STABILITY OF APPROXIMATION IN DISCRETE OPTIMIZATION

Juraj Hromkovič
Dept. of Computer Science, ETH Zurich
Swiss Federal Institute of Technology
ETH Zentrum RZ F2, CH-8092 Zurich
juraj.hromkovic@inf.ethz.ch

Abstract

One can try to parametrize the set of the instances of an optimization problem and look for in polynomial time achievable approximation ratio with respect to this parametrization. When the approximation ratio grows with the parameter, but is independent of the size of the instances, then we speak about stable approximation algorithms. An interesting point is that there exist stable approximation algorithms for problems like TSP that is not approximable within any polynomial approximation ratio in polynomial time (assuming P is not equal to NP). The investigation of the stability of approximation overcomes in this way the troubles with measuring the complexity and approximation ratio in the worst-case manner, because it may success in partitioning of the set of all input instances of a hard problem into infinite many classes with respect to the hardest of the particular inputs. We believe that approaches like this will become the core of the algorithmics, because they provide a deeper insight in the hardness of specific problems and in many application we are not interested in the worst-case problem hardness, but in the hardness of forthcoming problem instances.

1. Introduction

Immediately after introducing NP-hardness (completeness) [Co71] as a concept for proving intractability of computing problems [Ka72], the following question has been posed: If an optimization problem does not admit an efficiently computable optimal solution, is there a possibility to efficiently compute at least an approximation of the optimal solution? Several researchers [Jo74, Lo75, Chr76, IK75] provided already in the middle of the seventies a positive answer for some optimization problems. It may seem to be a fascinating effect if one jumps from the exponential complexity (a huge inevitable amount of physical work) to the polynomial complexity (tractable amount of physical work) due to a small change in the requirement —instead of an exact optimal solution one forces a solution whose quality differs from the quality of an optimal solution at most by $\varepsilon \cdot 100$ % for some ε. This effect is very strong, especially, if one considers problems for which this approximation concept works for any small ε (see the concept of approximation schemes in [IK75, MPS98, Pa94, BC93, Va03, Hr03]).

There is also another possibility to jump from NP to P. Namely, to consider the subset of inputs with a special, nice property instead of the whole set of inputs for which the problem is well-defined. A nice example is the Travelling Salesman Problem (TSP). TSP is not only NP-hard, but also the search of an approximation solution for TSP is NP-hard for every ε. But if one considers TSP for inputs satisfying the triangle inequality (the so-called **Δ-TSP)**, one can even design an approximation algorithm [Chr76] with the approximation ratio $\varepsilon = \frac{1}{2}$. The situation is still more interesting, if one considers the Euclidean TSP, where the distances between the nodes correspond to the distances in the Euclidean metrics. The Euclidean TSP is NP-hard [Pa77], but for every small $\varepsilon > 0$ one can design an **ε-approximation** algorithm [Ar96, Ar97, Mi96] with an almost linear time complexity.

The fascinating observations of huge quantitive changes mentioned above lead us to our proposal to consider the "stability" of approximation algorithms. Let us consider the following scenario. One has an optimization problem P for two sets of inputs L_1 and L_2, $L_1 \subset L_2$. For L_1 there exists an polynomial-time **ε-approximation** algorithm *A*, but for L_2 there is no polynomial-time **δ-approximation** algorithm for any $\delta > 0$ (if NP is not equal to P). We pose the following question: Is the algorithm *A* really useful for inputs from L_1 only? Let us consider a metrics *M* in L_2 determining the distance between any two inputs in L_2. Now, one can consider an input $x \in L_2 - L_1$, for which there exists an $y \in L_1$ such that $distance_M(x, y) \leq k$ for some positive real k. One can look for how "good" the algorithm *A* is for the input $x \in L_2 - L_1$. If for every $k > 0$ and every x with the distance at most k to L_1, *A* computes an $\delta_{\varepsilon,k}$ approximation of an optimal solution for x ($\delta_{\varepsilon,k}$ is considered to be a constant depending on k and ε only), then one can say that *A* is "(approximation) stable" according to the metrics *M*.

The idea of this concept is similar to that of the stability of numerical algorithms. But instead of observing the size of the change of the output value according to a small change of the input value, we look for the size of the change of the approximation ratio according to a small change in the specification (some parameters, characteristics) of the set of problem instances considered. If the exchange of the approximation ratio is small for every small change in the specification of the set of problem instances, then we have a stable algorithm. If a small change in the specification of the set of problem instances causes an essential (depending on the size of the input instances) increase of the relative error, then the algorithm is unstable.

The concept of stability enables us to show positive results extending the applicability of known approximation algorithms. As we shall see later, the concept also motivates to modify an unstable algorithm A in order to get a stable algorithm B that achieves the same approximation ratio on the original set of problem instances as A has, but B can also be successfully used outside of the original set of problem instances. This concept is useful because there are a lot of problems for which an additional assumption on the "parameters" of the problem instances leads to an essential decrease in the hardness of the problem. Such effects are the starting points for trying to partition the whole set of problem instances into a spectrum of classes according to polynomial-time approximability.

As one can observe this approach is similar to the concept of parametrized complexity of Downey and Fellows [DF95, DF99] in trying to overcome the troubles caused by measuring complexity and approximation ratio in the worst-case manner. The main aim of both concepts is partitioning of the set of all instances of a hard problem into infinite many classes with respect to the hardness of particular instances. We believe that approaches like these will be the core of future algorithmics, because they provide a deeper insight in the nature of the hardness of specific problems and in many applications we are not interested in the worst-case problem hardness, but in the hardness of forthcoming problem instances.

2. Definition of the Stability of Approximation Algorithms

We assume that the reader is familiar with the basic concepts and notions of algorithmics and complexity theory as presented in standard textbooks like [BC93, GJ79, Ho96, Pa94, We93, Hr04]. Next, we give a formal definition of the notion of an optimization problem. Let $\mathbb{N} = \{0, 1, 2, ...\}$ be the set of nonnegative integers, and let $\mathbb{R}^+$ be the set of positive reals.

DEFINITION 1 *An* ***optimization problem*** *U is an 7-tuple* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$, *where*

(i) Σ_I *is an alphabet called* ***input alphabet,***

(ii) Σ_O *is an alphabet called* ***output alphabet,***

(iii) $L \subseteq \Sigma_I^*$ *is a language over* Σ_I *called the* ***language of consistent inputs,***

(iv) $L_I \subseteq L$ *is a language over* Σ_I *called the* ***language ofactual inputs,***

(v) $\mathcal{M}$ *is a function from* $2^{\Sigma_O^*}$, *L towhere, for every* $x \in L$, $\mathcal{M}(x)$ *is called the* ***set of feasible solutions*** *for the input* x,

(vi) *cost is a function, called* ***cost function,*** *that for every pair* (u, x), *where* $u \in \mathcal{M}(x)$ *for some* $x \in L$, *assigns a positive real number* $cost(u, x)$,

(vii) *goal* $\in$ {*minimum, maximum*}.

For every $x \in L$, *we define*

$$\boldsymbol{Output_U(x) = \{y \in \mathcal{M}(x) | cost(y) = goal\{cost(z) | z \in \mathcal{M}(x)\}\}}$$

as the set of optimal solutions, and $\boldsymbol{Opt_U(x)= cost(y)}$ *for some* $y \in Output_U(x)$.

Clearly, the meaning for Σ_I, Σ_O, $\mathcal{M}$, *cost* and *goal* is the usual one. *L* may be considered as a set of consistent inputs, i.e., the inputs for which the optimization problem is consistently defined. L_I is the set of inputs considered and only these inputs are taken into account when one determines the complexity of the optimization problem *U*. This kind of definition is useful for considering the complexity of optimization problems parametrized according to their languages of actual inputs. In what follows ***Language*(*U*)** denotes the language L_I of actual inputs of *U*.

DEFINITION 2 *Let* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *be an optimization problem. We say that an algorithm A is a* ***consistent algorithm for U*** *if, for every input* $x \in L_I$, *A computes an output* $A(x) \in \mathcal{M}(x)$. *We say that A* ***solves U*** *if, for every* $x \in L_I$, *A computes an output* $A(x)$ *from* $Output_U(x)$. *The time complexity of A is defined as the function*

$$Time_A(n) = \max\{Time_A(x) \mid x \in L_I \cap \Sigma_I^n\}$$

from $\mathbb{N}$ *to* $\mathbb{N}$, *where* $Time_A(x)$ *is the length of the computation of A on* x.

DEFINITION 3 *Let* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *be an optimization problem, and let A be a consistent algorithm for U.*

For every $x \in L_I$, *the approximation ratio* $\boldsymbol{R_A(x)}$ *of A* ***on x*** *is defined as*

$$R_A(x) = \max\left\{\frac{cost(A(x))}{Opt_U(x)}, \frac{Opt_U(x)}{cost(A(x))}\right\}.$$

For any $n \in \mathbb{N}$, *we define the* ***approximation ration of A*** *as*

$$R_A(n) = \max\{R_A(x) \mid x \in L_I \cap (\Sigma_I)^n\}.$$

For any positive real δ, *we say that A is an* ***δ-approximation algorithm for U*** *if* $R_A(x) \leq \delta$ *for every* $x \in L_I$.

For every function $f : \mathbb{N} \to \mathbb{R}$, *we say that A is a* ***f(n)-approximation algorithm for U*** *if* $R_A(n) \leq f(n)$ *for every* $n \in \mathbb{N}$.

In order to define the notion of stability of approximation algorithms we need to consider something like a distance between a language L and a word outside L.

DEFINITION 4 *Let* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *and* $\overline{U} = (\Sigma_I, \Sigma_O, L, L, \mathcal{M}, cost, goal)$ *be two optimization problems with* $L_I \subset L$. *A* ***distance function for U according to*** L_I *is any function* $h_L : L \to \mathbb{R}^+$ *satisfying the property*

$$h_L(x) = 0 \textit{ for every } x \in L_I.$$

We define, for any $r \in \mathbb{R}^+$,

$$Ball_{r,h}(L_I) = \{w \in L \mid h(w) \leq r\}.$$

Let A be a consistent algorithm for $\overline{U}$, *and let A be an* ε***-approximation*** *algorithm for U for some* $\varepsilon \in \mathbb{R}^+$. *Let* p *be a positive real. We say that A is* ***p-stable according to h*** *if, for every real* $0 \leq r \leq p$, *there exists a* $\delta_{r,\varepsilon} \in \mathbb{R}^+$ *such that A is an* $\delta_{r,\varepsilon}$***-approximation*** *algorithm for* $U_r = (\Sigma_I, \Sigma_O, L, Ball_{r,h}(L_I), \mathcal{M}, cost, goal)$.[1]

A is ***stable according to h*** *if A is* ***p-stable*** *according to h for every* $p \in \mathbb{R}^+$. *We say that A is* ***unstable according to h*** *if A is not* ***p-stable*** *for any* $p > 0$.

For every positive integer r, *and every function* $f_r : \mathbb{N} \to \mathbb{R}^+$ *we say that A is* $(r, f(n))$***-quasistable according to h*** *if A is an* $f_r(n)$***-approximation*** *algorithm for* $U_r = (\Sigma_I, \Sigma_O, L, Ball_{r,h}(L_I), \mathcal{M}, cost, goal)$.

One may see that the notion of stability can be useful for answering the question how broadly a given approximation algorithm is applicable. If one is interested in negative results then one can try to show that for any reasonable distance measure the considered algorithm cannot be extended to work for a much larger set of inputs than the original one. In this way one can search for some more exact boundaries between polynomial approximability and polynomial non-approximability.

3. Examples

We consider the well-known TSP problem that is in its general form very hard for approximation. But if one considers complete graphs in which the triangle inequality holds, then we have a 1.5-approximation algorithm due to Christofides [Chr76]. The idea of this algorithm can be shortly described as follows.

[1]Note, that $\delta_{r,\varepsilon}$ is a constant depending on r and ε only.

CHRISTOFIDES ALGORITHM

```
Input:   A complete graph $G = (V, E)$, and a cost function $c : E \to \mathbb{N}^+$ satisfying the triangle inequality.
Step 1:  Construct a minimal spanning tree $T$ of $G$ according to $c$.
Step 2:  $S := \{v \in V \mid deg_T(v)$ is odd$\}$.
Step 3:  Compute a minimum-weight perfect matching $M$ on $S$ in $G$.
Step 4:  Create the multigraph $G' = (V, E(T) \cup M)$ and construct an Eulerian tour $\omega$ in $G'$.
Step 5:  Construct a Hamiltonian tour $H$ of $G$ by shortening $\omega$ (i.e., by removing all repetitions of the occurrences of every vertex in $\omega$ in one run via $\omega$ from the left to the right).
Output:  $H$.
```

Since the triangle inequality holds and Step 5 is executed by repeatedly shortening a path $x, u_1, ..., u_m, y$ by the edge $\{x, y\}$ (because $u_1, ..., u_m$ have already occured before in the prefix of ω) the cost of H is at most the cost of ω. Thus, the crucial point for the success of Christofides algorithm is the triangle inequality. A reasonable possibility to search for an extension of the application of this algorithm is to look for inputs that "almost" satisfy the triangle inequality. In what follows we do it in two different ways.

Let $\Delta - TSP = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, minimum)$ be a representation of the TSP with the triangle inequality. We may assume $\Sigma_I = \Sigma_O = \{0, 1, \#\}$, L contains codes of all weight functions for edges of complete graphs, and L_I contains codes of weight functions that satisfy the triangle inequality. Let, for every $x \in L$, $G_x = (V_x, E_x, weight_x)$ be the complete weighted graph coded by x. Obviously, the Christofides algorithm is consistent for $(\Sigma_I, \Sigma_O, L, L, \mathcal{M}, cost, minimum)$.

We define for every $x \in L$,

$$dist(x) =$$
$$\max\left\{0, \max\left\{\frac{weight(\{u,v\})}{weight(\{u,p\}) + weight(\{p,v\})} - 1 \,\middle|\, u, v, p \in V_x\right\}\right\}$$

For the simplicity we consider the size of x as the number of nodes of G_x instead of $|x|$.

We observe that $dist(G, c) \le r$ implies the so-called $(1+r)$-**relaxed** triangle inequality

$$c(\{u,v\}) \le (1+r)[c(\{u,w\}) + c(\{w,v\})]$$

for all three different vertices $u, v, w \in V(G)$.

Let, for every positive real number r,

$$\Delta\text{-}TSP_r = (\Sigma_I, \Sigma_O, L, Ball_{r,dist}(L_\Delta), \mathcal{M}, cost, minimum).$$

The next results show that the CHRISTOFIDES ALGORITHM assures only a very weak approximation for instances of $\Delta\text{-}TSP_r$ for any $r \in \mathbb{R}^+$. First, we show a partially positive result and then we prove that it cannot be essentially improved.

LEMMA 5 *For every positive real number r,* CHRISTOFIDES ALGORITHM *is $(r, O(n^{\log_2((1+r)^2)}))$-**quasistable** for dist.*

Proof. Let $I = (G, c) \in Ball_{r,dist}(L_\Delta)$ for an $r \in \mathbb{R}^+$. Let T_I be the minimal spanning tree constructed in Step 1. Let ω_I be the Eulerian tour constructed in Step 4 and $H_I = v_1, v_2, v_3, \ldots, v_n, v_{n+1}$, where $v_{n+1} = v_1$, be the Hamiltonian tour constructed by shortening ω_I in Step 5.

Clearly,

$$\omega_I = v_1, P_1, v_2, P_2, v_3, \ldots, v_n, P_n, v_{n+1},$$

where P_i is a path between v_i and v_{i+1} for $i = 1, 2, \ldots, n$. To exchange a path v, P, u of a length m, $m \in \mathbb{N}^+$, for the edge $\{v, u\}$ we proceed as follows. For any $p, s, t \in V(G)$, one can exchange the path p, s, t for the edge $\{p, t\}$ by the cost increase bounded by the multiplicative constant $(1 + r)$. This means that reducing the length m of a path to the length $\lceil m/2 \rceil$ increases the cost of the connection between u and v by at most $(1 + r)$ times. After at most $\lceil \log_2 m \rceil$ such reduction steps one reduces the path v, P, u of length m to the path v, u, and

$$cost(u, v) = c(\{v, u\}) \leq (1 + r)^{\lceil \log_2 m \rceil} \cdot cost(v, P, u). \qquad (1)$$

Let M_I be the matching constructed in Step 3. Following the analysis of the Christofides algorithm (see Theorem 4.3.5.5 in [Hr03] for instance) we get from(1)

$$cost(M_I) \leq \frac{1}{2} \cdot (1 + r)^{\lceil \log_2 n \rceil} \cdot cost(H_{Opt}), \qquad (2)$$

and

$$cost(H_I) \leq (1 + r)^{\lceil \log_2 n \rceil} cost(\omega_I). \qquad (3)$$

Thus,

$$\begin{aligned}
cost(H_I) &\leq (1 + r)^{\lceil \log_2 n \rceil} cost(\omega_I) = (1 + r)^{\lceil \log_2 \rceil} \left[cost(T_I) + cost(M_I)\right] \\
&\leq (1 + r)^{\lceil \log_2 n \rceil} \left[cost(H_{Opt}) + \frac{1}{2}(1 + r)^{\lceil \log_2 n \rceil} \cdot cost(H_{Opt})\right] \\
&= (1 + r)^{\lceil \log_2 n \rceil} \left(1 + \frac{1}{2}(1 + r)^{\lceil \log_2 n \rceil}\right) \cdot cost(H_{Opt}) \\
&= O\left(n^{\log_2((1+r)^2)} \cdot cost(H_{Opt})\right).
\end{aligned}$$

Now we show that the result of Lemma 5 cannot be essentially improved. To show this, we construct an input for which the CHRISTOFIDES ALGORITHM provides a very poor approximation.

We construct a weighted complete graph from $Ball_{r,dist}(L_\Delta)$ as follows (Figure 1). We start with the path $p_0, p_1, \ldots, p_n$ for $n = 2^k$, $k \in \mathbb{N}$, where every edge $\{p_i, p_{i+1}\}$ has weight 1. Then we add edges $\{p_i, p_{i+2}\}$ for $i = 0, 1, \ldots, n-2$ with weight $2 \cdot (1+r)$. Generally, for every $m \in \{1, \ldots, \log_2 n\}$, we define $weight(\{p_i, p_{i+2^m}\}) = 2^m \cdot (1+r)^m$ for $i = 0, \ldots, n - 2^m$. For all other edges one can take maximal possible weights in such a way that the constructed input is in $Ball_{r,dist}(L_I)$.

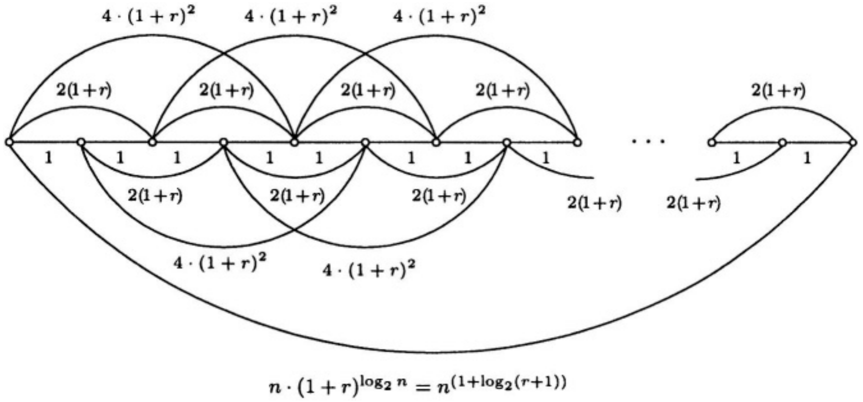

Figure 1.

Let us have a look on the work of the CHRISTOFIDES ALGORITHM on the input (*G, weight*). There is only one minimal spanning tree that corresponds to the path containing all edges of weight 1 (Figure 1). Since every path contains exactly two vertices of odd degree, the Eulerian graph constructed in Step 4 is the cycle $D = p_0, p_1, p_2, \ldots, p_n, p_0$ with the n edges of weight 1 and the edge of the maximal weight $n \cdot (1+r)^{\log_2 n} = n^{1+\log_2(1+r)}$. Since the Eulerian tour is a Hamiltonian tour (Figure 1), the output of the CHRISTOFIDES ALGORITHM is unambiguously the cycle $p_0, p_1, \ldots, p_n, p_0$ with cost $n + n(1+r)^{\log_2 n}$. The optimal tour for this input is $H_{Opt} =$

$$p_0, p_2, p_4, \ldots, p_{2i}, p_{2(i+1)}, \ldots, p_n, p_{n-1}, p_{n-3}, \ldots, p_{2i+1}, p_{2i-1}, \ldots, p_3, p_1, p_0.$$

This tour contains two edges $\{p_0, p_1\}$ and $\{p_{n-1}, p_n\}$ of weight 1 and all $n-2$ edges of weight $2\cdot(1+r)$. Thus, $cost(H_{Opt}) = 2 + 2\cdot(1+r)\cdot(n-2)$ and

$$\frac{cost(D)}{cost(H_{Opt})} = \frac{n + n\cdot(1+r)^{\log_2 n}}{2+2\cdot(1+r)\cdot(n-2)} \geq \frac{n^{1+\log_2(1+r)}}{2n\cdot(1+r)} = \frac{n^{\log_2(1+r)}}{2(1+r)}.$$

Thus, we have proved the following result.

LEMMA 6 *For every* $r \in \mathbb{R}^+$, *if the* CHRISTOFIDES ALGORITHM *is* $(r, f_r(n))$-*quasistable for dist, then*

$$f_r(n) \geq n^{\log_2(1+r)}/(2\cdot(1+r)).$$

COROLLARY 7 *The* CHRISTOFIDES ALGORITHM *is unstable for dist.*

The key question is whether one can modify the Christofides algorithm to get an algorithm that is stable according to *dist.* In what follows, we give a positive answer to this question.

As we have observed, the main problem is that shortening a path $u_1, u_2, \ldots, u_{m+1}$ to the edge u_1, u_{m+1} can lead to

$$cost(\{u_1, u_{m+1}\}) = (1+r)^{\lceil \log_2 m \rceil} \cdot cost(u_1, u_2, \ldots, u_{m+1}).$$

This can increase the cost of the constructed Hamiltonian path by the multiplicative factor $(1+r)^{\lceil \log_2 n \rceil}$ in the comparison with the cost of the Eulerian tour. The rough idea, then, is to construct a Hamiltonian tour by shortening only short paths of the minimal spanning tree constructed in Step 1 of the algorithm.

To realize this idea we shall prove that, for every tree $T = (V, E)$, the graph $T^3 = (V, \{\{x, y\} \mid x, y \in V,$ there is a path x, P, y in T of a length at most 3$\})$ contains a Hamiltonian tour H. This means that every edge $\{u, v\}$ of H has a corresponding unique path $u, P_{u,v}, v$ in T of a length at most 3. This is a positive development, but it still does not suffice for our purposes. The remaining problem is that we need to estimate a good upper bound on the cost of the path $P(H) = u_1, P_{u_1,u_2}, u_2, P_{u_2,u_3}, u_3, \ldots, u_{n-1}P_{u_{n-1},u_n}, u_n, P_{u_n,u_1}, u_1$ (in T) that corresponds to the Hamiltonian tour $u_1, u_2, \ldots, u_n, u_1$ in T^3. Note that in the naive 2-approximation algorithm the resulting Hamiltonian tour can be viewed as a shortening of the Eulerian tour[2] with a cost at most twice of the cost of T. But, we do not know the frequency of the occurrences of particular edges of T in $P(H)$. It may happen that the most expensive edges of T occur more frequently in $P(H)$ than the cheap edges. Observe also that $cost(T^3)$ cannot be bounded by $c \cdot cost(T)$ for any constant c independent on

[2]The Eulerian tour uses every edge of T exactly twice.

T, because T^3 may be even a complete graph for some trees T. Thus, we need the following technical lemma proving that T^3 contains a Hamiltonian tour H such that each edge of T occurs at most twice in $P(H)$.

DEFINITION 8 *Let T be a tree. For every edge $\{u, v\} \in E(T)$, let $u, P_{u,v}, v$ be the unique simple path between u and v in T.*

*Let k be a positive integer. Let $U = u_1, u_2, \ldots, u_m$ be any simple path in T^k. Then, we define the U-**path in** T as*

$$P_T(U) = u_1, P_{u_1,u_2}, u_2, P_{u_2,u_3}, \ldots, u_{m-1}, P_{u_{m-1},u_m}, u_m.$$

LEMMA 9 *Let T be a tree with $n \geq 3$ vertices, and let $\{p, q\}$ be an edge of T. Then, T^3 contains a Hamiltonian path $U = v_1, v_2, \ldots, v_n$, $p = v_1$, $v_n = q$, such that every edge of $E(T)$ occurs exactly twice in $P_T(H)$, where $H = U, p$ is a Hamiltonian tour in T^3.*

Proof. We prove this assertion by induction on the number of vertices of T.

(1) Let $n = 3$. The only tree of three vertices is

$$T = (\{v_1, v_2, v_3\}, \{\{v_1, v_2\}, \{v_2, v_3\}\})$$

and the corresponding T^3 is the complete graph of three vertices

$$(\{v_1, v_2, v_3\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}\}).$$

Thus, the only Hamiltonian tour in T^3 is v_1, v_2, v_3, v_1. The claim of Lemma 9 is true, since $P_T(v_1, v_2, v_3, v_4) = v_1, v_2, v_3, v_2, v_1$.

(2) Let $n \geq 4$ and assume that Lemma 9 is true for trees with fewer than n vertices. Let $T = (V, E)$ be a tree, $|V| = n$. Let $\{p, q\}$ be an arbitrary edge of T. Consider the graph $T' = (V, E - \{\{p, q\}\})$ that consists of two trees T_p and T_q, where T_p $[T_q]$ is the component of T' containing the vertex p $[q]$. Obviously, $|V(T_p)| \leq n - 1$ and $|V(T_q)| \leq n - 1$. Let p' and q', respectively, be a neighbor of p and q, if any, in T_p and T_q, respectively. Now, we fix some Hamiltonian paths U_p and U_q in T_p^3 and T_q^3, respectively. To do it, we distinguish three possibilities according to the cardinalities of T_p and T_q.

1 If $|V(T_p)| = 1$, then set $U_p = p = p'$.

2 If $|V(T_p)| = 2$, then set $U_p = p, p'$.

3 If $3 \leq |V(T_p)| \leq n - 1$, then we can apply the induction hypothesis. We set U_p to be a Hamiltonian path from p to p' in T_p^3 such that $P(U_p, p)$ contains every edge of T_p exactly twice.

A Hamiltonian path U_q in T_q^3 can be fixed in the same way as U_p was fixed above (Figure 2).

Now, consider the path U_p, U_q^R obtained by connecting U_p and the reverse of U_q by the edge $\{p', q'\}$. Observe, that $\{p', q'\} \in T^3$, because p', p, q, q' is a path in T. Following Figure 2, it is obvious that U_p, U_q^R is a Hamiltonian path in T^3, and that U_p, U_q^R, p is a Hamiltonian tour in T^3.

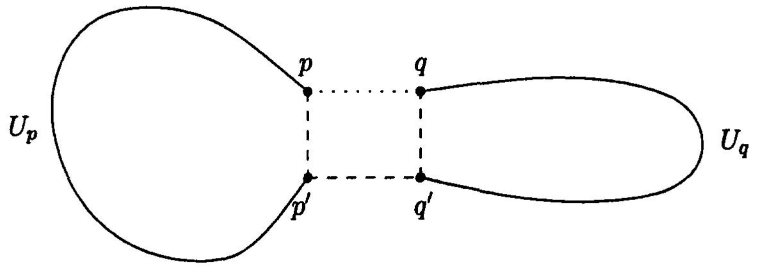

Figure 2.

Observe (by the induction hypothesis or the trivial cases with $|V(T_p)| \leq 2$) that $P_{T_p}(U_p, p')$ the Hamiltonian tour U_p, p' in T^3 contains every edge of T_p exactly twice. Thus, $P_{T_p}(U_p)$ contains every edge, but the edge $\{p, p'\}$ of T_p exactly twice. The edge $\{p, p'\}$ is contained exactly once in $P_{T_p}(U_p)$. Similarly, $P_{T_q}(U_q)$ contains every edge of T_q twice, but the edge $\{q, q'\}$ once. Finally, $P_T(U_p, U_q^R, p)$ contains every edge of T exactly twice, because

1 this is clear from the properties of U_p and U_q^R for every edge from $E - \{\{p, q\}, \{p, p'\}, \{q, q'\}\}$,

2 the edge $\{p', q'\} \in T^3$ connecting U_p and U_q (Figure 2) is realized by the path p', p, q, q' containing edges $\{p, p'\}$, $\{p, q\}$, and $\{q, q'\}$ of E, and

3 the connection of U_p, U_q^R with p is realized directly by the edge $\{p, q\}$.

SEKANINA'S ALGORITHM

Input: A complete graph $G = (V, E)$, and a cost function $c : E \rightarrow \mathbb{N}^+$.

Step 1: Construct a minimal spanning tree T of G according to c.

Step 2: Construct T^3.

Step 3: Find a Hamiltonian tour H in T^3 such that $P_T(H)$ contains every edge of T exactly twice.

Output: H.

THEOREM 10 SEKANINA'S ALGORITHM *is a polynomial-time* 2*-approximation algorithm for* Δ*-TSP.*

Proof. Obviously, Step 1 and 2 of SEKANINA'S ALGORITHM can be performed in time $O(n^2)$. Using Lemma 9 one can implement Step 3 in time $O(n)$. Thus, the time complexity of SEKANINA'S ALGORITHM is in $O(n^2)$.

Let H_{Opt} be an optimal solution for an input instance (G, c) of Δ-TSP. We have $cost(T) \leq cost(H_{Opt})$. The output H of SEKANINA'S ALGORITHM can be viewed as shortening the path $P_T(H)$ by removing repetitions of vertices in $P_T(H)$. Since $P_T(H)$ contains every edge of T exactly twice,

$$cost(P_T(H)) = 2 \cdot cost(T) \leq 2 \cdot cost(H_{Opt}). \qquad (4)$$

Since H is obtained from $P_T(H)$ by exchanging simple subpaths by an edge, and c satisfies the triangle inequality,

$$cost(H) \leq cost(P_T(H)). \qquad (5)$$

Combining (4) and (5) we obtain $cost(H) \leq 2 \cdot cost(H_{Opt})$.

THEOREM 11 *For every positive real number* r, SEKANINA'S ALGORITHM *is a polynomial-time* $2(1+r)^2$*-approximation algorithm for* Δ-TSP_r.

Proof. Since SEKANINA'S ALGORITHM always outputs a Hamiltonian tour, it is consistent for TSP. Obviously, the inequality (4) is also true for any input instance of the general TSP.

Let (G, c) be an input instance of Δ-TSP$_r$. Since $(G, c) \in Ball_{r,dist}(L_\Delta)$,

$$\begin{aligned} c(\{v_1, v_4\}) &\leq (1+r)^2 \cdot cost(v_1, v_2, v_3, v_4), \text{ and} \\ c(\{u_1, u_3\}) &\leq (1+r) \cdot cost(u_1, u_2, u_3) \end{aligned}$$

for all edges $\{u_1, u_3\}, \{v_1, v_4\} \in E(G)$ and every path v_1, v_2, v_3, v_4 between v_1 and v_4 and every path u_1, u_2, u_3 between u_1 and u_3. Since H is obtained from $P_T(H)$ by exchanging a simple subpath of $P_T(H)$ of length at most 3,

$$cost(H) \leq (1+r)^2 \cdot cost(P_T(H)). \qquad (6)$$

Combining (4) and (6) we finally obtain

$$cost(H) \leq 2 \cdot (1+r)^2 \cdot cost(P_T(H)).$$

COROLLARY 12 SEKANINA'S ALGORITHM *is stable according to dist.*

Thus, we have reached our final aim to divide the set of all instances of TSP into an infinite spectrum in such a way that the sets of this spectrum have upper bounds on the polynomial-time approximability of their input instances. The above analysis of TSP shows that it is reasonable to measure the hardness of the TSP instances by the distance function *dist,* i.e., by the degree of violation of the triangle inequality.

4. Conclusion and an Overview

In the previous sections we have introduced the concept of stability of approximations. Here we discuss the potential applicability and usefulness of this concept.

Using this concept, one can establish positive results of the following types:

1 An approximation algorithm or a PTAS can be successfully used for a larger set of inputs than the set usually considered.

2 We are not able to successfully apply a given approximation algorithm A (a PTAS) for additional inputs, but one can simply modify A to get a new approximation algorithm (a new PTAS) working for a larger set of inputs than the set of inputs of A.

3 To learn that an approximation algorithm is unstable for a distance measure could lead to the development of completely new approximation algorithms that would be stable according to the considered distance measure.

The following types of negative results may be achieved:

4. The fact that an approximation algorithm is unstable according to all "reasonable" distance measures and so that its use is really restricted to the original input set.

5. Let $Q = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal) \in NPO$ be well approximable. If, for a distance measure D and a constant r, one proves the nonexistence of any polynomial-time approximation algorithm for $Q_{r,D} = (\Sigma_I, \Sigma_O, L, Ball_{r,D}(L_I), \mathcal{M}, cost, goal)$, then this means that the problem Q is "unstable" according to D.

Thus, using the notion of stability one can search for a spectrum of the hardness of a problem according to the set of inputs. For instance, considering a hard problem like TSP or Clique Problem one could get an infinite sequence of input languages $L_0, L_1, L_2, \ldots$ given by some distance measure, where $R_r(n)$

is the best achievable approximation ratio for the language L_r. Results of this kind can essentially contribute to the study of the nature of hardness of specific problems.

The best results known for TSP instances satisfying the $\beta = (1+r)$-triangle inequality are the following ones:

1 Andreae and Bandelt [AB95] showed that the here presented Sekanina Algorithm provides a $(\beta^2 + \beta)$ approximation ratio, which is the best known for $2 \leq \beta \leq 3$;

2 Bender and Chekuri [BCh99] designed a $4 \cdot \beta$-approximation algorithm, which is the best for $\beta > 3$;

3 Böckenhauer at. al. [BHK+02] have modified the Christofides Algorithm in order to get a $\frac{3}{2} \cdot \beta^2$-approximation algorithm, which is the best for $1 < \beta < 2$.

Moreover Bender and Chekuri [BCh99] proved a lower bound on the polynomial time approximability of this TSP subproblem which grows linearly with β.

Further development of these ideas for different versions of the Hamiltonian path problem can be found by Forlizzi at. al. [FHP+04], where a few stable algorithms with respect to relaxed triangle inequality were designed.

Another possibility is to consider the so-called **α-strengthen triangle inequality,** where one requires

$$c(\{u,v\}) \leq \alpha \cdot [c(\{u,w\}) + c(\{w,v\})]$$

for an α with $1 > \alpha \geq 1/2$. Observe that for $\alpha = 1/2$ all edges have the same weight and so the problem becames trivial. Böckenhauer at. al. [BHK+00] designed three algorithms for TSP subproblems with instances satisfying the α-strengthen triangle inequality, which yield the approximation ratios starting with 1 for $\alpha = 1/2$ and growing with α to 3/2 for $\alpha = 1$. A very strong result has been proved by Böckenhauer and Seibert [BS00] who established an explicit lower bound on polynomial time approximability of TSP with sharped triangle inequality for any $\alpha > 1/2$ and this lower bounds grows with α. Thus, the TSP instances with weights from the interval $[1, 1 + \varepsilon]$ form an APX-hard problem for arbitrary small $\varepsilon > 0$. The subproblems with sharped triangle inequality were also successfully attacked for the minimum 2-connected spanning subgraph problems in [BBH+02].

[Ar96] S. Arora: Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In: *Proc. 37th IEEE FOCS,* IEEE 1996, pp. 2–11.

[Ar97] S. Arora: Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. In: *Proc. 38th IEEE FOCS,* IEEE 1997, pp. 554–563.

[BC93] D. P. Bovet, C. Crescenzi: *Introduction to the Theory ofComplexity,* Prentice-Hall 1993.

[Chr76] N. Christofides: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsbourgh, 1976.

[Co71] S. A. Cook: The complexity of theorem proving procedures. In: Proc *3rd ACM STOC,* ACM 1971, pp. 151–158.

[GJ79] M. R. Garey, D. S. Johnson: *Computers and Intractibility. A Guide to the Theory on NP-Completeness.* W. H. Freeman and Company, 1979.

[Ho96] D. S. Hochbaum (Ed.): *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company 1996.

[IK75] O. H. Ibarra, C. E. Kim: Fast approximation algorithms for the knapsack and sum of subsets problem. *J. of the ACM* 22 (1975), pp. 463–468.

[Jo74] D. S. Johnson: Approximation algorithms for combinatorial problems *JCSS* 9 (1974), pp. 256–278.

[Ka72] R. M. Karp: Reducibility among combinatorial problems. In: R. E. Miller, J.W. Thatcher (eds.): *Complexity ofComputer Computations,* Plenum Press 1972, pp. 85–103.

[Lo75] L.Lovasz: On the ratio of the optimal integral and functional covers. *Discrete Mathematics* 13 (1975), pp. 383–390.

[Mi96] I. S. B. Mitchell: Guillotine subdivisions approximate polygonal subdivisions: Part II — a simple polynomial-time approximation scheme for geometric **k-MST,** TSP and related problems. Technical Report, Dept. of Applied Mathematics and Statistics, Stony Brook 1996.

[MPS98] E. W. Mayr, H. J. Promel, A. Steger (Eds.): *Lecture on Proof Verification and Approximation Algorithms. Lecture Notes in Computer Science* 1967, Springer 1998.

[Pa77] Ch. Papadimitriou: The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science* 4 (1977), pp. 237–244.

[Pa94] Ch. Papadimitriou: *Computational Complexity,* Addison-Wesley 1994.

[We93] I. Wegener: *Theoretische Informatik: eine algorithmenorientierte Einfuhrung.* B.G. Teubner 1993.

[Hr04] **J. Hromkovič**: *Theoretical Computer Science,* Springer-Verlag 2004.

[BBH$^+$02] H.-J. Böckenhauer, D. Bongartz, **J. Hromkovič,** R. Klasing, G. Proietti, S. Seibert, W. Unger: On the Hardness of constructing mininal 2-connected spanning subgraphs in complete graphs with sharped triangle inequality. In: Proc *FSTTCS?02,* pp.59-70.

[Hr03] **J. Hromkovič:** *Algorithmicsfor Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics.* Springer-Verlag 2003.

[Va03] V. V. Vazirani: Approximation Algorithms. Springer-Verlag 2003.

[DF95] R.G. Downey, M. R. Fellows: Fixed-parameter tractibility and completeness I: Basic Results. *SIAM Journal of Computing.* 24 (1995), pp. 873–921.

[DF99] R.G. Downey, M. R. Fellows: *Parametrized Complexity.* Springer-Verlag 1999.

[FHP$^+$04] L. Forlizzi, **J. Hromkovič,** G. Proietti, S. Seibert: On the stability of approximation for Hamiltonian path problems. Unpublished manuscript.

[BHK$^+$00] H.-J. Böckenhauer, **J. Hromkovič,** R. Klasing, S. Seibert, W. Unger: Approximation algorithms for TSP with sharped triangle inequality. *Information Processing Letters.* 75 (2000), pp. 133-138.

[BS00] H.-J. Böckenhauer, S. Seibert: Improved lower bounds on the approximability of the traveling salesman problem. *Theoretical Informatics and Applications.* 34 (2000), pp. 213-255.

[AB95] T. Andreae, H. J. Bandelt: Performance guarentees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics.* 8 (1), pp. 1-16, February 1995.

[BCh99] M. Bender, C. Chekuri: Performance guarentees for TSP with a parametrized triangle inequality. In: *Proc. 6. International Workshop on Algorithms and Data Structures, WADS?99,* volume 1663 *Lecture Notes in Computer Science.* pp. 1-16, Springer, August 1999.

[BHK$^+$02] H.-J. Böckenhauer, **J. Hromkovič,** R. Klasing, S. Seibert, W. Unger: Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. *Theoretical Computer Science.* 285 (1), pp. 3-24, July 2002.