1 Introduction

This paper is chiefly a survey article. It gives a short introduction to domination analysis (DA) and a brief overview of results and applications. We prove three new results, Theorems 4.1, 5.1 and 5.4, that strengthen previously known results. Notice that the proofs of the new results are minor variations of the proofs of their previously known counterparts. Some open problems are also raised. We would refer the interested reader to [17] and [15] for more detailed information.

In Section 2 we provide motivation for studying DA, basic terminology and notation. Section 3 considers in some detail the minimum multiprocessor scheduling and overviews several results on other combinatorial optimization problems. Sections 4 and 5 are devoted to results on DA for the greedy algorithm and upper bounds on the maximum domination number for polynomial time traveling salesman heuristics.

2 Motivation and Terminology

Research on combinatorial optimization (CO) heuristics has produced a large variety of heuristics especially for well-known CO problems. Thus, we should develop ways of selecting 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 usually does not cover the hardest instances.
Let $P$ be a CO problem, and let $\mathcal{H}$ be a heuristic for $P$. Let $X(I)$ denote all feasible solutions for some instance $I \in P$ and let $|I|$ be the size of $I$. We denote the solution obtained by $\mathcal{H}$ for an instance $I$ of $P$ by $x(I)$. Furthermore let $\text{opt}(I)$ denote the optimal solution of $I$. When considering the cost of a solution $y$ we write $c(y)$.

The theoretical performance of CO heuristics is normally studied by the means of approximation analysis [3]. Usually, upper or lower bounds for the worst case performance ratio are obtained, where the performance ratio is defined as

$$\max_{I \in P : |I| = n} \left\{ \frac{c(x(I))}{c(\text{opt}(I))} : \frac{c(\text{opt}(I))}{c(x(I))} \right\}.$$ 

Introduced in [8], DA provides an alternative and a complement to approximation analysis. In DA, we are usually interested in the domination number or domination ratio of the heuristic solution.

**Definition 2.1** The domination number of a heuristic $\mathcal{H}$ is

$$\text{domn}(\mathcal{H}, n) = \min_{I \in P : |I| = n} \text{domn}(\mathcal{H}, I),$$

where $\text{domn}(\mathcal{H}, I) = |\{y \in X(I) : c(x(I)) \leq c(y)\}|$.

In other words, the domination number $\text{domn}(\mathcal{H}, n)$ is the maximum integer $m(n)$ such that the solution $x(I)$ obtained by $\mathcal{H}$ for any instance $I$ of $P$ of size $n$ is not worse than at least $m(n)$ feasible solutions of $I$ (including $x(I)$).

Since the number of all feasible solutions for $I$ of $P$ may vary for different choices of $I$ with the same size it is often useful to consider the domination ratio of a heuristic $\mathcal{H}$.

**Definition 2.2** The domination ratio of a heuristic $\mathcal{H}$ is

$$\text{domr}(\mathcal{H}, n) = \min_{I \in P : |I| = n} \frac{\text{domn}(\mathcal{H}, I)}{|X(I)|}.$$

Clearly, domination ratio belongs to the interval $(0, 1]$ and exact algorithms are of domination ratio 1.

In many cases, domination analysis is very useful. For example, the greedy algorithm has domination number 1 for many CO problems, see Section 4. In other words, the greedy algorithm, in the worst case, produces the unique worst possible solution. This is reflected in recent computational
experiments with the greedy algorithm, see, e.g., [19], where it was concluded that the greedy algorithm ‘might be said to self-destruct’.

The Asymmetric Traveling Salesman Problem (ATSP) is the problem of computing a minimum weight tour (Hamilton cycle) in a weighted complete digraph on \( n \) vertices. 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, denoted by \( \text{max TSP} \), is of interest.

APX is the class of CO problems that admit polynomial time approximation algorithms with a constant performance ratio [3]. It is well known that while max TSP belongs to APX, TSP does not [3]. 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 by first multiplying every weight by \(-1\) and then adding a big constant to all the weights in order to preserve non-negativity. 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 in the 1970’s, see, e.g., [6, 20, 24]. Notice that from the DA point of view max TSP and TSP are equivalent problems.

Zemel [24] 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 [3]) do not satisfy the last property, the parameter \( 1 - r \), where \( r \) is the domination ratio, does satisfy all of the properties.

Local search is one of the most successful approaches in the design of CO heuristics. Recently, several researchers carried out computational and theoretical investigation of local search with very large scale neighborhoods (see, e.g., [1, 7, 17]). It was believed that the larger the neighborhood the better solution is expected to be found. However, some computational experiments do not support this hypothesis. This means that some other parameters are responsible for the relative power of a neighborhood. Computational experiments and theoretical results (see, e.g., [17, 23]) on TSP indicate that one such parameter may well be the domination ratio of the corresponding local search.

In our view, it is advantageous to have bounds for both performance ratio and domination ratio of a heuristic whenever it is possible. Roughly speaking this will enable us to see a 2D picture rather than a 1D picture.
We will now illustrate definitions 2.1 and 2.2 using the ATSP. The fact that the nearest neighbor algorithm for ATSP is of domination number 1 (first proved in [16]) means that for every \( n \geq 2 \), there is an instance of ATSP on \( n \) vertices, for which the nearest neighbor algorithm finds the unique worst possible Hamilton cycle. Since the number of distinct Hamilton cycles in an \( n \)-vertex complete digraph is \((n-1)!\), we see that the nearest neighbor algorithm is of domination ratio \( 1/(n-1)! \). There are many ATSP algorithms of domination number at least \((n-2)!\) [17], i.e., in the worst case they guarantee that their Hamilton cycle is at least as good as \((n-2)! - 1\) other Hamilton cycles.

An algorithm \( A \) for a CO problem \( P \) is \( \text{DOM-good} \) if \( A \) is of polynomial time complexity and there exists a polynomial \( p \) in \( n \) such that the domination ratio of \( A \) is at least \( 1/p(n) \) for any size \( n \) of \( P \). A CO problem \( P \) is \( \text{DOM-easy} \) if it admits a \( \text{DOM-good} \) algorithm and \( P \) is \( \text{DOM-hard} \) if there is no \( \text{DOM-good} \) algorithm for \( P \). The above mentioned algorithms for ATSP are of domination ratio \( \Omega(1/(n-1)) \) and thus ATSP is \( \text{DOM-easy} \).

3 Combinatorial Optimization Problems

Domination analysis has mainly been studied for different algorithms for the TSP (see [16, 17, 22, 23]). Recently domination analysis has been applied to some other combinatorial optimization problems, such as the the minimum multiprocessor scheduling problem, weighted Max Cut, Max Clique, Min Vertex Cover and the assignment problem (see for example [2, 11, 13, 21]).

In this section we will introduce some of the results that have been obtained so far, starting with the minimum multiprocessor scheduling problem (MMSP).

Let \( p \geq 2 \) be an integer and let \( S \) be a finite set. A \( p \)-partition of \( S \) is a \( p \)-tuple \((A_1, A_2, \ldots, A_p)\) of subsets of \( S \) such that \( A_1 \cup A_2 \cup \ldots \cup A_p = S \) and \( A_i \cap A_j = \emptyset \) for all \( 1 \leq i < j \leq p \).

Let \( N \) denote the set \( \{1, 2, \ldots, n\} \) and let each \( i \in N \) be assigned a positive integral weight \( \sigma(i) \). For a subset \( A \) of \( N \), \( \sigma(A) = \sum_{i \in A} \sigma(i) \). The MMSP [3] can be stated as follows. We are given a triple \((N, \sigma, p)\), where \( p \) is an integer, \( p \geq 2 \). We are required to find a \( p \)-partition \( C \) of \( N \) that minimizes \( \sigma(A) = \max_{1 \leq i \leq p} \sigma(A_i) \) over all \( p \)-partitions \( A = (A_1, A_2, \ldots, A_p) \) of \( N \). We assume that \( p < n \) as the opposite case is trivial.

As an illustration, we first consider a special case of MMSP when \( p = 2 \), which we denote M2SP and consider the following greedy-type algorithm \( G \) for the M2SP: \( G \) sorts the weights such that \( \sigma(\pi(1)) \geq \sigma(\pi(2)) \geq \cdots \geq \sigma(\pi(n)) \).
\(\sigma(\pi(n))\), initiates \(A_1 = \{\pi(1)\}\), \(A_2 = \{\pi(2)\}\), and, for each \(j \geq 3\), puts \(\pi(j)\) into the set \(A_k\) of the current 2-partition with smallest \(\sigma(A_k)\), \(k = 1, 2\). It is easy to see that any 2-partition \((A_1, A_2)\) produced by \(G\) satisfies \(|\sigma(A_1) - \sigma(A_2)| \leq \sigma(\pi(1))\). Consider any 2-partition \((B_1, B_2)\) of \(N - \{\pi(1)\}\) with the weights \(\{\sigma(1), \sigma(2), \ldots, \sigma(n)\} - \{\sigma(\pi(1))\}\). If we add \(\sigma(\pi(1))\) to \(B_2\) if \(\sigma(B_1) \leq \sigma(B_2)\) and to \(B_1\), otherwise, then we obtain a 2-partition \((C_1, C_2)\) for the original problem with \(|\sigma(C_1) - \sigma(C_2)| \geq \sigma(\pi(1))\). Since the number of 2-partitions \((C_1, C_2)\) constructed as above is at least half of all 2-partitions, the domination ratio of \(G\) is at least \(\frac{1}{2}\).

Now we turn to the general MMSP. Notice that the size \(s\) of an instance of MMSP is \(\Theta(n + \sum_{i=1}^{n} \log \sigma(i))\). Consider the following approximation algorithm \(H\) for MMSP. If \(s \geq p^n\), then we simply solve the problem optimally, [10]. If \(s < p^n\), then sort the elements of the sequence \(\sigma(1), \sigma(2), \ldots, \sigma(n)\). For simplicity of notation, assume that \(\sigma(1) \geq \sigma(2) \geq \cdots \geq \sigma(n)\). Compute \(r = \lceil \log n/\log p \rceil\) and solve MMSP for \((\{1, 2, \ldots, r\}, \sigma, p)\) to optimality. Suppose we have obtained a \(p\)-partition \(A\) of \(\{1, 2, \ldots, r\}\). Now for \(i\) from \(r + 1\) to \(n\) add \(i\) to the set \(A_j\) of the current \(p\)-partition \(A\) with smallest \(\sigma(A_j)\).

The following theorem proved in [10] generalizes a similar result for M2SP shown in [2].

**Theorem 3.1** The algorithm \(H\) runs in time \(O(s^2 \log s)\). We have

\[
\lim_{s \to \infty} \text{domr}(H, s) = 1.
\]

We will now define the **weighted Max \(k\)-SAT** problem. We are given a collection of boolean clauses each containing at most \(k\) literals and a weight assigned to each clause. The problem of finding a truth assignment such that the total weight of satisfied clauses is maximized is called the weighted Max \(k\)-SAT problem. If the number of literals in a clause is not restricted, then we call this the **weighted Max SAT** problem.

Recently, Alon, Gutin and Krivelevich [2] proved that some polynomial time algorithms for the weighted Max Cut, weighted Max \(k\)-SAT and several of its generalizations are of domination ratio \(\Omega(1)\). In particular, they showed the following for the **weighted Max Cut** problem, where we are given a complete weighted graph \(G\) and we want to find a bipartition \(V_1, V_2\) of the vertex set \(V\), such that the sum of all weights of the edges across \(V_1\) and \(V_2\) is maximized:

**Theorem 3.2** There exists a linear time, deterministic approximation algorithm for the weighted Max Cut whose domination ratio exceeds \(1/40\).
We mentioned above that there are many ATSP heuristic of domination ratio $\Omega(1/n)$. However, it is not known yet whether there exists a polynomial time ATSP heuristic of domination ratio $\Omega(1)$.

While the unweighted (i.e., all weights are equal 1) max SAT is proved to be $\text{DOM}$-easy in [11], it is an open problem whether the weighted max SAT is $\text{DOM}$-easy or not. Another problem of unknown ‘status’ is the quadratic assignment problem (QAP). In [14], a polynomial time QAP algorithm is considered; it is proved that the algorithm is of domination ratio $\Omega(1/n^2)$ when $n$ is a prime power. The result seems to be non-extendible to every $n$ using the approach of [14].

The Max Clique problem is the problem of finding a maximum complete subgraph $G'$ of a given graph $G$. When looking at the Min Vertex Cover problem, we are given a graph $G$ and we want to find a minimum set of vertices which covers all edges.

Using a very powerful non-approximability result on Max Clique (see [18]) the following theorem was proved in [11].

**Theorem 3.3** The Max Clique and Min Vertex Cover are $\text{DOM}$-hard unless $P=NP$.

DA has been used for several other problems not mentioned above; for example, in [21], two algorithms for the frequency assignment problem are compared using DA. A similar investigation for two generalized ATSP heuristics was carried out in [5].

### 4 Greedy Algorithm

We mentioned above that the nearest neighbor algorithm for ATSP is of domination number 1. The same result holds for the greedy algorithm for ATSP [16]. The last result is extended to a wide class of CO problems in [13].

Recall that by the assignment problem we understand the problem of finding a perfect matching of minimum total weight in a given weighted complete bipartite graph $K_{n,n}$. It turns out that the domination number of the greedy algorithm for the assignment problem is 1.

In this section we prove an extension of the main theorem in [13]. Although our proof is similar to the one in [13], we provide it for illustrative reasons. In the end of this section, we briefly discuss results of a recent paper [4].
An independence system \((E, \mathcal{F})\) 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 \(w\) that assigns a real weight \(w(e)\) to every element \(e \in E\). The weight \(w(S)\) 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.

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

(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 the domination number of greedy for every matroid \((E, \mathcal{F})\) is the number of bases: greedy always finds an optimal solution for the \((E, \mathcal{F})\)-optimization problem. Thus, it might be somewhat surprising to have the following theorem:

**Theorem 4.1** 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 \(B \in \mathcal{F}\), \(B \neq B'\),

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

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 $w(x_i) = iM$ and let $w(x) = 1 + jM$ if $x \not\in B'$, $x \in I(x_1, x_2, \ldots, x_{j-1})$ but $x \not\in I(x_1, x_2, \ldots, x_j)$. Clearly, greedy constructs $B'$ and $w(B') = M(k+1)/2$.

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

Observe that

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

but $y_i \not\in I(x_1, x_2, \ldots, x_{a_i})$. Hence, by (12), $y_i$ lies in $I(x_1, x_2, \ldots, x_j) \cap B$, provided $j \leq a_i - 1$. Thus, $y_i$ is counted $a_i$ times in $\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B|$. Hence,

$$w(B) = \sum_{i=1}^{s} w(y_i) \leq s + M \sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B|$$

$$\leq s + M(k(k+1)/2 - 1) = s - M + w(B'),$$

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

Using Theorem 4.1 it is easy to prove that the greedy algorithm for ATSP, STSP and the assignment problem is of domination number one.

In [13], it is noted that the strict inequality (1) cannot be relaxed to the non-strict one as it is satisfied by some matroids.

Bang-Jensen, Gutin and Yeo [4] considered the $(E, \mathcal{F})$-optimization problems, in which the weights are taken from a finite range. For such problems, the authors of [4] 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 Upper Bounds for TSP Domination Number

In [12], upper bounds were obtained for the cardinality of polynomial time searchable ATSP neighborhoods. It turns out that minor changes in the proofs of [12] lead us to stronger and more general results on the maximum possible domination number of polynomial time ATSP heuristics. This is the topic of this section.
It is realistic to assume that any ATSP algorithm spends at least one unit of time on every arc of the complete digraph $K^*_n$ that it considers. We use this assumption in the rest of this section.

**Theorem 5.1** Let $A$ be an ATSP heuristic that runs in time at most $t(n)$. Then the domination number of $A$ does not exceed $\max_{1 \leq n' \leq n} (t(n)/n')^{n'}$.

**Proof:** Let $D = (K^*_n, w)$ be an instance of ATSP and let $H$ be the tour that $A$ returns, when its input is $D$. Let $DOM(H)$ denote all tours in $D$ which are not lighter than $H$ including $H$ itself. We assume that $D$ is a worst instance for $A$, namely $\text{domn}(A, n) = |DOM(H)|$. Since $A$ 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 $A$ actually examines; observe that $|E| \leq t(n)$ by the assumption above. Let $A(H)$ be the set of arcs in $H$. Let $F$ be the set of arcs in $H$ that are not examined by $A$, and let $G$ denote the set of arcs in $D - A(H)$ that are not examined by $A$.

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 $a \in G$ and $a$ is in a tour $H' \in DOM(H)$. By making a very light (possibly negative), we can ensure that $w(H') < w(H)$, 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

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

where we applied the arithmetic-geometric mean inequality. \hfill \Box

**Corollary 5.2** Let $A$ be an ATSP heuristic that runs in time at most $t(n)$. Then the domination number of $A$ does not exceed $\max\{e^{t(n)/e}, (t(n)/n)^n\}$, where $e$ is the basis of natural logarithms.

**Proof:** Let $U(n) = \max_{1 \leq n' \leq n} (t(n)/n')^{n'}$. By differentiating $f(n') = (t(n)/n')^{n'}$ with respect to $n'$ we can readily obtain that $f(n')$ increases for
1 ≤ n′ ≤ t(n)/e, and decreases for t(n)/e ≤ n′ ≤ n. Thus, if n ≤ 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 ≥ 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 5.2.

**Corollary 5.3** Let A be an ATSP heuristic that runs in time at most t(n).
For t(n) ≥ en, the domination number of A does not exceed (t(n)/n)^n.

Note that the restriction t(n) ≥ en is important since otherwise the
bound of Corollary 5.3 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.

It is proved in [9] that there are O(n)-time ATSP algorithms of domina-
tion number 2^{o(n)}. It follows from the last corollary that this result cannot
be improved.

We finish this section with a result that improves (and somewhat clari-
fies) Theorem 20 in [23]. Our proof is a modification of the proof of Theorem
20 in [23].

**Theorem 5.4** Unless P=NP, there is no polynomial time ATSP algorithm
of domination number at least (n−1)! − [n−n^α]! for any constant α < 1.

**Proof:** Assume that there is a polynomial time algorithm H with domina-
tion number at least (n−1)! − [n−n^α]! for some constant α < 1. Choose
an integer s > 1 such that 1/s < α.

Consider a weighted complete digraph (K_n^*, w). We may assume that
all weights are non-negative as otherwise we may add a large number to
each weight. Choose any pair of distinct vertices u and v in K_n^*. Consider
another complete digraph D on n^s − n vertices, in which all weights are 0
and which is vertex disjoint from K_n^*. Add all possible arcs between K_n^*
and D such that the weights of all arcs coming into u and going out of v
are 0 and the weights of all other arcs are M, where M is larger than n
times the maximum weight in (K_n^*, w). Let the resulting weighted complete
digraph be denoted by K_{n^s}^*, and note that we have now obtained an instance
(K_{n^s}^*, w') of ATSP.

Apply H to (K_{n^s}^*, w') (observe that H is polynomial in n for (K_{n^s}^*, w')).
Notice that there are exactly (n^s − n)! Hamilton cycles in (K_{n^s}^*, w') of weight
L, where L is the weight of a lightest Hamilton \((u, v)\)-path in \(K_n^s\). Each of the \((n^s - n)!\) Hamilton cycles is obviously optimal. Observe that the domination number of \(\mathcal{H}\) on \(K_n^s\) is at least \((n^s - 1)! - \lfloor n^s - (n^s)^\alpha \rfloor!\). However, for sufficiently large \(n\), we have

\[(n^s - 1)! - \lfloor n^s - (n^s)^\alpha \rfloor! \geq (n^s - 1)! - (n^s - n)! + 1\]
as \(n^s \alpha \geq n + 1\) for \(n\) large enough. Thus, a Hamilton cycle produced by \(\mathcal{H}\) is always among the optimal solutions (for \(n\) large enough). This means that we can obtain a lightest Hamilton \((u, v)\)-path in \(K_n^s\) in polynomial time, which is impossible since the lightest Hamilton \((u, v)\)-path problem is a well-known NP-hard problem. We have arrived at a contradiction.

\[\blacklozenge\]

Acknowledgement This research was partially supported by a Leverhulme Foundation grant.

References


