Domination Analysis of Combinatorial Optimization Algorithms and Problems

Gregory Gutin[*] Anders Yeo[†]

Abstract

We provide an overview of an emerging area of domination analysis (DA) of combinatorial optimization algorithms and problems. We consider DA theory and its relevance to computational practice.

1 Introduction

In the recently published book [19], Chapter 6 is partially devoted to domination analysis (DA) of the Traveling Salesman Problem (TSP) and its heuristics. The aim of this chapter is to provide an overview of the whole area of DA. In particular, we describe results that significantly generalize the corresponding results for the TSP.

To make reading of this chapter more active, we provide questions that range from simple to relatively difficult ones. Also, we add research questions that supply the interested reader with open and challenging problems.

This chapter is organized as follows. In Subsection 1.1 of this section we motivate the use of DA in combinatorial optimization. We provide a short introduction to DA in Subsection 1.2. We conclude this section by giving additional terminology and notation.

One of the goals of DA is to analyze the domination number or domination ratio of various algorithms. Domination number (ratio) of a heuristic H for a combinatorial optimization problem P is the maximum number (fraction) of all solutions that are not better than the solution found by H for any instance of P of size n. In Section 2 we consider TSP heuristics of large domination number. In Subsection 2.1 we provide a theorem that allows one

[*]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, gutin@cs.rhul.ac.uk

[†]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, anders@cs.rhul.ac.uk

to prove that a certain Asymmetric TSP heuristic is of very large domination number. We also provide an application of the theorem. In Subsection 2.2 we show how DA can be used in analysis of local search heuristics. Upper bounds for the domination numbers of Asymmetric TSP heuristics are derived in Subsection 2.3.

Section 3 is devoted to DA for other optimization problems. We demonstrate that problems such as the Minimum Partition, Max Cut, Max k-SAT and Fixed Span Frequency Assignment admit polynomial time algorithms of large domination number. On the other hand, we prove that some other problems including the Maximum Clique and the Minimum Vertex Cover do not admit algorithms of relatively large domination ratio unless P=NP.

Section 4 shows that, in the worst case, the greedy algorithm obtains the unique worst possible solution for a wide family of combinatorial optimization problems and, thus, in the worst case, the greedy algorithm is no better than the random choice for such problems. We conclude the chapter by a short discussion of DA practicality.

1.1 Why domination analysis ?

Exact algorithms allow one to find optimal solutions to NP-hard combinatorial optimization (CO) problems. Many research papers report on solving large instances of some NP-hard problems (see, e.g., Chapters 2 and 4 in [19]). The running time of exact algorithms is often very high for large instances, and very large instances remain beyond the capabilities of exact algorithms.

Even for instances of moderate size, if we wish to remain within seconds or minutes rather than hours or days of running time, only heuristics can be used. Certainly, with heuristics, we are not guaranteed to find optimum, but good heuristics normally produce near-optimal solutions. This is enough in most applications since very often the data and/or mathematical model are not exact anyway.

Research on CO heuristics has produced a large variety of heuristics especially for well-known CO problems. Thus, we need to choose the best ones among them. In most of the literature, heuristics are compared in computational experiments. While experimental analysis is of definite importance, it cannot cover all possible families of instances of the CO problem at hand and, in particular, it normally does not cover the hardest instances.

Approximation Analysis [4] is a frequently used tool for theoretical evaluation of CO heuristics. Let $\mathcal{H}$ be a heuristic for a combinatorial minimization problem P and let $\mathcal{I}_n$ be the set of instances of P of size n. In approximation analysis, we use the performance ratio $r_{\mathcal{H}}(n) = \max\{f(I)/f^*(I) : \ I \in \mathcal{I}_n\}$, where $f(I)$ $(f^*(I))$ is the value of the heuristic (optimal) solution of I. Unfortunately, for many CO problems, estimates for $r_{\mathcal{H}}(n)$ are not constants and provide only a vague picture of the quality of heuristics.

Domination Analysis (DA) provides an alternative and a complement to approximation analysis. In DA, we are interested in the domination number or domination ratio of heuristics (these parameters have been defined earlier). In many cases, DA is very useful. For example, we will see in Section 4 that the greedy algorithm has domination number 1 for many CO problems. In other words, the greedy algorithm, in the worst case, produces the unique worst possible solution. This is in line with latest computational experiments with the greedy algorithm, see, e.g., [28], where the authors came to the conclusion that the greedy algorithm 'might be said to self-destruct' and that it should not be used even as 'a general-purpose starting tour generator'.

The *Asymmetric Traveling Salesman Problem (ATSP)* is the problem of computing a minimum weight tour (Hamilton cycle) passing through every vertex in a weighted complete digraph on n vertices. See Figure 1. The *Symmetric TSP (STSP)* is the same problem, but on a complete undirected graph. When a certain fact holds for both ATSP and STSP, we will simply speak of *TSP*. Sometimes, the maximizing version of TSP is of interest, we denote it by *max TSP*.

APX is the class of CO problems that admit polynomial time approximation algorithms with a constant performance ratio [4]. It is well known that while max TSP belongs to APX, TSP does not. This is at odds with the simple fact that a 'good' approximation algorithm for max TSP can be easily transformed into an algorithm for TSP. Thus, it seems that both max TSP and TSP should be in the same class of CO problems. The above asymmetry was already viewed as a drawback of performance ratio already in the 1970's, see, e.g., [11, 30, 40]. Notice that from the DA point view max TSP and TSP are equivalent problems.

Zemel [40] was the first to characterize measures of quality of approximate solutions (of binary integer programming problems) that satisfy a few basic and natural properties: the measure becomes smaller for better solutions, it equals 0 for optimal solutions and it is the same for corresponding solutions of equivalent instances. While the performance ratio and even the relative error (see [4]) do not satisfy the last property, the parameter $1 - r$, where r is the domination ratio, does satisfy all of the properties.

Local Search (LS) is one of the most successful approaches in constructing heuristics for CO problems. Recently, several researchers started inves-

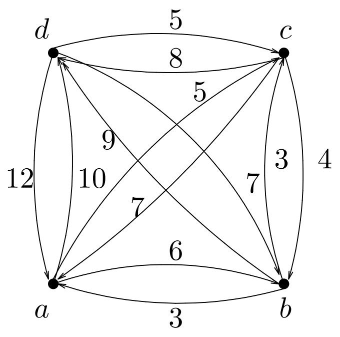

Figure 1: A complete weighted digraph

tigation of LS with Very Large Scale Neighbourhoods (see, e.g., [1, 12, 26]). The hypothesis behind this approach is that the larger the neighbourhood the better quality solution are expected to be found [1]. However, some computational experiments do not support this hypothesis, see, e.g., [15], where an LS with small neighbourhoods proves to be superior to that with large neighbourhoods. This means that some other parameters are responsible for the relative power of a neighbourhood. Theoretical and experimental results on TSP indicate that one such parameter may well be the domination ratio of the corresponding LS.

Sometimes, Approximation Analysis cannot be naturally used. Indeed, a large class of CO problems are multicriteria problems [14], which have several objective functions. (For example, consider STSP in which edges are assigned both time and cost, and one is required to minimize both time and cost.) We say that one solution s' of a multicriteria problems dominates another one s'' if the values of all objective functions at s' are not worse than those at s'' or the value of at least one objective function at s' is better than the value of the same objective function at s''. This definition allows us to naturally introduce the domination ratio (number) for multicriteria optimization heuristics. In particular, an algorithm that always finds a Pareto solution is of domination ratio 1.

In our view, it is advantageous to have bounds for both performance ratio and domination number (or, domination ratio) of a heuristic whenever it is possible. Roughly speaking this will enable us to see a 2D rather than 1D picture. For example, consider the double minimum spanning tree heuristic (DMST) for the Metric STSP (i.e., STSP with triangle inequality). DMST starts from constructing a minimum weight spanning tree T in the complete graph of the STSP, doubles every edge in T, finds a closed Euler trail E in the 'double' T, and cancels any repetition of vertices in E to obtain a TSP tour H. It is well-known and easy to prove that the weight of H is at most twice the weight of the optimal tour. Thus, the performance ratio for DMST is bounded by 2. However, Punnen, Margot and Kabadi [34] proved that the domination number of DMST is 1.

1.2 Introduction to domination analysis

Domination Analysis was formally introduced by Glover and Punnen [16] in 1997. Interestingly, important results on domination analysis for the TSP can be traced back to the 1970s, see Rublineckii [36] and Sarvanov [37].

Let $\mathcal{P}$ be a CO problem and let $\mathcal{H}$ be a heuristic for $\mathcal{P}$. The *domination number* $\mathrm{domn}(\mathcal{H},\mathcal{I})$ of $\mathcal{H}$ *for a particular instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of feasible solutions of $\mathcal{I}$ that are not better than the solution s produced by $\mathcal{H}$ including s itself. For example, consider an instance $\mathcal{T}$ of the STSP on 5 vertices. Suppose that the weights of tours in $\mathcal{T}$ are 3,3,5,6,6,9,9,11,11,12,14,15 (every instance of STSP on 5 vertices has 12 tours) and suppose that the greedy algorithm computes the tour T of weight 6. Then $\mathrm{domn}(\texttt{greedy},\mathcal{T}) = 9$. In general, if $\mathrm{domn}(\mathcal{H},\mathcal{I})$ equals the number of feasible solutions in $\mathcal{I}$, then $\mathcal{H}$ finds an optimal solution for $\mathcal{I}$. If $\mathrm{domn}(\mathcal{H},\mathcal{I}) = 1$, then the solution found by $\mathcal{H}$ for $\mathcal{I}$ is the unique worst possible one.

The *domination number* $\mathrm{domn}(\mathcal{H}, n)$ of $\mathcal{H}$ is the minimum of $\mathrm{domn}(\mathcal{H},\mathcal{I})$ over all instances $\mathcal{I}$ of size n. Since the ATSP on n vertices has $(n-1)!$ tours, an algorithm for the ATSP with domination number $(n-1)!$ is exact. The domination number of an exact algorithm for the STSP is $(n-1)!/2$. If an ATSP heuristic $\mathcal{A}$ has domination number equal 1, then there is an assignment of weights to the arcs of each complete digraph K_n^*, $n \geq 2$, such that $\mathcal{A}$ finds the unique worst possible tour in K_n^*.

When the number of feasible solutions depends not only on the size of the instance of the CO problem at hand (for example, the number of independent sets of vertices in a graph G on n vertices depends on the structure of G), the domination ratio of an algorithm $\mathcal{A}$ is of interest: the *domination ratio* of $\mathcal{A}$, $\mathrm{domr}(\mathcal{A}, n)$, is the minimum of $\mathrm{domn}(\mathcal{A},\mathcal{I})/\mathrm{sol}(\mathcal{I})$, where $\mathrm{sol}(\mathcal{I})$ is the number of feasible solutions of $\mathcal{I}$, taken over all instances $\mathcal{I}$ of size n. Clearly, domination ratio belongs to the interval $(0, 1]$ and exact algorithms are of domination ratio 1.

The *Minimum Partition Problem* (MPP) is the following CO problem: given n nonnegative numbers $a_1, a_2, \ldots, a_n$, find a bipartition of the set

$\{1, 2, \ldots, n\}$ into sets X and Y such that $d(X, Y) = |\sum_{i \in X} a_i - \sum_{i \in Y} a_i|$ is minimum. For simplicity, we assume that solutions X, Y and X', Y' are different as long as $X \neq X'$, i.e. even if $X = Y'$ (no symmetry is taken into consideration). Thus, the MPP has 2^n solutions.

Consider the following greedy-type algorithm $\mathcal{G}$ for the MPP: $\mathcal{G}$ sorts the numbers such that $a_{\pi(1)} \geq a_{\pi(2)} \geq \cdots \geq a_{\pi(n)}$, initiates $X = \{\pi(1)\}$, $Y = \{\pi(2)\}$, and, for each $j \geq 3$, puts $\pi(j)$ into X if $\sum_{i \in X} a_i \leq \sum_{i \in Y} a_i$, and into Y, otherwise. It is easy to see that any solution X, Y produced by $\mathcal{G}$ satisfies $d(X, Y) \leq a_{\pi(1)}$.

Consider any solution X', Y' of the MPP for the input $\{a_1, a_2, \ldots, a_n\} - \{a_{\pi(1)}\}$. If we add $a_{\pi(1)}$ to Y' if $\sum_{i \in X'} a_i \leq \sum_{i \in Y'} a_i$ and to X', otherwise, then we obtain a solution X'', Y'' for the original problem with $d(X'', Y'') \geq d(X, Y)$. Thus, the domination number of $\mathcal{G}$ is at least 2^{n-1} and we have the following:

Proposition 1.1 *The domination ratio of $\mathcal{G}$ is at least 0.5.*

In fact, a slight modification of $\mathcal{G}$ is of domination ratio very close to 1, see Section 3.

Let us consider another CO problem. In the *Assignment Problem* (AP), we are given a complete bipartite graph B with n vertices in each partite set and a non-negative weight $\mathrm{wt}(e)$ assigned to each edge e of B. We are required to find a perfect matching (i.e., a collection of n edges with no common vertices) in B of minimum total weight.

The AP can be solved to optimality in time $O(n^3)$ by the Hungarian algorithm. Thus, the domination number of the Hungarian algorithm equals $n!$, the total number of perfect matchings in B.

For some instances of the AP, the $O(n^3)$ time may be too high and thus we may be interested in having a faster heuristic for the AP. Perhaps, the first heuristics that comes into mind is the greedy algorithm (`greedy`). The greedy algorithm starts from the empty matching X and, at each iteration, it appends to X the cheapest edge of B that has no common vertices with edges already in X. (A description of `greedy` for a much more general combinatorial optimization problem is provided in Section 4.)

The proof of the following theorem shows that the greedy algorithm fails on many 'non-exotic' instances of the AP.

Theorem 1.2 *For the AP,* `greedy` *has domination number 1.*

Proof: Let B be a complete bipartite graph with n vertices in each partite set and let $u_1, u_2, ..., u_n$ and $v_1, v_2, ..., v_n$ be the two partite sets of B. Let

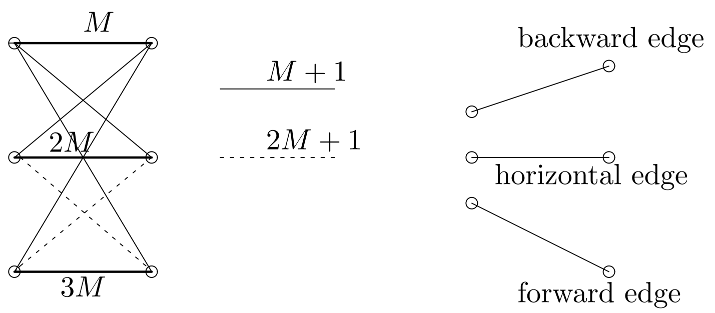

Figure 2: Assignment of weights for $n = 3$; classification of edges

M be any number greater than n. We assign weight $i \times M$ to the edge u_iv_i for $i = 1, 2, ..., n$ and weight $\min\{i, j\} \times M + 1$ to every edge u_iv_j, $i \neq j$; see Figure 2 for illustration in the case $n = 3$.

We classify edges of B as follows: u_iv_j is horizontal (forward, backward) if $i = j$ ($i < j$, $i > j$). See Figure 2.

The greedy algorithm will choose edges $u_1v_1, u_2v_2, ..., u_nv_n$ (and in that order). We denote this perfect matching P and we will prove that P is the unique most expensive perfect matching in B. The weight of P is $\mathrm{wt}(P) = M + 2M + ... + nM$.

Choose an arbitrary perfect matching P' in B distinct from P. Let P' have edges $u_1v_{p_1}, u_2v_{p_2}, ..., u_nv_{p_n}$. By the definition of the costs in B, $\mathrm{wt}(u_iv_{p_i}) \leq M \times i + 1$. Since P' is distinct from P, it must have edges that are not horizontal. This means it has backward edges. If $u_kv_{p_k}$ is a backward edge, i.e. $p_k < k$, then $\mathrm{wt}(u_kv_{p_k}) \leq M(k-1)+1 = (Mk+1)-M$. Hence,

$$\mathrm{wt}(P') \leq (M + 2M + ... + nM) + n - M = \mathrm{wt}(P) + n - M.$$

Thus, $\mathrm{wt}(P') < \mathrm{wt}(P)$. □

Question 1.3 *Formulate the greedy algorithm for the ATSP.*

Question 1.4 *Consider the following mapping f from the arc set of K_n^* into the edge set of $K_{n,n}$, the complete bipartite graph on $2n$ vertices. Let $x_1, \ldots, x_n$ be vertices of K_n^*, and let $\{u_1, \ldots, u_n\}$ and $\{v_1, \ldots, v_n\}$ be partite sets of $K_{n,n}$. Then $f(x_ix_j) = u_iv_{j-1}$ for each $1 \leq i \neq j \leq n$, where $v_0 = v_n$. Show that f maps every Hamilton cycle of K_n^* into a matching of $K_{n,n}$.*

Question 1.5 *Using the mapping f of Question 1.4 and Theorem 1.2, prove that the greedy algorithm has domination number 1 for the ATSP.*

1.3 Additional terminology and notation

Following the terminology in [20], a CO problem $\mathcal{P}$ is called *$\mathcal{DOM}$-easy* if there exists a polynomial time algorithm, $\mathcal{A}$, such that $domr(\mathcal{A}, n) \geq 1/p(n)$, where $p(n)$ is a polynomial in n. In other words, a problem is $\mathcal{DOM}$-easy, if, in polynomial time, we can always find a solution, with domination number at least a polynomial fraction of all solution. If no such algorithm exists, $\mathcal{P}$ is called *$\mathcal{DOM}$-hard*.

For a digraph D, $V(D)$ $(A(D))$ denotes the vertex (arc) set of H. The same notation are used for paths and cycles in digraphs. A *tour* in a digraph D is a Hamilton cycle in D. A *complete digraph* K_n^* is a digraph in which every pair x, y of distinct vertices is connected by the pair $(x, y), (y, x)$ of arcs. The *out-degree* $d^+(v)$ (*in-degree* $d^-(v)$) of a vertex v of a digraph D is the number of arcs leaving v (entering v). It is clear that $|A(D)| = \sum_{v \in V(D)} d^+(v) = \sum_{v \in V(D)} d^-(v)$.

We will often consider weighted digraphs, i.e., pairs (D, wt), where wt is a mapping from $A(D)$ into the set of reals. For an arc $a = (x, y)$ in (K_n^*, wt), the *contraction* of a results in a complete digraph with vertex set $V' = V(K_n^*) \cup \{v_a\} - \{x, y\}$ and weight function wt', where $v_a \notin V({K^*}_n)$, such that the weight $\mathrm{wt}'(u, w)$, for $u, w \in V'$, equals $\mathrm{wt}(u, x)$ if $w = v_a$, $\mathrm{wt}(y, w)$ if $u = v_a$, and $\mathrm{wt}(u, w)$, otherwise. The above definition has an obvious extension to a set of arcs; for more details, see [6].

For an undirected graph G, $V(G)$ $(E(G))$ denotes the vertex (edge) set of H. A *tour* in a graph G is a Hamilton cycle in G. A complete graph K_n is a graph in which every pair x, y of distinct vertices is connected by edge xy. Weighted graphs have weights assigned to their edges.

For a pair of given functions $f(k), g(k)$ of a non-negative integer argument k, we say that $f(k) = O(g(k))$ if there exist positive constants c and k_0 such that $0 \leq f(k) \leq cg(k)$ for all $k \geq k_0$. If there exist positive constants c and k_0 such that $0 \leq cf(k) \leq g(k)$ for all $k \geq k_0$, we say that $g(k) = \Omega(f(k))$. Clearly, $f(k) = O(g(k))$ if and only if $g(k) = \Omega(f(k))$. If both $f(k) = O(g(k))$ and $f(k) = \Omega(g(k))$ hold, then we say that $f(k)$ and $g(k)$ are of the same order and denote it by $f(k) = \Theta(g(k))$.

2 TSP heuristics with large domination number

Since there is a recent survey on domination analysis of TSP heuristics [19], we restrict ourselves to giving a short overview of three important topics. All results will be formulated specifically for the ATSP or the STSP, but in many cases similar results hold for the symmetric or asymmetric counterparts as well.

2.1 ATSP heuristics of domination number at least $\Omega((n-2)!)$

We will show how the domination number of an ATSP heuristic can be related to the average value of a tour. This result was (up till now) used in all proofs that a heuristic has domination number at least $\Omega((n-2)!)$. Examples of such heuristics are the greedy expectation algorithm introduced in [21], vertex insertion algorithms and k-opt (see [19]). Using the above-mentioned result we will prove that vertex insertion algorithms have domination number at least $\Omega((n-2)!)$.

A *decomposition* of $A(K_n^*)$ into tours, is a collection of tours in K_n^*, such that every arc in K_n^* belongs to exactly one of the tours. The following lemma was proved for odd n by Kirkman (see [9], p. 187), and the even case result was established in [39].

Lemma 2.1 *For every $n \geq 2$, $n \neq 4$, $n \neq 6$, there exists a decomposition of $A(K_n^*)$ into tours.*

An *automorphism*, f, of $V(K_n^*)$ is a bijection from $V(K_n^*)$ to itself. Note that if C is a tour in K_n^* then $f(C)$ is also a tour K_n^*.

We define $\tau(K_n^*)$ as the average weight of a tour in K_n^*. As there are $(n-1)!$ tours in K_n^*, and $(n-2)!$ tours in K_n^*, which use a given arc e (see Question 2.2), we note that

$$\tau(K_n^*) = \frac{1}{(n-1)!} \sum_{e \in A(K_n^*)} \mathrm{wt}(e) \times (n-2)!,$$

which implies that $\tau(K_n^*) = \mathrm{wt}(K_n^*)/(n-1)$, where $\mathrm{wt}(K_n^*)$ is the sum of all weights in K_n^*.

Question 2.2 *Let $e \in A(K_n^*)$ be arbitrary. Show that there are $(n-2)!$ tours in K_n^*, which use the arc e.*

Question 2.3 *Let $D = \{C_1, \ldots, C_{n-1}\}$ be a decomposition of $A(K_n^*)$ into tours. Assume that C_{n-1} is the tour in D of maximum weight. Show that* $\mathrm{wt}(C_{n-1}) \geq \tau(K_n^*)$.

Question 2.4 *Let $D = \{C_1, \ldots, C_{n-1}\}$ be a decomposition of $A(K_n^*)$ into tours. Let α be an automorphism of $V(K_n^*)$. Prove that α maps D into a decomposition of $A(K_n^*)$ into tours.*

We are now ready to prove the main result of this section.

Theorem 2.5 *Assume that H is a tour in K_n^* such that $\mathrm{wt}(H) \le \tau(K_n^*)$. If $n \ne 6$, then H is not worse than at least $(n-2)!$ tours in K_n^*.*

Proof: The result is clearly true for $n = 2, 3$. If $n = 4$, the result follows from the fact that the most expensive tour, R, in K_n^* has weight at least $\tau(K_n^*) \ge \mathrm{wt}(H)$. So the domination number of H is at least two (H and R are two tours of weight at least $\mathrm{wt}(H)$).

Assume that $n \ge 5$ and $n \ne 6$. Let $V(K_n^*) = \{x_1, x_2, \dots, x_n\}$. By Lemma 2.1 there exists a decomposition, $D = \{C_1, ..., C_{n-1}\}$ of $A(K_n^*)$ into tours. Furthermore there are $(n-1)!$ automorphisms, $\{\alpha_1, \alpha_2, \dots, \alpha_{(n-1)!}\}$, of $V(K_n^*)$, which map vertex x_1 into x_1. Now let D_i be the decomposition of $A(K_n^*)$ into tours, obtained by using α_i on D. In other words, $D_i = \{\alpha_i(C_1), \alpha_i(C_2), \dots, \alpha_i(C_{n-1})\}$ (see Question 2.4).

Note that if R is a tour in K_n^*, then R belongs to exactly $(n-1)$ decompositions in $\{D_1, D_2, \dots, D_{(n-1)!}\}$, as one automorphism will map C_1 into R, another one will map C_2 into R, etc. Therefore R will lie in exactly the $(n-1)$ decompositions which we obtain from these $(n-1)$ automorphisms.

Now let E_i be the most expensive tour in D_i. By Question 2.3 we see that $\mathrm{wt}(E_i) \ge \tau(K_n^*)$. As any tour in the set $\mathcal{E} = \{E_1, E_2, \dots, E_{(n-1)!}\}$ appears at most $(n-1)$ times, the set $\mathcal{E}$ has at least $(n-2)!$ distinct tours, which all have weight at least $\tau(K_n^*)$. As $\mathrm{wt}(H) \le \tau(K_n^*)$, this proves the theorem. □

The above result has been applied to prove that a wide variety of ATSP heuristics have domination number at least $\Omega((n-2)!)$. Below we will show how the above result can be used to prove that ATSP vertex insertion algorithms have domination number at least $(n-2)!$.

Let (K_n^*, wt) be an instance of the ATSP. Order the vertices $x_1, x_2, \dots, x_n$ of K_n^* using some rule. The generic vertex insertion algorithm proceeds as follows. Start with the cycle $C_2 = x_1x_2x_1$. Construct the cycle C_j from C_{j-1} ($j = 3, 4, 5, \dots, n$), by inserting the vertex x_j into C_{j-1} at the optimum place. This means that for each arc $e = xy$ which lies on the cycle C_{j-1} we compute $\mathrm{wt}(xx_j) + \mathrm{wt}(x_jy) - \mathrm{wt}(xy)$, and insert x_j into the arc $e = xy$, which obtains the minimum such value. We note that $\mathrm{wt}(C_j) = \mathrm{wt}(C_{j-1}) + \mathrm{wt}(xx_j) + \mathrm{wt}(x_jy) - \mathrm{wt}(xy)$.

Theorem 2.6 *The generic vertex insertion algorithm has domination number at least $(n-2)!$.*

Proof: We will prove that the generic vertex insertion algorithm produces a tour of weight at most $\tau(K_n^*)$ by induction. Clearly this is true for $n = 2$, as there is only one tour in this case. Now assume that it is true for K_{n-1}^*. This implies that $\mathrm{wt}(C_{n-1}) \leq \tau(K_n^* - x_n)$. Without loss of generality assume that $C_{n-1} = x_1x_2 \ldots x_{n-1}x_1$. Let $\mathrm{wt}(X, Y) = \sum_{x\in X,\ y\in Y} c(xy)$ for any disjoint sets X and Y. Since C_n was chosen optimally we see that its weight is at most (where $x_0 = x_{n-1}$ in the sum)

$$
\begin{aligned}
&(\textstyle\sum_{i=0}^{n-2} \mathrm{wt}(C_{n-1}) + \mathrm{wt}(x_ix_n) + \mathrm{wt}(x_nx_{i+1}) - \mathrm{wt}(x_ix_{i+1}))/(n-1) \\
&= \mathrm{wt}(C_{n-1}) + (\mathrm{wt}(V - x_n, x_n) + \mathrm{wt}(x_n, V - x_n) - \mathrm{wt}(C_{n-1}))/(n-1) \\
&\leq ((n-2)\tau(K_n^* - x_n) + \mathrm{wt}(V - x_n, x_n) + \mathrm{wt}(x_n, V - x_n))/(n-1) \\
&= (\mathrm{wt}(K_n^* - x_n) + \mathrm{wt}(V - x_n, x_n) + \mathrm{wt}(x_n, V - x_n))/(n-1) \\
&= \mathrm{wt}(K_n^*)/(n-1) = \tau(K_n^*).
\end{aligned}
$$

This completes the induction proof. Theorem 2.5 now implies that the domination number of the generic vertex insertion algorithm is at least $(n-2)!$. □

2.2 Domination numbers of local search heuristics

In TSP local search (LS) heuristics, a neighborhood $N(T)$ is assigned to every tour T, a set of tours in some sense close to T. The *best improvement* LS proceeds as follows. We start from a tour T_0. In the i'th iteration ($i \geq 1$), we search in the neighborhood $N(T_{i-1})$ for the best tour T_i. If the weights of T_{i-1} and T_i do not coincide, we carry out the next iteration. Otherwise, we output T_i.

One of the first exponential size TSP neighborhoods (called `assign` in [12]) was considered independently by Sarvanov and Doroshko [38], and Gutin [17]. We describe this neighborhood and establish a simple upper bound on the domination number of the best improvement LS based on this neighborhood. We will see that the domination number of the best improvement LS based on `assign` is significantly smaller than that of the best improvement LS based on `2-opt`, a well-known STSP heuristic.

Consider a weighted K_n. Assume that $n = 2k$. Let $T = x_1y_1x_2y_2 \ldots x_ky_kx_1$ be an arbitrary tour in K_n. The neighborhood `assign`, $N_a(T)$, is defined as follows: $N_a(T) = \{x_1y_{\pi(1)}x_2y_{\pi(2)} \ldots x_ky_{\pi(k)}x_1 : (\pi(1), \pi(2), \ldots, \pi(k))$ is a permutation of $(1, 2, \ldots, k)\}$. Clearly, $N_a(T)$ contains $k!$ tours. We will show that we can find the tour of minimum weight in $N_a(T)$ in polynomial time.

Let B be a complete bipartite graph with partite sets $\{z_1, z_2, \ldots, z_n\}$ and $\{y_1, y_2, \ldots, y_n\}$, and let the weight of $z_i y_j$ be $\mathrm{wt}(x_i y_j) + \mathrm{wt}(y_j x_{i+1})$ (where $x_{n+1} = x_1$). Let M be a perfect matching in B, and assume that z_i is matched to $y_{m(i)}$ in M. Observe that the weight of M is equal to the weight of the tour $x_1 y_{m(1)} x_2 y_{m(2)} \ldots x_n y_{m(n)} x_1$. Since every tour in $N_a(T)$ corresponds to a perfect matching in B, and visa versa, a minimum weight perfect matching in B corresponds to a minimum weight tour in $N_a(T)$. Since we can find a minimum weight perfect matching in B in $O(n^3)$ time using the Hungarian method, we obtain the following theorem.

Theorem 2.7 *The best tour in $N_a(T)$ can be found in $O(n^3)$ time.*

While the size of $N_a(T)$ is quite large, the domination number of the best improvement LS based on `assign` is relatively small. Indeed, consider K_n with vertices $\{x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_k\}$. Suppose that the weights of all edges of the forms $x_i y_j$ and $y_i x_j$ equal 1 and the weights of all other edges equal 0. Then, starting from the tour $T = x_1 y_1 x_2 y_2 \ldots x_k y_k x_1$ of weight n the best improvement will output a tour of weight n, too. However, there are only $(k!)^2/(2k)$ tours of weight n in K_n and the weight of no tour in K_n exceeds n. We have obtained the following:

Proposition 2.8 *For STSP, the domination number of the best improvement LS based on* `assign` *is at most $(k!)^2/(2k)$, where $k = n/2$.*

The k-`opt`, $k \geq 2$, neighborhood of a tour T consists of all tour that can be obtained by deleting a collection of k edges (arcs) and adding another collection of k edges (arcs). It is easy to see that one iteration of the best improvement k-`opt` LS can be completed in time $O(n^k)$. Rublineckii [36] showed that every local optimum for the best improvement 2-opt and 3-opt for STSP is of weight at least the average weight of a tour and, thus, by an analog of Theorem 2.5 is of domination number at least $(n-2)!/2$ when n is even and $(n-2)!$ when n is odd. Observe that this result is of restricted interest since to reach a k-opt local optimum one may need exponential time (cf. Section 3 in [27]). However, Punnen, Margot and Kabadi [34] managed to prove the following result.

Theorem 2.9 *For the STSP the best improvement* `2-opt` *LS produces a tour, which is not worse than at least $\Omega((n-2)!)$ other tours, in at most $O(n^3\ logn)$ iterations.*

The last two assertions imply that after a polynomial number of iterations the best improvement `2-opt` LS has domination number at least $\Omega(2^n/n^{3.5})$ times larger than that of the best improvement `assign` LS.

Theorem 2.9 is also valid for the best improvement `3-opt` LS and some other LS heuristics for TSP, see [26, 34].

2.3 Upper bounds for domination numbers of ATSP heuristics

It is realistic to assume that any ATSP algorithm spends at least one unit of time on every arc of K_n^* that it considers. We use this assumption in the rest of this subsection.

Theorem 2.10 *[24, 22] Let $\mathcal{A}$ be an ATSP heuristic of complexity $t(n)$. Then the domination number of $\mathcal{A}$ does not exceed* $\max_{1\leq n'\leq n}(t(n)/n')^{n'}$.

Proof: Let $D = (K_n^*, \mathrm{wt})$ be an instance of the ATSP and let H be the tour that $\mathcal{A}$ returns, when its input is D. Let $DOM(H)$ denotes all tours in D which are not lighter than H including H itself. We assume that D is the worst instance for $\mathcal{A}$, namely $\mathrm{domn}(\mathcal{A}, n) = |DOM(H)|$. Since $\mathcal{A}$ is arbitrary, to prove this theorem, it suffices to show that $|DOM(H)| \leq \max_{1\leq n'\leq n}(t(n)/n')^{n'}$.

Let E denote the set of arcs in D, which $\mathcal{A}$ actually examines; observe that $|E| \leq t(n)$ by the assumption above. Let F be the set of arcs in H that are not examined by $\mathcal{A}$, and let G denote the set of arcs in $D - A(H)$ that are not examined by $\mathcal{A}$.

We first prove that every arc in F must belong to each tour of $DOM(H)$. Assume that there is a tour $H' \in DOM(H)$ that avoids an arc $a \in F$. If we assign to a a very large weight, H' becomes lighter than H, a contradiction.

Similarly, we prove that no arc in G can belong to a tour in $DOM(H)$. Assume that an $a \in G$ and a is in a tour $H' \in DOM(H)$. By making a very light, we can ensure that $\mathrm{wt}(H') < \mathrm{wt}(H)$, a contradiction.

Now let D' be the digraph obtained by contracting the arcs in F and deleting the arcs in G, and let n' be the number of vertices in D'. Note that every tour in $DOM(H)$ corresponds to a tour in D' and, thus, the number of tours in D' is an upper bound on $|DOM(H)|$. In a tour of D', there are at most $d^+(i)$ possibilities for the successor of a vertex i, where $d^+(i)$ is the out-degree of i in D'. Hence we obtain that

$$|DOM(H)| \le \prod_{i=1}^{n'} d^+(i) \le \left(\frac{1}{n'}\sum_{i=1}^{n'} d^+(i)\right)^{n'} \le \left(\frac{t(n)}{n'}\right)^{n'},$$

where we applied the arithmetic-geometric mean inequality. □

Corollary 2.11 *[24, 22] Let $\mathcal{A}$ be an ATSP heuristic of complexity $t(n)$. Then the domination number of $\mathcal{A}$ does not exceed* $\max\{e^{t(n)/e}, (t(n)/n)^n\}$*, where e is the basis of natural logarithms.*

Proof: Let $U(n) = \max_{1\le n'\le n}(t(n)/n')^{n'}$. By differentiating $f(n') = (t(n)/n')^{n'}$ with respect to n' we can readily obtain that $f(n')$ increases for $1 \le n' \le t(n)/e$, and decreases for $t(n)/e \le n' \le n$. Thus, if $n \le t(n)/e$, then $f(n')$ increases for every value of $n' < n$ and $U(n) = f(n) = (t(n)/n)^n$. On the other hand, if $n \ge t(n)/e$ then the maximum of $f(n')$ is for $n' = t(n)/e$ and, hence, $U(n) = e^{t(n)/e}$. □

The next assertion follows directly from the proof of Corollary 2.11.

Corollary 2.12 *[24, 22] Let $\mathcal{A}$ be an ATSP heuristic of complexity $t(n)$. For $t(n) \ge en$, the domination number of $\mathcal{A}$ does not exceed $(t(n)/n)^n$.*

Note that the restriction $t(n) \ge en$ is important since otherwise the bound of Corollary 2.12 can be invalid. Indeed, if $t(n)$ is a constant, then for n large enough the upper bound becomes smaller than 1, which is not correct since the domination number is always at least 1.

Question 2.13 *Fill in details in the proof of Corollary 2.11.*

Question 2.14 *Using Corollary 2.11 show that ATSP $O(n)$-time algorithms can have domination number at most $2^{\Theta(n)}$.*

Question 2.15 *Show that there exist ATSP $O(n)$-time algorithms of domination number at least $2^{\Omega(n)}$. Compare the results of the last two questions.*

We finish this subsection with the following:

Theorem 2.16 *[34] Unless P=NP, there is no polynomial time ATSP algorithm of domination number $(n-1)! - k$ for any constant k.*

Proof: Fix any m such that $k < m!$ and consider weighted K^*_{n-m}, $n > m$. Choose a pair u, v of distinct vertices in K^*_{n-m}. Let M be an arbitrary number larger than n times the maximum weight of an arc in K^*_{n-m}. Add m new vertices and necessary arcs to obtain K^*_n. The weight function wt' in K^*_n attains the same values on the arcs of K^*_{n-m} apart from the arc (u, v) for which $\mathrm{wt}'(u, v) = M$. Every arc xy between $K^*_{n-m} - \{u, v\}$ and the m new vertices has $\mathrm{wt}'(x, y) = M$. The rest of the arcs in K^*_n are of weight 0.

Let $\mathcal{A}$ be a polynomial time ATSP algorithm of domination number $(n-1)! - k$. Using $\mathcal{A}$ we can find a tour T in K^*_n, which is not worse than at least $(n-1)! - k - 1$ other tours. Let $\mathcal{T}$ denote the set of tours in K^*_n, each of which consists of the lightest Hamilton (u, v)-path in K^*_{n-m} and arcs of weight 0. Clearly, any tour in $\mathcal{T}$ is lighter than any tour not in $\mathcal{T}$. Since $|\mathcal{T}| = m! < k$, $T \in \mathcal{T}$. Thus, after removal of the new m vertices from T, we will get the lightest Hamilton (u, v)-path in K^*_{n-m}. However, it is well-known that the problem to find a lightest Hamilton path from u to v in a complete digraph, with fixed pair u, v of vertices, is NP-hard. We have arrived to a contradiction with the assumption that P$\neq$NP. $\square$

For a result that is stronger than Theorem 2.16, see [34].

3 Heuristics of large domination ratio for other CO problems

In this section, we consider other CO problems which have heuristics with relatively large domination ratios, as well as some CO problems which provably do not have heuristics with large domination ratios (unless P=NP). Even though most initial research on domination analysis has been done on TSP, there is now a wide variety of other problems, which have been studied in this respect.

3.1 Minimum Partition and Multiprocessor Scheduling

We already considered the Minimum Partition Problem (MPP) in Subsection 1.2, where we described a simple algorithm of domination ratio at least 0.5. In this subsection we introduce a slightly more complicated algorithm of domination ratio close to 1.

Let B_n be the set of all n-dimensional vectors $(\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ with $\{-1, 1\}$ coordinates. The MPP can be stated as follows: given n nonnegative numbers $\{a_1, a_2, \ldots, a_n\}$, find a vector $(\epsilon_1, \epsilon_2, \ldots, \epsilon_n) \in B_n$ such that $|\sum_{i=1}^n \epsilon_i a_i|$

is minimum.

Consider the following greedy-type algorithm $\mathcal{B}$. Initially sort the numbers such that $a_{\pi(1)} \geq a_{\pi(2)} \geq \cdots \geq a_{\pi(n)}$. Choose an integral constant $p > 0$ and fix $k = \lfloor p \log_2 n \rfloor$. Solve the MP to optimality for $a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(k)}$, i.e., find optimal values of $\epsilon_{\pi(1)}, \epsilon_{\pi(2)}, \ldots, \epsilon_{\pi(k)}$. (This can be trivially done in time $O(n^p)$.) Now for each $j > k$, if $\sum_{i=1}^{j-1} \epsilon_{\pi(i)} a_{\pi(i)} < 0$, then set $\epsilon_{\pi(j)} = +1$, and otherwise $\epsilon_{\pi(j)} = -1$.

Theorem 3.1 *[2] The domination ratio of $\mathcal{B}$ is at least* $1 - \binom{k}{\lfloor k/2 \rfloor}/2^k = 1 - \Theta(\frac{1}{\sqrt{k}})$.

To prove this theorem, without loss of generality, we may assume $a_1 \geq a_2 \geq \ldots \geq a_n$.

Observe that if $\min |\sum_{i=1}^{k} \epsilon_i a_i| \geq \sum_{i=k+1}^{n} a_i$, then $\mathcal{B}$ outputs an optimal solution. Otherwise, it can be easily proved by induction that the solution produced satisfies $|\sum_{i=1}^{n} \epsilon_i a_i| \leq a_{k+1}$. Thus, we may assume the last inequality.

Now it suffices to prove the following:

Proposition 3.2 *The number of vectors $(\epsilon_1, \ldots, \epsilon_n) \in B_n$ for which $|\sum_{i=1}^{n} \epsilon_i a_i| < a_{k+1}$ is at most $\binom{k}{\lfloor k/2 \rfloor} 2^{n-k}$.*

To prove this proposition, we will use the following lemma:

Lemma 3.3 *[13] Let $a_1 \geq a_2 \geq \cdots \geq a_k$ and let (a, b) be an arbitrary open interval such that $b - a \leq 2a_k$. Then the number of vectors $(\delta_1, \ldots, \delta_k) \in B_k$ such that $\sum_{i=1}^{k} \delta_i a_i \in (a, b)$ is at most $\binom{k}{\lfloor k/2 \rfloor}$.*

Fix a vector $(\epsilon_{k+1}, \ldots, \epsilon_n) \in B_{n-k}$. Denote the sum $\sum_{i=k+1}^{n} \epsilon_i a_i$ by S. Now $|\sum_{i=1}^{n} \epsilon_i a_i| < a_{k+1}$ if and only if $\sum_{i=1}^{k} \epsilon_i a_i$ belongs to the open interval $(-S - a_{k+1}, -S + a_{k+1})$. However, by the lemma above, there are at most $\binom{k}{\lfloor k/2 \rfloor}$ vectors $(\epsilon_1, \ldots, \epsilon_k)$ with this property. Since we can fix $(\epsilon_{k+1}, \ldots, \epsilon_n) \in B_{n-k}$ in $|B_{n-k}| = 2^{n-k}$ ways, the assertion of the proposition follows, implying the assertion of Theorem 3.1 as well.

For an integer $p \geq 2$, a p-partition of a set A is a collection $A_1, A_2, \ldots, A_p$ of subsets of A such that $\cup_{i=1}^{p} A_i = A$ and $A_i \cap A_j = \emptyset$ for each $1 \leq i \neq j \leq p$. Theorem 3.1 was generalized in [18], where the following minimum p-processor scheduling problem was considered. We are given an integer $p \geq 2$

and a sequence $w_1, w_2, \ldots, w_n$ of positive integers, and we are required to find a p-partition $N_1, N_2, \ldots, N_p$ of $\{1, 2, \ldots, n\}$ such that $\max_{i=1}^{p} \sum_{j \in N_i} w_j$ is as small as possible. Notice that the minimum 2-processor scheduling problem is equivalent to MMP from the Domination Analysis point of view.

3.2 Max Cut

The Max Cut (MC) is the following problem: given a weighted complete graph $G = (V, E, \mathrm{wt})$, find a bipartition (a *cut*) (X, Y) of V such that the sum of weights of the edges with one end vertex in X and the other in Y, called the *weight of the cut* (X, Y), is maximum.

We will show that the MC is $\mathcal{DOM}$-easy, just as TSP is. (For the definition of $\mathcal{DOM}$-easy problems, see Subsection 1.3.)

Theorem 3.4 *[20] The MC is $\mathcal{DOM}$-easy. In fact, there is an algorithm, for the MC, of domination number at least $\Omega(2^n/n)$.*

Proof: Let $G = (V, E)$ be a weighted complete graph with $n = |V|$ vertices and let W be the sum of the weights of the edges in G. Clearly, the average weight of a cut of G is $\overline{W} = W/2$.

Consider the following well-known approximation algorithm $\mathcal{C}$ that always produces a cut of weight at least $\overline{W}$. The algorithm $\mathcal{C}$ considers the vertices of G in any fixed order $v_1, v_2, \ldots, v_n$, initiates $X = \{v_1\},\ Y = \{v_2\}$, and adds v_i, $i \geq 3$, to X or Y depending on whether the sum of the weights of edges between v and Y or between v and X is larger. We will prove that $\mathcal{C}$ is of domination number at least $\Omega(2^n/n)$. To show this, it suffices to prove that the cuts in G of weight at most $\overline{W}$ (we call them *bad cuts*) constitute at least an $O(1/n)$ part of all cuts.

We call a cut (X, Y) of G a *k-cut* if $|X| = k$. We evaluate the fraction of bad cuts among k-cuts when $k \leq n/2 - 2\sqrt{n}$.

For a fixed edge uv of G among $\binom{n}{k}$ k-cuts there are $2\binom{n-2}{k-1}$ k-cuts that contain uv. Thus, the average weight of a k-cut is $\overline{W}_k = 2\binom{n-2}{k-1}W/\binom{n}{k}$. Let b_k be the number of bad k-cuts. Then, $(\binom{n}{k} - b_k)\overline{W}/\binom{n}{k} \leq \overline{W}_k$. Hence,

$$b_k \geq \binom{n}{k} - 4\binom{n-2}{k-1} \geq \binom{n}{k}\left(1 - \frac{4k(n-k)}{n(n-1)}\right).$$

It is easy to verify that $1 - 4k(n-k)/(n(n-1)) > 1/n$ for all $k \leq n/2 - 2\sqrt{n}$. Hence, G has more than $\frac{1}{n}\sum_{k \leq n/2 - 2\sqrt{n}} \binom{n}{k}$ bad cuts. By the famous DeMoivre-Laplace theorem of probability theory, it follows that the last sum is at least $c2^n$ for some positive constant c. Thus, G has more than $c2^n/n$ bad cuts. □

Using a more advanced probabilistic approach Alon, Gutin and Krivelevich [2] recently proved that the algorithm $\mathcal{C}$ described above is of domination ratio larger than 0.025.

3.3 Max-k-SAT

One of the best-known NP-complete decision problems is 3-SAT. This problem is the following: We are given a set V of variables and a collection C of clauses each with exactly 3 literals (a literal is a variable or a negated variable in V; in a clause, literals are separated by "OR"'s). Does there exist a truth assignment for V, such that every clause is *true*?

We will now consider the more general optimization problem max-k-SAT. This is similar to 3-SAT, but there are k literals in each clause, and we want to find a truth assignment for V which maximizes the number of clauses that are *true*, i.e., *satisfied*. Let $U = \{x_1, \ldots, x_n\}$ be the set of variables in the instance of max-k-SAT under consideration. Let $\{C_1, \ldots, C_m\}$ be the set of clauses. We assume that k is a constant.

Berend and Skiena [10] considered some well-known algorithms for max-k-SAT and the algorithms turned out to have domination number at most $n+1$. However an algorithm considered in [20] is of domination number at least $\Omega(2^n/n^{\lfloor k/2 \rfloor})$. We will study this algorithm.

Assign a truth assignment to all the variables at random. Let p_i be the probability that C_i (i'th clause) is satisfied. Observe that if some variable and its negation belong to C_i, then $p_i = 1$, otherwise $p_i = 1 - 2^{-k'}$ where k' is the number of distinct variables in C_i. Thus, the average number of satisfied clauses in a random truth assignment is $E = \sum_{i=1}^{m} p_i$.

For simplicity, in the sequel *true* (*false*) will be replaced by the binaries 1 (0). By a construction described in Section 15.2 of [3], there exists a binary matrix $A = (a_{ij})$ with n columns and $r = O(n^{\lfloor k/2 \rfloor})$ rows such that the following holds: Let B be an arbitrary submatrix of A, consisting of k of its columns (chosen arbitrarily), and all r of its rows. Every binary k-vector coincides with exactly $r/2^k$ rows of B. We give a short example below, with $n = 4$ and $k = 3$ ($r = 8$). The matrix A can be constructed in polynomial time [3].

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}$$

Note that no matter which 3 columns we consider, we will always get the vectors (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1) equally many times (in this case, once) in the 8 rows.

Observe that each row, say row j, corresponds to a truth assignment β_j (where the i'th variable gets the truth assignment of the i'th column, i.e. $x_1 = a_{j1}, \ldots, x_n = a_{jn}$). Let T_j be the number of clauses satisfied by β_j. Consider a polynomial time algorithm $\mathcal{S}$ that computes $T_1, \ldots, T_r$ and outputs $\beta^*(A)$ that satisfies $T^*(A) = \max_{j=1}^{r} T_j$ clauses.

We will prove that $\mathcal{S}$ has domination number at least $\Omega(2^n/n^{\lfloor k/2 \rfloor})$. Since rp_i of the r truth assignments will satisfy the i'th clause, we conclude that $\sum_{i=1}^{r} T_i = rE$ (recall that $E = \sum_{i=1}^{m} p_i$; see also Question 3.6). Therefore the truth assignment $\beta^*(A)$ must satisfy at least E clauses. Furthermore by a similar argument we conclude that the row $\beta_*(A)$ corresponding to the truth assignment with fewest satisfied clauses, which we shall call $W(A)$, has at most E satisfied clauses.

Let $X \subseteq \{1, 2, \ldots, n\}$ be arbitrary and let A_X be the matrix obtained from A by changing all 0's to 1's and all 1's to 0's in the i'th column in A, for all $i \in X$. Note that A_X has the same properties as A. Observe that a truth assignment can appear at most r times in $\mathcal{T} = \{\beta_*(A_X) : X \subseteq \{1, 2, \ldots, n\}\}$, as a truth assignment cannot appear in the j'th row of A_X and A_Y, if $X \neq Y$. Therefore $\mathcal{T}$ contains at least $2^n/r$ distinct truth assignments all with at most E satisfied clauses. Therefore, we have proved the following:

Theorem 3.5 *[20] The algorithm $\mathcal{S}$ is of domination number at least $\Omega(2^n/n^{\lfloor k/2 \rfloor})$.*

Question 3.6 *Consider the given algorithm for max-k-SAT. Prove that rp_i rows will result in the i'th clause being* true*, so $\sum_{i=1}^{r} T_i = rE$.*

Question 3.7 *[20] Show that Theorem 3.5 can be extended to the weighted version of max-k-SAT, where each clause C_i has a weight w_i and we wish to maximize the total weight of satisfied clauses.*

Alon, Gutin and Krivelevich [2] recently proved, using an involved probabilistic argument, that the algorithm of Theorem 3.5 is, in fact, of domination number $\Omega(2^n)$.

3.4 Fixed span frequency assignment problem

In [31] the domination number is computed for various heuristics for the *Fixed Span Frequency Assignment Problem* (fs-FAP), which is defined as follows. We are given a set of vertices $\{x_1, x_2, \ldots, x_n\}$ and an $n \times n$ matrix $C = (c_{ij})$. We want to assign a frequency f_i to each vertex x_i, such that $|f_i - f_j| \geq c_{ij}$ for all $i \neq j$. However when f_i has to be chosen from a set of frequencies $\{0, 1, \ldots, \sigma - 1\}$, where σ is a fixed integer, then this is not always possible. If $|f_i - f_j| < c_{ij}$, then let $x_{ij} = 1$, and otherwise let $x_{ij} = 0$.

We are also given a matrix $W = (w_{ij})$ of weights, and we want to minimize the sum $\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} w_{ij}$. In other words we want to minimize the weight of all the edges that are *broken* (i.e. which have $|f_i - f_j| < c_{ij}$). Put $c_{ii} = 0$ for all $i = 1, 2, \ldots, n$. Since every vertex may be assigned a frequency in $\{0, 1, \ldots, \sigma - 1\}$, the following holds.

Proposition 3.8 *The total number of solutions for the fs-FAP is* σ^n*.*

A heuristic for the fs-FAP, which has similarities with the greedy expectation algorithm for the TSP (see [21]) is as follows (see [31] for details). We will assign a frequency to each vertex $x_1, x_2, \ldots, x_n$, in that order. Assume that we have already assigned frequencies to $x_1, x_2, \ldots, x_{i-1}$ and suppose that we assign frequency f_i to x_i. For all $j > i$, let p_{ij} be the probability that $|f_i - f_j| < c_{ij}$, if we assign a random frequency to j. For all $j < i$ let $x_{ij} = 1$ if $|f_i - f_j| < c_{ij}$ and $x_{ij} = 0$ otherwise. We now assign the frequency f_i to x_i, which minimizes the following:

$$\sum_{j=1}^{i-1} w_{ij} x_{ij} + \sum_{j=i+1}^{n} w_{ij} p_{ij}.$$

In other words we choose the frequency which minimizes the weight of the constraints that get broken added to the average weight of constraints that will be broken by assigning the remaining vertices with random frequencies. It is not too difficult to see that the above approach produces an assignment of frequencies, such that the weight of the broken edges is less than or equal to the average, taken over all assignments of frequencies. Koller and Noble [31] proved the following theorem where $\mathcal{G}$ is the algorithm described above.

Theorem 3.9 *[31] The domination number of* $\mathcal{G}$ *is at least* $\sigma^{n - \lceil log_2 n \rceil - 1}$.

Note that the following holds.

$$\sigma^{n - \lceil log_2 n \rceil - 1} \geq \sigma^{n - log_2 n - 2} \geq \frac{\sigma^n}{\sigma^2 n^{log_2 \sigma}}$$

Therefore $\mathcal{G}$ finds a solution which is at least as good as a polynomial fraction of all solutions (σ is a constant). This means that the fs-FAP is $\mathcal{DOM}$-easy.

In [31] it is furthermore shown that $\mathcal{G}$ has higher domination number than the better-known greedy-type algorithm, which minimizes only $\sum_{j=1}^{i-1} w_{ij}x_{ij}$, in each step.

3.5 $\mathcal{DOM}$-hard problems

In this section we consider two well-known graph theory problems, which are somewhat different from the previous problems we have considered. Firstly, the number of feasible solutions, for an input of size n, depends on the actual input, and not just its size.

A *clique* in a graph G is a set of vertices in G such that every pair of vertices in the set are connected by an edge. The *Maximum Clique Problem* (MCl) is the problem of finding a clique of maximum cardinality in a graph. A *vertex cover* in a graph G is a set S of vertices in G such that every edge is incident to a vertex in S. The *Minimum Vertex Cover Problem* (MVC) is the problem of finding a minimum cardinality vertex cover. It is easy to see that the number of cliques in a graph depends on its structure, and not only on the number of vertices. The same holds for vertex covers.

The problems we have considered in the previous subsections have been $\mathcal{DOM}$-easy. We will show that MCl and MVC are $\mathcal{DOM}$-hard unless P=NP.

Theorem 3.10 *[20] MCl is $\mathcal{DOM}$-hard unless P=NP.*

Proof: We use a result by Håstad [29], which states that, provided that P$\neq$NP, MCl is not approximable within a factor $n^{1/2-\epsilon}$ for any $\epsilon > 0$, where n is the number of vertices in a graph.

Let G be a graph with n vertices, and let q be the number of vertices in a maximum clique Q of G. Let $\mathcal{A}$ be a polynomial time algorithm and let $\mathcal{A}$ find a clique M with m vertices in G.

Since the clique Q 'dominates' all 2^q of its subcliques and the clique M 'dominates' at most $\binom{n}{m}2^m$ cliques in G, the domination ratio r of $\mathcal{A}$ is at most $\binom{n}{m}2^m/2^q$. By the above non-approximability result of Håstad [29], we may assume that $mn^{0.4} \leq q$. Thus,

$$r \leq \frac{\binom{n}{m}2^m}{2^q} \leq \frac{(en/m)^m 2^m}{2^q} \leq \frac{(n/m)^m(2e)^m}{2^{mn^{0.4}}} = 2^s,$$

where $s = m(\log n - \log m + 1 + \log e - n^{0.4})$. Clearly, 2^s is smaller than $1/p(n)$ for any polynomial $p(n)$ when n is sufficiently large. $\square$

An *independent set* in a graph is a set S of vertices such that no edge is joins two vertices in S. The *Maximum Independent Set* problem (MIS) is the problem of finding a minimum cardinality independent set in a graph.

Question 3.11 *Using Theorem 3.10 prove that MIS is $\mathcal{DOM}$-hard unless P=NP.*

Question 3.12 *Let $G = (V, E)$ be a graph. Prove that S in an independent set in G if and only if $V - S$ is a vertex cover in G.*

Question 3.13 *Using the results of the previous two questions prove that MVC is $\mathcal{DOM}$-hard unless P=NP.*

3.6 Other problems

There are many other combinatorial optimization problems studied in the literature that were not considered above. We will overview some of them.

In the *Generalized TSP*, we are given a weighted complete directed or undirected graph G and a partition of its vertices into non-empty sets $V_1, \dots, V_k$. We are required to compute a lightest cycle in G containg exactly one vertex from each V_i, $i = 1, \dots, k$. In the case when all V_i's are of the same cardinality, Ben-Arieh et al. [8] proved that the Generalized TSP is $\mathcal{DOM}$-easy.

The *Quadratic Assignment Problem* (QAP) can be formulated as follows. We are given two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of integers. Our aim is to find a permutation π of $\{1, 2, \dots, n\}$ that minimizes the sum

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}.$$

Using group-theoretical approaches, Gutin and Yeo [25] proved only that QAP is $\mathcal{DOM}$-easy when n is a prime power.

Conjecture 3.14 *QAP is $\mathcal{DOM}$-easy (for every value of n).*

It was noted in [20] that Theorem 3.10 holds for some cases of the following general problem: the *Maximum Induced Subgraph with Property* Π (MISP), see Problem GT25 in the compendium of [4]). The property Π must be hereditary, i.e., every induced subgraph of a graph with property Π has property Π, and non-trivial, i.e., it is satisfied for infinitely many graphs and false for infinitely many graphs. Lund and Yannakakis [32] proved that MISP is not approximable within n^ϵ for some $\epsilon > 0$ unless P=NP, if Π is hereditary, non-trivial and is false for some clique or independent set (e.g., planar, bipartite, triangle-free). This non-approximability result can be used as in the proof of Theorem 3.10.

4 Greedy algorithm

The main practical message of this section is that one should be careful while using the classical greedy algorithm in combinatorial optimization (CO): there are many instances of CO problems for which the greedy algorithm will produce the unique worst possible solution. Moreover, this is true for several well-known optimization problems and the corresponding instances are not exotic, in a sense. This means that not always the paradigm of greedy optimization provides any meaningful optimization at all.

In this section we provide a wide extension of Theorem 1.2, which slightly generalizes the main theorem in [23]. Interestingly, the proof of the extension is relatively easy.

An *independence system* is a pair consisting of a finite set E and a family $\mathcal{F}$ of subsets (called *independent sets*) of E such that (I1) and (I2) are satisfied.

(I1) The empty set is in $\mathcal{F}$;

(I2) If $X \in \mathcal{F}$ and Y is a subset of X, then $Y \in \mathcal{F}$.

A maximal (with respect to inclusion) set of $\mathcal{F}$ is called a *base*. Clearly, an independence system on a set E can be defined by its bases. Notice that bases may be of different cardinality.

Many combinatorial optimization problems can be formulated as follows. We are given an independence system $(E, \mathcal{F})$ and a weight function wt that assigns a real weight $\mathrm{wt}(e)$ to every element $e \in E$. The weight $\mathrm{wt}(S)$ of $S \in \mathcal{F}$ is defined as the sum of the weights of the elements of S. It is required to find a base $B \in \mathcal{F}$ of minimum weight. In this section, we will consider only such problems and call them the $(E, \mathcal{F})$-*optimization problems.*

If $S \in \mathcal{F}$, then let $I(S) = \{x : S \cup \{x\} \in \mathcal{F}\} - S$. The greedy algorithm (or, `greedy`, for short) constructs a base as follows: `greedy` starts from an empty set X, and at every step `greedy` takes the current set X and adds to it a minimum weight element $e \in I(X)$; `greedy` stops when a base is built.

Consider the following example. Let $E' = \{a, b, c, d\}$. We define an independence system $(E', \mathcal{F}')$ by listing its two bases: $\{a, b, c\}$, $\{c, d\}$. Recall that the independent sets of $(E', \mathcal{F}')$ are the subsets of its bases. Let the weights of a, b, c, d be $1, 5, 0, 2$, respectively. (Notice that the weight assignment determines an instance of the $(E', \mathcal{F}')$-optimization problem.) `greedy` starts from $X = \emptyset$, then adds c to X. At the next iteration it appends a to X. `greedy` cannot add d to $X = \{a, c\}$ since $d \notin I(X)$. Thus, `greedy` appends b to X and stops.

Since $6 = \mathrm{wt}(a, b, c) > \mathrm{wt}(c, d) = 2$, the domination number of `greedy` is 1 for this instance of the $(E', \mathcal{F}')$-optimization problem.

Note that if we add (I3) below to (I1),(I2), then we obtain one of the definitions of a matroid [33]:

(I3) If U and V are in $\mathcal{F}$ and $|U| > |V|$, then there exists $x \in U - V$ such that $V \cup \{x\} \in \mathcal{F}$.

It is well-known that domination number of `greedy` for every matroid $(E, \mathcal{F})$ is $|\mathcal{F}|$: `greedy` always finds an optimum for the $(E, \mathcal{F})$-optimization problem. Thus, it is surprising to have the following theorem that generalizes Theorem 1.2.

Theorem 4.1 *[23, 24] Let $(E, \mathcal{F})$ be an independence system and $B' = \{x_1, \ldots, x_k\}$, $k \geq 2$, a base. Suppose that the following holds for every base $B \in \mathcal{F}$, $B \neq B'$,*

$$\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| < k(k+1)/2. \tag{1}$$

Then the domination number of `greedy` *for the $(E, \mathcal{F})$-optimization problem equals 1.*

Proof: Let M be an integer larger than the maximal cardinality of a base in $(E, \mathcal{F})$. Let $\mathrm{wt}(x_i) = iM$ and $\mathrm{wt}(x) = 1 + jM$ if $x \notin B'$, $x \in I(x_1, x_2, \ldots, x_{j-1})$ but $x \notin I(x_1, x_2, \ldots, x_j)$. Clearly, `greedy` constructs B' and $\mathrm{wt}(B') = Mk(k+1)/2$.

Let $B = \{y_1, y_2, \ldots, y_s\}$ be a base different from B'. By the choice of wt made above, we have that $\mathrm{wt}(y_i) \in \{aM, aM + 1\}$ for some positive integer a.

Clearly

$$y_i \in I(x_1, x_2, \ldots, x_{a-1}),$$

but $y_i \notin I(x_1, x_2, \ldots, x_a)$. Hence, by (I2), y_i lies in $I(x_1, x_2, \ldots, x_j) \cap B$, provided $j \le a-1$. Thus, y_i is counted a times in $\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B|$. Hence,

$$\begin{aligned} \mathrm{wt}(B) &= \sum_{i=1}^{s} \mathrm{wt}(y_i) \le s + M \sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| \\ &\le s + M(k(k+1)/2 - 1) = s - M + \mathrm{wt}(B'), \end{aligned}$$

which is less than the weight of B' as $M > s$. Since $\mathcal{A}$ finds B', and B is arbitrary, we see that `greedy` finds the unique heaviest base. $\square$

The strict inequality (1) cannot be relaxed to the non-strict one due to Question 4.3.

Question 4.2 *Let $(E, \mathcal{F})$ be a matroid. Using (I3), show that for two distinct bases B and $B' = \{x_1, x_2, \ldots, x_k\}$, we have that $|I(x_1, x_2, \ldots, x_j) \cap B| \ge k - j$ for $j = 0, 1, \ldots, k$. Thus,*

$$\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| \ge k(k+1)/2.$$

Question 4.3 *[23] Consider a matroid $(E', \mathcal{F}')$ in which E' consists of the columns of matrix $M = (I|2I)$, where I is the $k \times k$ identity matrix, and $\mathcal{F}'$ consists of collections of linearly independent columns of M. (Check that $(E', \mathcal{F}')$ is a matroid.) Let B and $B' = \{x_1, x_2, \ldots, x_k\}$ be two distinct bases of our matroid. Show that $\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| = k(k+1)/2$.*

Recall that by the Assignment Problem (AP) we understand the problem of finding a lightest perfect matching in a weighted complete bipartite graph $K_{n,n}$.

Question 4.4 *Prove Corollary 4.5 applying Theorem 4.1*

Corollary 4.5 *[23] Every greedy-type algorithm $\mathcal{A}$ is of domination number 1 for the Asymmetric TSP, Symmetric TSP and AP.*

Research Question 4.6 *Describe new wide families of the $(E, \mathcal{F})$-optimization problems for which* `greedy` *is of domination number 1.*

Bang-Jensen, Gutin and Yeo [7] considered the $(E, \mathcal{F})$-optimization problems, in which every base is of the same cardinality and wt assumes only a finite number of integral values. For such problems, the authors of [7] completely characterized all cases when `greedy` *may* construct the unique worst possible solution. Here the word may means that `greedy` may choose any element of E of the same weight.

5 Practicality of domination analysis

Earlier in this chapter we have seen that domination analysis (DA) provides theoretical explanations of the poor computational behavior of `greedy` for certain optimization problems and of the fact that some very large neighborhoods in local search are computationally much weaker than some 'small' neighborhoods.

One might wonder whether a heuristic $\mathcal{A}$, that is significantly better that another heuristic $\mathcal{B}$ from the point of view of DA, is better that $\mathcal{B}$ in computational experiments. In particular, whether `greedy` is worse, in computational experiments, than any ATSP heuristic of domination number at least $(n-2)!$? Generally speaking the answer to this natural question is negative. This is because computational experiments and domination analysis indicate different aspects of quality of heuristics. Nevertheless, it seems that many heuristics of very small domination number such as `greedy` for TSP fail also in computational experiments and thus are not very robust.

Koller and Noble [31] showed that the heuristic $\mathcal{G}$ that they introduced for the frequency assignment problem is of larger domination number than the well-known greedy algorithm. However, the greedy algorithm is usually better in computational experiments than $\mathcal{G}$. Judging only by the computational experiments, $\mathcal{G}$ is of no interest. However, $\mathcal{G}$ might well be of interest when difficult instances of the frequency assignment problem are considered.

Ben-Arieh et al. [8] studied some heuristics for the Generalized TSP defined above. They investigated three modifications of a generic heuristic. In the computational experiment in [8] one of the modifications was clearly inferior to the other two. The best two behaved very similarly. Nevertheless, the authors of [8] managed to 'separate' the two modifications by showing that one of the modifications was of much larger domination number.

Acknowledgment We would like to thank Tommy Jensen, Alek Vainshtein and the editors for a number of helpful comments. Research of both authors is supported in part by a Leverhulme Trust grant.

References

[1] R.K. Ahuja, Ö. Ergun, J.B. Orlin and A.P. Punnen, A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.* **123** (2002) 75–102.

[2] N. Alon, G. Gutin and M. Krivelevich, Algorithms with large domination ratio. To appear in*J. Algorithms*.

[3] N. Alon and J.H. Spencer, *The Probabilistic Method*, 2nd edition, Wiley, New York, 2000.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and Approximation*, Springer, Berlin, 1999.

[5] E. Bach and J. Shallit, *Algorithmic number theory, Volume 1*, MIT Press, Cambridge, Ma., 1996.

[6] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer-Verlag, London, 2000.

[7] J. Bang-Jensen, G. Gutin and A. Yeo, When the greedy algorithm fails. Submitted.

[8] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo and A. Zverovitch, Transformations of Generalized ATSP into ATSP: experimental and theoretical study. *Oper. Res. Lett.* **31** (2003) 357–365.

[9] C. Berge, *The Theory of Graphs*, Methuen, London, 1958.

[10] D. Berend and S.S. Skiena, Combinatorial dominance guarantees for heuristic algorithms. Manuscript, 2002.

[11] G. Cornuejols, M.L. Fisher and G.L. Nemhauser, Location of bank accounts to optimize float; an analytic study of exact and approximate algorithms. *Management Sci.* **23** (1977) 789–810.

[12] V.G. Deineko and G.J. Woeginger, A study of exponential neighbourhoods for the traveling salesman problem and the quadratic assignment problem, *Math. Prog. Ser. A* **87** (2000) 519–542.

[13] P. Erdős, On a lemma of Littlewood and Offord, *Bull. Amer. Math. Soc.* **51** (1945) 898–902.

[14] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multicriteria combinatorial optimization. *OR Spektrum* **22** (2000) 425–460.

[15] Ö. Ergun, J.B. Orlin and A. Steele-Feldman, Creating very large scale neighborhoods out of smaller ones by compounding moves: a study on the vehicle routing problem. Manuscript, 2002.

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

[17] G. Gutin. On an approach to solving the TSP. In *Proceedings of the USSR Conference on System Research,* pages 184-185. Nauka, Moscow, 1984. (in Russian).

[18] G. Gutin, T. Jensen and A. Yeo, Domination analysis for minimum multiprocessor scheduling. Submitted.

[19] G. Gutin and A.P. Punnen, eds., *The Traveling Salesman Problem and its Variations*, Kluwer, Dordrecht, 2002.

[20] G. Gutin, A. Vainshtein and A. Yeo, Domination Analysis of Combinatorial Optimization Problems. *Discrete Appl. Math.* **129** (2003) 513–520.

[21] G. Gutin and A. Yeo, Polynomial approximation algorithms for the TSP and the QAP with a factorial domination number. *Discrete Appl. Math.* **119** (2002) 107–116.

[22] G. Gutin and A. Yeo, Upper bounds on ATSP neighborhood size. *Discrete Appl. Math.* **129** (2003) 533–538.

[23] G. Gutin and A. Yeo, Anti-matroids. *Oper. Res. Lett.* **30** (2002) 97–99.

[24] G. Gutin and A. Yeo, Introduction to domination analysis. Submitted.

[25] G. Gutin, A. Yeo and A. Zverovitch, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. *Discrete Appl. Math.* **117** (2002) 81–86.

[26] G. Gutin, A. Yeo and A. Zverovitch, Exponential Neighborhoods and Domination Analysis for the TSP. In *The Traveling Salesman Problem and its Variations* (G. Gutin and A.P. Punnen, eds.), Kluwer, Dordrecht, 2002.

[27] D.S. Johnson and L.A. McGeoch, The traveling salesman problem: A case study in local optimization. In *Local Search in Combinatorial Optimization* (E.H.L. Aarts and J.K. Lenstra, eds.), Wiley, Chichester, 1997.

[28] D.S. Johnson, G. Gutin, L. McGeoch, A. Yeo, X. Zhang, and A. Zverovitch, Experimental Analysis of Heuristics for ATSP. In *The Traveling Salesman Problem and its Variations* (G. Gutin and A. Punnen, eds.), Kluwer, Dordrecht, 2002.

[29] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, *Acta Mathematica* **182** (1999) 105–142.

[30] H. Kise, T. Ibaraki and H. Mine, Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times. *J. Oper. Res. Soc. Japan* **22** (1979) 205–223.

[31] A.E. Koller and S.D. Noble, Domination analysis of greedy heuristics for the frequency assignment problem. To appear in *Discrete Math.*

[32] C. Lund and M. Yannakakis, The approximation of maximum subgraph problems. *Proc. 20th Int. Colloq. on Automata, Languages and Programming*, Lect. Notes Comput. Sci. 700 (1993) Springer, Berlin, 40–51.

[33] J. Oxley, *Matroid Theory.* Oxford Univ. Press, Oxford, 1992.

[34] A.P. Punnen, F. Margot and S.N. Kabadi, TSP heuristics: domination analysis and complexity. *Algorithmica* **35** (2003) 111-127.

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

[36] V.I. Rublineckii, Estimates of the accuracy of procedures in the Traveling Salesman Problem. *Numerical Mathematics and Computer Technology,*, no. 4 (1973) 18–23 (in Russian).

[37] V.I. Sarvanov, On the minimization of a linear from on a set of all n-elements cycles. *Vestsi Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk* no. 4 (1976) 17–21 (in Russian).

[38] V.I. Sarvanov and N.N. Doroshko, Approximate solution of the traveling salesman problem by a local algorithm with scanning neighbourhoods of factorial cardinality in cubic time. In *Software: Algorithms and Programs.* Math. Institute of Belorussian Acad. of Sci., Minsk, no. 31, 11-13 (1981) (in Russian).

[39] T.W. Tillson, A hamiltonian decomposition of K^*_{2m}, $m \geq 8$. *J. Combinatorial Theory* **B 29** (1980) 68-74.

[40] E. Zemel, Measuring the quality of approximate solutions to zero-one programming problems. *Math. Oper. Res.* **6** (1981) 319–332.