V_i||V_j| < 5$, $w(i, j)$ is just the average of all weights of the arcs between V_i and V_j).

5 Computational Experiments with FST Transformations

We implemented the three types of the FST transformation together with three TSP heuristics used in Step 2: `greedy` (the greedy algorithm), `insertion` (the random vertex insertion algorithm) and `karp` (Karp's patching algorithm). The three algorithms are very popular TSP heuristics and are described in, e.g., [8]. Nevertheless, we provide short descriptions of the heuristics for the sake of completeness.

The greedy algorithm considers the arcs of K_k^* in non-decreasing order of their weights. It retains the arc uv under consideration if and only if uv together with previously retained arcs form a subset of arcs of a Hamilton cycle in K_k^*.

The heuristic `insertion` chooses randomly two initial vertices i_1 and i_2 in K_k^* and forms the cycle $i_1 i_2 i_1$. Then, in every iteration, `insertion` chooses randomly a vertex ℓ of K_k^* which is not in the current cycle $i_1 i_2 ... i_s i_1$ and inserts ℓ in the cycle (i.e., replaces an arc $i_m i_{m+1}$ of the cycle with the path $i_m \ell i_{m+1}$) such that the weight of the cycle increases as little as possible. The heuristic stops when all vertices have been included in the current cycle.

Karp's patching algorithm starts from finding a lightest disjoint collection F of cycles in K_k^* covering all vertices of K_k^*. Then, `karp` iterates by choosing a pair of cycles C', C'' in F with maximum number of vertices, patching them into one cycle C of minimum possible weight, and replacing, in F, the cycles C', C'' with C. The operation of patching of C' and C'' consists of deleting an arc in C' and an arc in C'', and appending two arcs between C' and C'' such that the result is a cycle through all vertices of C; and C''. There are $|C'||C''|$ such patchings, but `karp` chooses the one that results in the lightest cycle.

The results of our computational experiments on the testbed introduced above are provided in Tables 2-4, where Column 2 (3,4) is for the minimum (10-average, average) transformation results. The results are the excesses (in percent) of the heuristic solutions over the optimal ones.

One can clearly see that `greedy` is not good for the FST transformation as it is inferior to the other two heuristics. The best heuristic seems to be `karp`. Now consider Table 4. The table indicates that the 10-average FST transformation is inferior to the other two transformations and the two transformations seem to produce solutions of similar quality. Thus, according to the experiments the best choices seem to be `karp` with either the minimum or average FST transformation. Moreover, to get further improvements, one may combine the two transformations, i.e. run both and choose the best solution among them. The instances of Table 4 give the average of 7.32 % and the maximum of 17.56 % for the combined transformations. This is a significant improvement over the two separate transformations.

The three tables do not have information on the running times. This is due to the fact that the running times are very small. Most of the running times are within one hundredth of a second. The slowest algorithm is `greedy`, which takes up to 0.2 seconds for the largest instances on test.

Certainly, one should keep in mind that our testbed, as testbeds in practically all computational experiments, is very restricted. Thus, our conclusions regarding the choice of the heuristic and transformations should be considered in the light of the above natural limitation.

In the next section, we analyze the two 'best' transformations from the point of view of domination analysis. Certainly, domination analysis (DA) has its limitations too, but its advantages are the facts that its measure is objective as it does not depend on the chosen testbed (unless one restricts the

Instance	min (%)	avg10 (%)	avg (%)
4br17	116.13	0.00	116.13
7ftv33	9.45	29.83	11.34
8ftv35	29.14	10.29	10.29
8ftv38	27.40	27.79	30.72
9p43	1.46	1.60	0.67
9ftv44	51.76	29.61	9.41
10ftv47	50.97	19.51	31.11
10ry48p	27.85	8.56	13.46
11ft53	10.73	24.89	17.75
12ftv55	49.64	16.98	10.60
13ftv64	67.37	52.40	39.12
14ft70	11.26	22.75	16.62
15ftv70	76.60	50.67	13.47
20kro124p	30.70	35.29	22.36
35ftv170	61.16	69.54	52.37
65rbg323	35.24	68.58	42.89
72rbg358	14.29	40.55	29.15
81rbg403	11.37	23.59	16.07
89rbg443	61.08	75.95	54.43
Average	39.14	32.02	28.31
Max	116.13	75.95	116.13

Table 2: Results of `greedy`

set of instances under consideration) and DA makes us aware of the worst possible outcomes of the computations (in terms of the DA measure).

6 Domination Analysis

In this section we compare theoretically the two 'best' FST transformations. In our comparison we will use the domination number, which is a measure used in domination analysis. The domination number can be defined for algorithms for many combinatorial optimization problems, but we restrict ourselves to the GTSP and moreover to the GTSP, where all k clusters are of the same cardinality m. The reader may easily extend our definition to other problems.

The *domination number* of a GTSP heuristic $\mathcal{A}$, $dom(\mathcal{A}) = dom(\mathcal{A}, m, k)$, is the maximum integer $d(m, k)$ such that, for every instance of GTSP with m vertices in each of k clusters, $\mathcal{A}$ computes a tour that is not worse than at least $d(m, k) - 1$ other tours. In particular, the 'random choice' heuristic, in the worst case, produces the unique worst possible solution, i.e., this heuristic is of domination number 1. As with approximation ratio, it is natural to say that a GTSP heuristic $\mathcal{A}$ is better than a GTSP heuristic $\mathcal{B}$ if $dom(\mathcal{A}, m, k) > dom(\mathcal{B}, m, k)$ for every $m \geq c_1$ and $k \geq c_2$, where c_1 and c_2 are 'relatively small' constants.

Comparing the minimum and average FST transformations, we will make some assumptions. First, we assume that $m \geq 2$ since in the case of $m = 1$ both transformations do the same. Also, we will assume that the minimum FST transformation finds a *optimal* cycle in Step 2 (and we call the minimum FST transformation with this assumption `minFST`), while the average FST transformation uses any TSP heuristic of domination number at least $(k-2)!$ (and we call the average FST transformation with this assumption `averFST`). Despite our last assumption, we will prove that $dom(\texttt{minFST}, m, k) < dom(\texttt{averFST}, m, k)$ for any $m \geq 2$ and $k \geq 4$.

Notice that the assumption that `averFST` uses a TSP heuristic of domination number at least $(k-2)!$ allows us to keep the time complexity of `averFST` polynomial: there are several polynomial time TSP heuristics with domination number at least $(k-2)!$. Among such heuristics (for all least $k \neq 6$) are `insertion`, a modification of `karp`, and many local search TSP heuristics (even when they

Instance	min (%)	avg10 (%)	avg (%)
4br17	0.00	0.00	0.00
7ftv33	14.08	11.34	11.34
8ftv35	0.00	10.29	0.00
8ftv38	17.81	0.98	11.35
9p43	0.47	2.05	0.47
9ftv44	12.16	29.02	12.55
10ftv47	7.03	16.87	16.87
10ry48p	1.43	11.22	1.43
11ft53	22.66	19.64	15.07
12ftv55	1.16	28.74	2.90
13ftv64	9.18	49.01	13.14
14ft70	14.31	21.32	15.70
15ftv70	1.18	49.83	25.59
20kro124p	19.63	20.21	10.78
35ftv170	27.72	80.83	15.10
65rbg323	26.96	64.76	32.91
72rbg358	14.86	57.72	20.20
81rbg403	5.30	22.56	8.21
89rbg443	37.34	73.10	31.01
Average	12.28	29.97	12.88
Max	37.34	80.83	32.91

Table 3: Results of `insertion`

are restricted to some polynomial running time, in which case they may not even find a local minimum yet) [11, 21, 22]. The $k \neq 6$ constraint is due to a minor shortcoming of a method that allows us to obtain such bounds on domination number, and thus we strongly believe that the $k \neq 6$ constraint is unnecessary. Moreover, in constant time, one can consider all tours for $k = 6$, if needed.

In this section, it will be more convenient to consider the GTSP on the complete k-partite digraph obtained from a complete digraph K_n^* with clusters $V_1, \dots, V_k$ by deleting arcs between vertices from the same cluster. (The operation of deletion does not change the problem since no deleted arc can be in a tour of GTSP.) In what follows, D is a weighted complete k-partite digraph with weight function $c: A(D) \rightarrow \mathbb{R}$, where $A(D)$ is the arc set of D; $V_1, V_2, \dots, V_k$ are the partite sets of D, each of the same cardinality $m \geq 2$.

The following three lemmas will be used to prove Theorem 6.4 that provides a lower bound on dom(`averFST`).

Lemma 6.1 *The number of k-cycles in D equals $(k-1)!\ m^k$.*

Proof: To avoid counting any k-cycle of D more than once, we assume that every such cycle starts at a vertex in V_1. It remains to observe that there are m^k k-cycles in $C(V_1, V_{i_2}, \dots, V_{i_k})$ for every permutation $i_2, i_3, \dots, i_k$ of $2, 3, \dots, k$ and there are $(k-1)!$ such permutations. □

Lemma 6.2 *There exists a decomposition of $A(C(V_1, \dots, V_k))$ into k-cycles.*

Proof: We prove this lemma by induction on k. The lemma is clearly true for $k = 2$, as we simply take all possible 2-cycles.

We will now show that it is true for $k = 3$. It is well-known that the edge set of any complete bipartite graph, with m vertices in each partite set, can be decomposed into m perfect matchings. So let $M_1, M_2, \dots, M_m$ be the perfect matchings obtained, when we consider the underlying graph of the subdigraph of $C(V_1, V_2, V_3)$ induced by $V_1 \cup V_2$. Now let $V_3 = \{x_1, x_2, \dots, x_m\}$, and for each $i \in \{1, 2, \dots, m\}$ orient all arcs from V_1 to V_2 in M_i, and for each arc obtained in this way, add the 2-path, starting at the end-point of the arc, going through x_i, and ending at the starting point of the arc. In this manner we obtain a number of 3-cycles. If we do this for all i, then we obtain the desired decomposition.

Instance	min (%)	avg10 (%)	avg (%)
4br17	0.00	0.00	0.00
7ftv33	10.92	11.34	11.34
8ftv35	21.90	10.29	10.29
8ftv38	17.42	10.57	10.57
9p43	0.05	0.77	0.05
9ftv44	21.18	22.16	8.43
10ftv47	0.00	25.83	10.54
10ry48p	2.74	0.00	1.24
11ft53	24.02	22.24	9.18
12ftv55	1.16	34.69	0.00
13ftv64	13.42	70.90	13.28
14ft70	13.26	17.67	12.65
15ftv70	0.00	108.42	13.30
20kro124p	11.22	14.83	7.43
35ftv170	26.64	43.65	15.85
65rbg323	10.83	58.39	17.20
72rbg358	7.50	45.17	16.31
81rbg403	3.33	22.65	5.73
89rbg443	17.56	55.70	29.27
Average	10.69	30.28	10.14
Max	26.64	108.42	29.27

Table 4: Results of `karp`

We will now prove the induction step, so assume that $k \geq 4$. Let $DC = C_1 \cup C_2 \cup \ldots \cup C_{m^2}$ be a $(k-2)$-cycle decomposition of $C(V_1, \ldots, V_{k-2})$ that exists by the induction hypothesis. Let $V_i = \{x_1^i, x_2^i, \ldots, x_m^i\}$, $i = 1, 2, \ldots, k$ and let Z_{ij} be the cycle in DC that passes through the arc (x_i^{k-2}, x_j^1), for any given i, j. We can insert, in every Z_{ij}, the path $x_i^{k-2} x_j^{k-1} x_i^k x_j^1$ obtaining a k-cycle decomposition of $C(V_1, \ldots, V_k)$. □

Lemma 6.3 *There are at least m^{k-2} k-cycles in $C(V_1, \ldots, V_k)$, with weight greater than the average weight of a k-cycle in $C(V_1, \ldots, V_k)$, which is $c(A(C(V_1, \ldots, V_k)))/m^2$.*

Proof: By Lemma 6.2 we can find a k-cycle decomposition, $DC_1 = C_1 \cup C_2 \cup \ldots \cup C_d$, of $A(C(V_1, \ldots, V_k))$, where $d = |A(C(V_1, \ldots, V_k))|/k = m^2$. Let $\alpha : V(D) \to V(D)$ be an automorphism, such that vertices of V_i are mapped into vertices of V_i, for all $i = 1, 2, \ldots, k$. There exist $l = (m!)^k$ such automorphisms as the vertices in each partite set can be permuted in $m!$ ways. Let $\mathcal{DC} = \{DC_1, DC_2, \ldots, DC_l\}$ be the l decompositions of $A(C(V_1, \ldots, V_k))$ into k-cycles by using all possible automorphisms described above.

Let $T = x_1 x_2 \ldots x_k x_1$ be a k-cycle in $A(C(V_1, \ldots, V_k))$ and let a cycle C_i from DC_1 be $C_i = c_1 c_2 \ldots c_k c_1$. Observe that if some automorphism maps C_i into T, then the vertices $\{c_1, c_2, \ldots, c_k\}$ are mapped into the vertices $\{x_1, x_2, \ldots, x_k\}$. However for all other $m-1$ vertices in each partite set, there is total freedom, so there are exactly $((m-1)!)^k$ automorphisms that map C_i into T. Since there are $d = m^2$ cycles in DC_1, we conclude that T lies in exactly $m^2((m-1)!)^k$ different decompositions in $\mathcal{DC}$.

Observe that the average weight of a cycle in a decomposition DC_i is $c(A(C(V_1, \ldots, V_k)))/d$ which is also the average weight of a k-cycle in $A(C(V_1, \ldots, V_k))$. Therefore, the heaviest cycle in a decomposition DC_i, has weight greater than or equal to the average weight of a k-cycle in $A(C(V_1, \ldots, V_k))$.

Now let Q_i be the heaviest k-cycle in DC_i for every $i = 1, 2, \ldots, l$. Note that there are at least $l/(m^2((m-1)!)^k)$ distinct tours in $\{Q_1, Q_2, \ldots, Q_l\}$. So we are now done as $l/(m^2((m-1)!)^k) = (m!)^k/(m^2((m-1)!)^k) = m^{k-2}$.

□

Theorem 6.4 *dom(`averFST`) $\geq m^k + ((k-2)! - 1)m^{k-2}$ for every $k, m \geq 2$.*

Proof: Let K_k^* be the complete digraph constructed on Step 1 of averFST. Let H_1 be the tour in K_k^* found on Step 2 of averFST, and let $\mathcal{H} = \{H_1, H_2, \dots, H_{(k-2)!}\}$ be $(k-2)!$ distinct tours in K_k^*, which have weight more or equal to that of H_1. Let $H_i = v_{i,1}v_{i,2}\dots v_{i,k}v_{i,1}$, where $v_{i,s} \in V_{i,s}$. By Lemma 6.1, every $C(V_{i,1}, \dots, V_{i,k}, V_{i,1})$ has exactly m^k k-cycles.

Let Q be the k-cycle which averFST finds. By the definition of Step 3 of averFST, we see that there are m^k k-cycles in $C(V_{1,1}, \dots, V_{1,k}, V_{1,1})$ with weight greater than or equal to $c(Q)$ (as Q is optimal in $C(V_{1,1}, \dots, V_{1,k}, V_{1,1})$). For any $H_i \in \mathcal{H} - H_1$, Lemma 6.3 implies that there are at least m^{k-2} k-cycles with weight at least

$$c(A(C(V_{i,1}, \dots, V_{i,k}, V_{i,1})))/m^2 \geq c(A(C(V_{1,1}, \dots, V_{1,k}, V_{1,1})))/m^2 \geq c(Q).$$

Since there are $(k-2)! - 1$ tours in $\mathcal{H} - H$, this implies that there are at least $m^k + ((k-2)! - 1)m^{k-2}$ k-cycles with weight greater than or equal to $c(Q)$ in D. $\square$

The following lemma will be used in Theorem 6.6.

Lemma 6.5 $\binom{k-i}{i} \leq \sqrt{\frac{3}{\pi k}}\left(\frac{1+\sqrt{5}}{2}\right)^k$, *holds for all* $0 \leq k \leq n/2$.

Since a complete proof of this lemma is quite long and very technical, and not of great interest for this paper, we will only give a short outline of the proof. Let $\gamma(i) = \binom{k-i}{i}$, and by evaluating $\gamma(i+1) - \gamma(i)$ we note that $\gamma(i)$ takes its maximum value at $i_0 = \lfloor \frac{5k+7-\sqrt{5k^2+10k+9}}{10} \rfloor$. When $k \geq 36$ we note that $i_0 \in [0.25k, 0.29k]$. Now using Stirling's formula we see that $\binom{k-i}{i} \leq \sqrt{\frac{f(\alpha)}{2\pi k}} \times g(\alpha)^k$, where $i = \alpha k$, $f(\alpha) = \frac{1-\alpha}{\alpha(1-2\alpha)}$ and $g(\alpha) = \frac{(1-\alpha)^{1-\alpha}}{\alpha^\alpha(1-2\alpha)^{1-2\alpha}}$. In the interval $[0.25, 0.29]$, the maximum of $f(\alpha)$ is at $\alpha = 0.25$, and the maximum of $g(\alpha)$ is at $\alpha = \frac{5-\sqrt{5}}{10}$. So substituting these values into the above inequality, we get the inequality of Lemma 6.5. When $k \leq 35$, the lemma can be checked by computer.

Theorem 6.6 *Let z_k be the number of Hamilton cycles in $K_k^* - A(H)$, where H is a Hamilton cycle in K_k^*. Then*

$$\begin{aligned} dom(\texttt{minFST}) &\leq m^k + \sum_{i=3}^{\lfloor k/2 \rfloor}\left[\binom{k-i}{i}\left(\binom{k-i}{i} + \binom{k-i-1}{i-1}\right) z_i m^{k-i}\right] && \text{(i)} \\ &\leq m^k + \max_{3 \leq i \leq \lfloor k/2 \rfloor}\{2\binom{k-i}{i}^2 (k-1)!\} m^{k-2} && \text{(ii)} \\ &\leq m^k + \frac{6}{\pi k}\left(\frac{3+\sqrt{5}}{2}\right)^k \lfloor \tfrac{k}{2} - 1 \rfloor! \times m^{k-2} && \text{(iii)} \end{aligned}$$

Proof: Let v_i^1 and v_i^2 be distinct vertices in V_i, for all $i = 1, 2, \dots, k$. Define the following weights, where M is a number larger than n:
$c(v_i^1, v_{i+1}^2) = 0$, for all $i = 1, 2, \dots, k$ ($v_{k+1}^2 = v_1^2$, by definition).
$c(v_i^2, v) = 2M$, for all $v \in V_{i+1} - \{v_{i+1}^2\}$ and $i = 1, 2, \dots, k$ ($V_{k+1} = V_1$).
The weight of every arc in $C(V_1, \dots, V_k, V_1)$ not considered above is M.
All arcs outside of $C(V_1, \dots, V_k, V_1)$ have weight one.

We will now show the inequality (i). Observe that minFST finds the tour $H = 12\dots(k-1)k1$ in K_k^* on Step 2, as all the arcs in this tour have weight zero. The k-cycle found on Step 3 must have weight at least kM, as if the k-cycle uses an arc of weight less than M (namely, zero), then, before it uses another arc of weight 0, it must use an arc of weight $2M$. In fact, minFST finds a k-cycle of weight exactly kM ($v_1^1v_2^1v_3^1\dots v_k^1v_1^1$ has weight kM). This means that $dom(\texttt{minFST}) \leq f(k)$ is the number of k-cycles in D with weight at least kM.

Observe that every k-cycle Z of D of weight at least kM must have at least as many arcs of weight $2M$ as Z has arcs of weight 1. Indeed, let q' and q'' be the number of arcs of weight 1 and $2M$ respectively in Z. Then $kM \leq c(Z) \leq q' + 2Mq'' + (k - q' - q'')M$. Thus, $q'' \geq q' - q'/M$. Since q' and q'' are both integral, we may conclude that $q'' \geq q'$. Thus, $dom(\texttt{minFST}) \leq g(k)$, where $g(k)$ is the number of k-cycles in D in which the number of arcs of weight 1 does not exceed the number of arcs of weight $2M$.

Observe that $g(k) = \sum_{i=0}^{\lfloor k/2 \rfloor} |G(k,i)|$, where $G(k,i)$ is the set of k-cycles in D in which the number of arcs of weight 1 equals i and the number of arcs of weight $2M$ is at least i. The following construction generates all cycles in $G(k,i)$.

Let W_1 be any set of i arcs in $H = 12\dots(k-1)k1$ such that no two arcs share a vertex and let W_2 be any set of i arcs in H, which are disjoint from W_1. Replace the arcs of W_2 in H by a collection R_2 of $|W_2|$ arcs such that $R_2 \cap W_2 = \emptyset$ and $(A(H) - W_2) \cup R_2$ form a Hamilton cycle R in K_k^*. To construct a k-cycle $Z \in G(k,i)$, replace every arc ij of R by any arc v_iv_j of D from V_i to V_j such that every arc $ij \in W_1$ is replaced by an arc of weight $2M$ and all the arcs v_iv_j form a k-cycle. It is not difficult to see that every cycle in $G(k,i)$ can be built by this construction (just reverse the construction).

There are m^k k-cycles in D visiting the partite sets in the order given by R, and at most m^{k-i} of these use an arc of weight $2M$ between every two partite sets which have an arc from W_1 between them (since to have such arcs only vertices $v_i^2 \in V_i$ can be in the corresponding places of the cycle). By Theorem 17 in [11], W_1 can be chosen in $\binom{k-i}{i} + \binom{k-i-1}{i-1}$ different ways. Observe that W_2 can clearly be chosen in $\binom{k-i}{i}$ ways, while R_2 can be picked in z_i ways. So $\left[\binom{k-i}{i}\left(\binom{k-i}{i} + \binom{k-i-1}{i-1}\right) z_i m^{k-i}\right]$ is an upper bound on $|G(k,i)|$ and, thus, on the number of k-cycles in D with exactly i arcs of weight 1, and total weight greater than or equal to kM. When $i = 0$, the number of such cycles is m^k, and when $i = 1$ or $i = 2$, there are no such cycles. Therefore we obtain the inequality (i).

We will now show the inequalities (ii) and (iii). Firstly observe that $\binom{k-i}{i} \geq \binom{k-i-1}{i-1}$, when $i \leq k/2$. Secondly observe that $z_i \leq (i-1)!$ and $\left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{3+\sqrt{5}}{2}$. Now the fact that $m \geq 2$ and Lemma 6.5 imply the following:

$$\begin{aligned}
dom(\texttt{minFST}) &\leq m^k + \sum_{i=3}^{\lfloor k/2 \rfloor} \left[\binom{k-i}{i}\left(\binom{k-i}{i} + \binom{k-i-1}{i-1}\right) z_i m^{k-i}\right] \\
&\leq m^k + \sum_{i=3}^{\lfloor k/2 \rfloor} \left[\binom{k-i}{i}\left(2\binom{k-i}{i}\right)(i-1)! \times m^{k-i}\right] \\
&\leq m^k + \max_{3 \leq i \leq \lfloor k/2 \rfloor}\{2\binom{k-i}{i}^2 (i-1)!\} \times \sum_{i=3}^{\lfloor k/2 \rfloor} m^{k-i} \\
&\leq m^k + \max_{3 \leq i \leq \lfloor k/2 \rfloor}\{2\binom{k-i}{i}^2 (i-1)!\} \times m^{k-2} \\
&\leq m^k + \frac{6}{\pi k}\left(\frac{3+\sqrt{5}}{2}\right)^k \lfloor \tfrac{k}{2} - 1 \rfloor! \times m^{k-2}
\end{aligned}$$

□

Theorem 6.7 *For every $m \geq 2$ and $k \geq 4$, we have $dom(\texttt{minFST}) < dom(\texttt{averFST})$.*

Proof: By Theorem 6.4, we have $dom(\texttt{averFST}) \geq m^k + ((k-2)! - 1)m^{k-2}$. It is not difficult to show that when $k \geq 12$, $\frac{6}{\pi k}\left(\frac{3+\sqrt{5}}{2}\right)^k \times \lfloor \frac{k}{2} - 1 \rfloor!$ grows slower than than $(k-2)! - 1$. It is easy to check by computer that the latter is larger than the former for $k = 12$ and the latter is larger than the coefficient of m^{k-2} in (ii) of Theorem 6.6. □

References

[1] D. Applegate, R.E. Bixby, V. Chvátal W., Cook, On the Solution of Traveling Salesman Problems, Doc. Math., J. DMV, Extra Vol. ICM Berlin 1998, III (1998) 645–656. The `Concorde` code is currently available from `http://www.math.princeton.edu/tsp/concorde.html`

[2] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo, A. Zverovitch, Process Planning for Rotational Parts and the Generalized Traveling Salesman Problem, Int. J. Production Res., to appear.

[3] V. Dimitrijević, Z. Sarić, An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs, Information Sciences 102 (1997) 105–110.

[4] M. Fischetti, J.J. Salazar, P. Toth, Exact methods for the ATSP, in: G. Gutin, A.P. Punnen (Eds.), Traveling Salesman Problem and Its Variations, Kluwer Academic Publishrers, Dordrecht, 2002.

[5] M. Fischetti, J.J. Salazar, P. Toth, The symmetric generalized traveling salesman polytope, Networks 26 (1995) 113–123.

[6] M. Fischetti, J.J. Salazar, P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, Oper. Res. 45 (1997) 378–394.

[7] M. Fischetti, J.J. Salazar, P. Toth, The generalized traveling salesman and orienteering problem, in: G. Gutin, A.P. Punnen (Eds.), Traveling Salesman Problem and Its Variations, Kluwer Academic Publishrers, Dordrecht, 2002.

[8] F. Glover, G. Gutin, A. Yeo, A. Zverovich, Construction heuristics for the asymmetric TSP, Europ. J. Oper. Res. 129 (2001) 555–568.

[9] F. Glover, A.P. Punnen, The travelling salesman problem: new solvable cases and linkages with the development of approximation algorithms, J. Oper. Res. Soc. 48 (1997) 502–510.

[10] G. Gutin, A.P. Punnen (Eds.), Traveling Salesman Problem and Its Variations, Kluwer Academic Publishrers, Dordrecht, 2002.

[11] G. Gutin, A. Yeo, A. Zverovitch, Exponential Neighborhoods and Domination Analysis for the TSP, in: G. Gutin, A.P. Punnen (Eds.), Traveling Salesman Problem and Its Variations, Kluwer Academic Publishrers, Dordrecht, 2002.

[12] G. Laporte, A. Asef-Vaziri, C. Sriskandarajah, Some Applications of the Generalized Traveling Salesman Problem, J. Oper. Res. Soc. 47 (1996) 1461–1467.

[13] G. Laporte, H. Mercure, Y. Nobert, Generalized Travelling Salesman Problem through n sets of nodes: the asymmetrical cases, Discrete Appl. Math. 18 (1987) 185–197.

[14] G. Laporte, Y. Nobert, Generalized Travelling Salesman through n sets of nodes: an integer programming approach, INFOR 21 (1983) 61–75.

[15] G. Laporte, F. Semet, Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem, INFOR (37) 1999 114–120.

[16] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, D.B. Shmoys (Eds.), Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, Chichester, 1985.

[17] Y. Lien, E. Ma, B.W.S. Wah, Transformation of the generalized traveling salesman problem into the standard traveling salesman problem, Information Sciences 74 (1993) 177–189.

[18] C.E. Noon, The Generalized Traveling Salesman Problem, PhD thesis, Department of Industrial and Operations Research, University of Tennessee, 1988.

[19] C.E. Noon, J.C. Bean, A Lagrangian based approach for the asymmetric generalized traveling salesman problem, Oper. Res. 39 (1991) 623–632.

[20] C.E. Noon, J.C. Bean, An efficient transformation of the generalized traveling salesman problem, INFOR 31 (1993) 39–44.

[21] A.P. Punnen, S. Kabadi, Domination analysis of some heuristics for the traveling salesman problem, Discrete Appl. Math. 119 (2002) 117–128.

[22] A.P. Punnen, F. Margot, S. Kabadi, TSP heuristics: domination analysis and complexity, Algorithmica, to appear.

[23] G. Reinelt, TSPLIB — A Traveling Salesman Problem Library, ORSA J. Comput. 3 (1991) 376–384, `http://www.crpc.rice.edu/softlib/tsplib/`.

[24] J.J. Salazar, Algoritmi per il problema del Commesso Viaggiatore Generalizzato, PhD thesis, Royal Spanish College in Bologna, 1992 (in Italian).

[25] M.M. Sepehri, The symmetric generalized travelling salesman problem, PhD thesis, University of Tennessee, Knoxville, 1991.