A Modern View on Stability of Approximation

Ralf Klasing, Tobias Mömke

A Modern View on Stability of Approximation*

Ralf Klasing[1] and Tobias Mömke[2]

[1] CNRS, LaBRI, University of Bordeaux, France ralf.klasing@labri.fr
[2] University of Bremen, Germany moemke@uni-bremen.de

Abstract. In order to attack hard optimization problems that do not admit any polynomial-time approximation scheme (PTAS) or α-approximation algorithm for a reasonable constant α (or even with a worse approximability), Hromkovič et al. [12, 31] introduced the notion of *Stability of Approximation*. The main idea of the stability concept is to try to split the set of all input instances into (potentially infinitely many) classes with respect to the achievable approximation ratio. The crucial point in applying this concept is to find parameters that well capture the difficulty of problem instances of the particular hard problem. The concept of stability of approximation turns out to be ubiquitous in the research of approximation algorithms and is applicable beyond approximation. In the literature, however, the relation to stability of approximation has stayed implicit. The purpose of this survey is to take a broader view and to explicitly analyze algorithmic problems and their algorithms with respect to stability of approximation.

I would like to express my belief that, in order to make essential progress in algorithmics, one has to move from measuring the problem hardness in a worst-case manner to classifying the hardness of the instances of the investigated algorithmic problems.

Juraj Hromkovič [33]

1 Introduction

The study of algorithms investigates for which algorithmic problems a solution with specific properties can be obtained such that the computation does not exceed given resource constraints. The properties of the solutions are determined by the structure of the solution such as for instance "the solution is a tree" or "the solution is a sorted list of numbers." Some common resource measures are time and space (*e.g.*, the algorithm should run in polynomial time and/or space), but there are further reasonable measures such as the amount of randomness, advice, security, non-determinism, etc.

* Research partially funded by Deutsche Forschungsgemeinschaft grant MO2889/1-1, and by "The Investments for the Future" Programme IdEx Bordeaux - CPU (ANR-10-IDEX-03-02).

In this survey, we are not able to cover the broad and deep field of algorithmics as a whole. Instead, we focus our attention on optimization problems. We therefore consider problems where the solutions are associated with a cost or profit, and we want to find a feasible solution (complying with the specified properties) such that the cost is minimized or the profit is maximized.

For simplicity, let us for now concentrate on minimization problems. Let us consider the traveling salesman problem (TSP). We are given a complete graph G with a cost function $\mathsf{cost}\colon E(G) \to \mathbb{R}^+$. The feasible solutions are exactly the Hamiltonian cycles of G, *i.e.*, all sets of edges such that each vertex has degree 2 and the induced subgraph of these edges is connected. The goal is to find a feasible solution (Hamiltonian cycle) of minimum cost.

We would like to find a solution in polynomial time. Since the (general) TSP is known to be NP-hard, unless $\mathsf{P} = \mathsf{NP}$ we cannot guarantee to find an optimal solution. We therefore naturally obtain the notion of approximation algorithms: we do not restrict our view to optimal solutions. Instead we allow a set of feasible solutions that satisfy the following quality constraints: an algorithm A is an α-approximation algorithm if all solutions computed by A are at most a factor α worse than the optimum.

It is well-known that without restricting the cost function, the TSP cannot be approximated in polynomial time.[1] The situation is different if the cost function cost is a metric (in particular, cost satisfies the triangle inequality, *i.e.*, for all distinct vertices $u, v, w \in V(G)$, $\mathsf{cost}(u, w) \leq \mathsf{cost}(u, v) + \mathsf{cost}(v, w)$). In metric graphs, the TSP can be approximated in polynomial time with an approximation ratio $\alpha = 1.5$ using Christofides' algorithm [21]. If we further restrict the costs such that $\mathsf{cost}(u, v) = 1$ for all $u, v \in V(G)$, the problem boils down to computing an *arbitrary* Hamiltonian cycle, which can be done in polynomial time (since G is complete).

The approximability of the TSP reveals a concept that we can see in a broader context: the hardness of a problem (here measured in the achievable approximation ratio) depends on the set of admissible instances. The more we restrict the class of instances, the easier it is to compute a solution to the problem. The dependency of admissible instances and approximation was first formalized by Hromkovič et al. [12, 31], where the authors introduced the notion of *Stability of Approximation*.

The idea behind this concept is to find a parameter (characteristic) of the input instances that captures the hardness of particular inputs. An approximation algorithm is called *stable* with respect to this parameter if its approximation ratio grows with this parameter but not with the size of the input instances.

[1] Since the encoding of numbers contributes to the size of the instance, the approximation ratio of the TSP is bounded by some function. For example, if the instance is an n-vertex graph with edge costs in the order of 2^{2^n}, the encoding of the instance consists of more than 2^n bits which allows for an exponential running time in n. Since these considerations are merely an artefact of the machine model, we consider optimization problems with a super-polynomial lower bound on the approximation ratio as inapproximable.

Note that the idea of the concept of approximation stability is similar to that of stability of numerical algorithms. Instead of observing the size of the change of the output value according to a small change of the input value, one looks for the size of the change of the approximation ratio according to a small change in the specification of the set of consistent input instances.

The concept of stability of approximation turns out to be ubiquitous in research on approximation algorithms and is applicable beyond approximation. In the literature, however, the relation to stability of approximation has stayed implicit. The purpose of this survey is to take a broader view and to explicitly analyze algorithmic problems and their algorithms with respect to stability of approximation.

The survey is organized as follows. In Section 2, we formally introduce the concept of stability of approximation. In Section 3, we study the concept in the context of graph problems where the edge costs form a metric. In Section 4, we investigate stability of approximation in the context of scheduling problems. In Section 5, we consider the problems of vertex coloring and maximum independent set in bounded-degree graphs. In Section 6, we study stability of approximation in the context of parameterized algorithms. In Section 7, we consider graph problems where the vertices are points in some space and the distances are measured according to the properties of the space. In particular, we consider Euclidean space and doubling spaces. In Section 8, we show that the concept of stability of approximation naturally translates to concepts such as online algorithms, by studying online scheduling and online matching. In Section 9, we draw several conclusions from our investigation of stability of approximation.

2 Stability of approximation

We now give the formal definition of the stability of approximation algorithms [12, 14, 31, 32].

In general, an optimization problem U is determined by a set of instances L_I, a set of feasible solutions $\mathcal{M}$ with subsets $\mathcal{M}(x)$ for each $x \in L_I$, a cost function $\mathsf{cost}\colon \mathcal{M} \to \mathbb{R}^+$ and a goal (minimization or maximization).[2] Additionally, we distinguish between instances L_I that we want to allow, and all instances L that are feasible (and thus $L_I \subseteq L$). For instance, when considering the metric TSP, L_I contains all (encodings of) edge-weighted graphs where the cost function is a metric. The definition of the TSP, however, does not depend on the cost restriction. Instead, we could use an arbitrary cost function. Then L is the set of all (encodings of) edge-weighted graphs $G = (V, E)$ with all cost functions $\mathsf{cost}\colon E \to \mathbb{R}^+$.

Let U be an optimization problem. For every $x \in L_I$, we define $Output_U(x) = \{y \in \mathcal{M}(x) \mid \mathsf{cost}(y) = \mathsf{goal}\{\mathsf{cost}(z) \mid z \in \mathcal{M}(x)\}\}$ as the set of optimal solutions for x and U, and $Opt_U(x) = \mathsf{cost}(y)$ for some $y \in Output_U(x)$. We say that an algorithm A is a *consistent algorithm for* U if, for every input $x \in L_I$, A

[2] For simplicity we assume instances and solutions to be encoded over the binary alphabet.

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, and Σ_I is an alphabet called the *input alphabet* of U.

Let U be an optimization problem, and let A be a consistent algorithm for U. For every $x \in L_I$, the *approximation ratio* $R_A(x)$ *of* A *on* x is defined as $R_A(x) = \max\left\{\frac{\mathsf{cost}(A(x))}{Opt_U(x)}, \frac{Opt_U(x)}{\mathsf{cost}(A(x))}\right\}$. For any $n \in \mathbb{N}$, we define the *approximation ratio of* A as $R_A(n) = \max\{R_A(x) \mid x \in L_I \cap \Sigma_I^n\}$. For any positive real $\delta > 1$, we say that A is a *δ-approximation algorithm for* U if $R_A(x) \leq \delta$ for every $x \in L_I$. For every function $f\colon \mathbb{N} \to \mathbb{R}^{>1}$, we say that A is an *$f(n)$-approximation algorithm for* U if $R_A(n) \leq f(n)$ for every $n \in \mathbb{N}$.

Stability of approximation is concerned with the approximation behavior of instances in $L \setminus L_I$. To this end, we consider a distance function $h\colon L \to \mathbb{R}^{\geq 0}$ that expresses how much we deviate from L_I. We require h to be polynomial time computable and that $h(x) = 0$ for all $x \in L_I$.

For each $r \in \mathbb{R}^{\geq 0}$, we consider the ball $B_{r,h}(L_I) := \{x \in L \mid h(x) \leq r\}$. Let A be an algorithm that computes a feasible solution for each $x \in L$ and an α-approximation for each $x \in L_I$ and $\alpha \geq 1$. Then A is called *p-stable according to* h if for each $0 \leq r \leq p$ there is a constant α'_r such that A is an α'_r-approximation algorithm for each $x \in B_{h,r}(L_I)$. We call A *stable according to* h if it is p-stable according to h for all $p \in \mathbb{R}^{\geq 0}$.

If in the definition we replace the constant α'_r by a function $f_r\colon \mathbb{N} \to \mathbb{R}^{\geq 0}$ that maps the length of the encoding of the instance to a number, we call A *f_r-quasistable according to* h.

3 Relaxing the metric

The original context in which stability of approximation was studied was in the context of graph problems where the edge costs form a metric [12, 14, 30, 31].

A complete weighted graph $G = (V, E, \mathsf{cost})$ is called Δ_β-metric, for some $\beta \geq 1/2$, if the cost function $\mathsf{cost}(\cdot, \cdot)$ satisfies $\mathsf{cost}(v, v) = 0$, $\mathsf{cost}(u, v) = \mathsf{cost}(v, u)$, and the β-triangle inequality, *i.e.*, $\mathsf{cost}(u, v) \leq \beta \cdot (\mathsf{cost}(u, x) + \mathsf{cost}(x, v))$ for all vertices $u, v, x \in V$. If $\beta > 1$ then we speak about the *relaxed triangle inequality*, and if $\beta < 1$ we speak about the *sharpened triangle inequality*.

The concept of *stability of approximation* has been successfully applied to several fundamental hard optimization problems. A nice example is the Traveling Salesman Problem (TSP), which does not admit any polynomial-time approximation algorithm with an approximation ratio bounded by a polynomial in the size of the input instance, but is $\frac{3}{2}$-approximable for metric input instances. Here, one can characterize the input instances by their "distance" to metric instances. This can be expressed by the β-triangle inequality for any $\beta \geq 1$.

In a sequence of papers [1, 2, 7, 10–13, 39], it was shown that

1. there are stable approximation algorithms for the TSP whose approximation ratio[3] grows with β, and
2. for every $\beta > \frac{1}{2}$ one can prove explicit lower bounds on the polynomial-time approximability growing with β.

Hence, one can partition the set of all input instances of the Traveling Salesman Problem into infinitely many subclasses according to the degree of violation of the triangle inequality, and for each subclass one can guarantee upper and lower bounds on the approximation ratio (that are linear in β).

Similar studies demonstrated that the β-triangle inequality can serve as a measure of hardness of the input instances for other optimization problems as well. In particular, for the problem of constructing 2-connected spanning subgraphs of a given complete edge-weighted graph, it was proved in [8] (with an explicit lower bound on the approximability) that this minimization problem is APX-hard even for graphs satisfying a sharpened triangle inequality for any $\beta > \frac{1}{2}$, *i.e.*, even if the edge costs are from an interval $[1, 1+\varepsilon]$ for an arbitrarily small $\varepsilon > 0$. On the other hand, an upper bound of $\frac{2}{3} + \frac{1}{3} \cdot \frac{\beta}{1-\beta}$ on the approximability is achieved in [8] for all inputs satisfying the sharpened triangle inequality. For the problem of finding, for a given positive integer $k \geq 2$ and an edge-weighted graph G, a minimum k-edge- or k-vertex-connected spanning subgraph, a linear-time approximation algorithm was given in [9] with approximation ratio $\frac{\beta}{1-\beta}$ for any $\frac{1}{2} \leq \beta < 1$ (which does not depend on k).

Certain problems, such as the star p-hub center problem (Δ_β-SpHCP) [19] and the p-hub center problem (Δ_β-pHCP) [20], exhibit a *phase transition phenomenon* when parameterized by the β-triangle inequality. More precisely, considering the class $\mathcal{C}_\beta$ of Δ_β-metric graphs, there exists β_0 such that these problems are solvable in polynomial time when restricted to $\mathcal{C}_\beta$ for any $\beta \leq \beta_0$, whereas for $\beta > \beta_0$ they are approximable in the class $\mathcal{C}_\beta$ with upper and lower bounds on the approximation ratio depending on β.

More precisely, Chen et al. [19] showed that for an arbitrary $\varepsilon > 0$, to approximate Δ_β-SpHCP with a ratio $g(\beta) - \varepsilon$ is NP-hard, and $r(\beta)$-approximation algorithms were given for the same problem, where $g(\beta)$ and $r(\beta)$ are functions of β. If $\beta \leq \frac{3-\sqrt{3}}{2}$, one has $r(\beta) = g(\beta) = 1$, *i.e.*, Δ_β-SpHCP is polynomial-time solvable. If $\frac{3-\sqrt{3}}{2} < \beta \leq \frac{2}{3}$, one has $r(\beta) = g(\beta) = \frac{1+2\beta-2\beta^2}{4(1-\beta)}$. For $\frac{2}{3} \leq \beta \leq 1$, $r(\beta) = \min\{\frac{1+2\beta-2\beta^2}{4(1-\beta)}, 1 + \frac{4\beta^2}{5\beta+1}\}$. Moreover, for $\beta \geq 1$, one has $r(\beta) = \min\{\beta + \frac{4\beta^2-2\beta}{2+\beta}, 2\beta+1\}$ and $g(\beta) = \beta + \frac{1}{2}$. For $\beta \geq 2$, the approximability of the problem (*i.e.*, upper and lower bound) is linear in β.

Chen et al. [20] proved that for an arbitrary $\varepsilon > 0$, to approximate Δ_β-pHCP with a ratio $\tilde{g}(\beta) - \varepsilon$ is NP-hard, and $\tilde{r}(\beta)$-approximation algorithms were given for the same problem, where $\tilde{g}(\beta)$ and $\tilde{r}(\beta)$ are functions of β. If $\beta \leq \frac{3-\sqrt{3}}{2}$, one has $\tilde{r}(\beta) = \tilde{g}(\beta) = 1$, *i.e.*, Δ_β-pHCP is polynomial-time solvable. If $\frac{3-\sqrt{3}}{2} < \beta \leq \frac{5+\sqrt{5}}{10}$, one has $\tilde{r}(\beta) = \tilde{g}(\beta) = \frac{3\beta-2\beta^2}{3(1-\beta)}$. For $\frac{5+\sqrt{5}}{10} \leq \beta \leq 1$,

[3] The currently best approximation ratio is $\min\{4\beta, 3\beta/4 + 3\beta^2/4\}$ [7, 39].

$\tilde{r}(\beta) = \min\{\beta + \beta^2, \frac{4\beta^2+5\beta+1}{5\beta+1}\}$ and $\tilde{g}(\beta) = \frac{4\beta^2+3\beta-1}{5\beta-1}$. Moreover, for $\beta \geq 1$, one has $\tilde{r}(\beta) = 2\beta$ and $\tilde{g}(\beta) = \beta \cdot \frac{4\beta-1}{3\beta-1}$. For $\beta \geq 1$, the approximability of the problem (*i.e.*, upper and lower bound) is linear in β.

4 Processing Time

We now turn our attention to a basic and well-studied scheduling problem. We want to minimize the flow time on multiple machines. Formally, we are given m identical machines and a set J of n jobs. Each job j has a processing time $1 \leq p_j \leq P$ and release time r_j. Jobs can be preempted, *i.e.*, we can interrupt the processing of a running job and continue later without any penalties. The entire job has to be processed on the same machine and the task of the algorithm is to decide which job is scheduled on which machine.

For a given algorithm, C_j denotes the completion time of a job j, and the flow time of j is defined as $F_j = C_j - r_j$, *i.e.*, the time that has passed between the release time and the completion time. Now, the goal is to minimize the total flow time $\sum_j F_j$.

Leonardi and Raz [37, 38] showed that there is an $O(\log(\min\{\frac{n}{m}, P\}))$-approximation algorithm for this problem. At each point in time during the execution of the algorithm, it chooses those tasks that have the shortest remaining processing time (SRPT).

The parameter P leads to a natural distance function in the context of stability of approximation. We define L as the set of all instances, without restricting the maximum processing time of a job. Then L_I induces the subproblem with unit processing times, *i.e.*, all processing times are exactly 1. For an instance x we define $h(x) = P(x) - 1$, where $P(x)$ is the maximum processing time over all jobs in x. Clearly, h is computable in polynomial time and $h(x) = 0$ for $x \in L_I$.

Since the approximation ratio increases monotonously with the distance from L_I, also the conditions for $B_{r,h}$ are satisfied. We conclude that the SRPT algorithm is stable according to h.

Garg and Kumar [26] considered a generalization of the problem, where each job can only be assigned to a specific subset of machines. Formally, for each job j there is a subset $S(j)$ of machines on which j may be scheduled. They showed that there is an $O(\log P)$-approximation algorithm for this problem. Additionally, they showed a lower bound of $\Omega(\frac{\log P}{\log \log P})$, which means that the ratio is almost tight. From our analysis above, we directly obtain that also this algorithm is stable with respect to h.

5 Degree-bounded Graphs

We now consider degree-bounded graphs. For a graph $G = (V, E)$, we write Δ for the maximum degree over all vertices in V.

5.1 Coloring

The maximum degree plays an important role when we want to color the vertices of G. We would like to find a minimum size set of colors such that each vertex is assigned a color and each edge is incident to two vertices with different colors. The chromatic number of G is the minimum number of colors sufficient to color all vertices of G. It is NP-hard to approximate the chromatic number of an n-vertex graph with an approximation ratio of $n^{1-\varepsilon}$ [44], for an arbitrary $\varepsilon > 0$.

There is a basic and well-known result on graph coloring: one can color each graph with $\Delta+1$ colors. We simply color the graph greedily. We iteratively choose uncolored vertices and color them with the first color c from $\{1, 2, \ldots, \Delta + 1\}$ such that no neighbor is colored c. The color c exists because of the degree bound: there are at most Δ colors with which neighbors can be colored. In particular, the algorithm directly gives a $(\Delta+1)$-approximation algorithm. We can improve the approximation ratio by observing that one can check in polynomial time if a graph is bipartite (cf. [23]), which is equivalent to being 2-colorable. If the graph is bipartite, we can efficiently determine a 2-coloring. We can therefore either compute an optimal solution or we can assume that the optimum is at least 3. The modified algorithm has an approximation ratio of $(\Delta + 1)/3$.

The degree of G determines a useful distance function. We define L as the set of all instances, without restricting the degree. Then L_I induces the subproblem with $\Delta = 3$, *i.e.*, the first number such that graph coloring is NP-hard. For an instance x we define $h(x) = \max\{0, \Delta - 3\}$. The value of h is easy to compute and $h(x) = 0$ for $x \in L_I$.

Since the approximation ratio increases monotonously with Δ, we conclude that the coloring algorithm is stable according to h.

5.2 Independent Set

We continue with the maximum independent set problem in bounded-degree graphs. We would like to find a maximum-size set of vertices S such that they are pairwise non-adjacent.

A well-known result of Halldórsson and Radhakrishnan [29] states that there is a $(\Delta + 2)/3$-approximation algorithm for this problem. The approximation ratio follows from a sophisticated analysis which includes the use of a generalization of Turán's bound.

As for coloring, finding a maximum independent set is easy for degree-2-bounded graphs (we cannot do better than to take every other vertex on each path or cycle). Therefore, the same sets of problem instances and the same distance function as for coloring also apply for maximum independent set. The algorithm is therefore stable according to h.

6 Parameterization

Stability of approximation also appears in the context of parameterized algorithms. As an example, let us again consider graph coloring.

While graph coloring is NP-hard to approximate in general (see Section 5.1), for restricted graph classes the situation is much better. For instance, coloring a perfect graph is polynomially solvable [27].

Here, we focus on a class of graphs determined by their pathwidth. The pathwidth expresses the similarity of a graph to a simple path. In fact, the graphs of pathwidth 1 are exactly $\{K_3, K^*_{1,3}\}$-free graphs (or, equivalently, disjoint unions of caterpillar trees), where $K^*_{1,3}$ is the graph obtained from $K_{1,3}$ by subdividing each edge exactly once [24]. Since here we only use this parameter indirectly, we refer the reader to the book of Downey and Fellows [25] for a formal definition.

A graph has pathwidth zero if and only if it is a single vertex. For pathwidths $k \geq 2$, we have the following result.

Theorem 1. *(Kierstead and Trotter [34]) For graphs of pathwidth k there is a $(3k-2)$-approximation algorithm for graph coloring.*

Therefore, the pathwidth determines a natural distance function. We set L_I to be the set of all graphs of pathwidth zero. Then the distance function h is simply specified by $h(x) = k$ if the given instance x is a graph of pathwidth k.

We claim that the algorithm of Kierstead and Trotter is k-stable for each constant k. Since the dependence of the approximation ratio on k is monotonous, there is a constant α_r for each $r < k$ such that for all instances in $B_{r,h}(L_I)$ the algorithm is an α_r-approximation. It is left to show that h is polynomially computable. This is the point where we have to use that k is a constant: computing the pathwidth is a hard problem. For constant k, however, there is a polynomial time algorithm that computes the pathwidth (in the language of parameterized algorithms, computing the pathwidth is *fixed-parameter tractable*) [15, 17].

For non-constant k, the problem of computing the pathwidth is NP-hard to approximate [16]. We therefore conclude that even though the algorithm of Kierstead and Trotter is k-stable for each constant k, it is neither stable nor quasi-stable.

7 Dimensionality

Graph problems usually have a natural version of the problem where the vertices are points in some space and the distances are measured according to the properties of the space. For instance, we can consider the TSP where the vertices are points in the Euclidean plane. Then a vertex u is a point (u_1, u_2) and the distance between two vertices u and v is the Euclidean distance $((u_1-v_1)^2+(u_2-v_2)^2)^{1/2}$.

One of the important factors influencing the hardness of a problem is the dimension of the considered space. Generally speaking, a higher dimension increases the freedom to choose instances and therefore leads to harder problems.

7.1 Euclidean space

We now consider the Euclidean space more thoroughly. Many of the well-studied problems that are APX-hard in general metric spaces allow for a PTAS in the

Euclidean plane $\mathbb{R}^2$. This is the case for the traveling salesman problem [3, 4], the Steiner tree problem [3, 4], the Steiner forest problem [18], the k-median problem [5, 35], and many related problems.

The hardness of these problems, however, crucially depends on the dimension of the Euclidean space. For $\log n$ dimensions, Trevisan showed that the TSP is APX-hard [43]. This forces us to focus on spaces $\mathbb{R}^d$ for small values of d (either constant, or in some cases in the order of $\log \log n$).

The running time of the $(1+\varepsilon)$-approximation algorithms also depends on ε to be a fixed constant. Since $(1+\varepsilon)$ is a factor, the absolute error increases with the input size n, and it is therefore desirable to have an ε that decreases with increasing n. We are interested in how small ε can be (depending on n) such that the algorithm still runs in polynomial time.

To this end, we again take the TSP as a showcase problem. Arora [3] showed how to obtain a randomized $(1+1/c)$-approximation in time $O(n(\log n)^{(O(\sqrt{d})\cdot c)^{d-1}})$. By choosing $\beta \in O(\sqrt{d})^{d-1}$, we obtain the running time $O(n(\log n)^{\beta \cdot c^{d-1}})$, and we want the time to be polynomial, *i.e.*,

$$
\begin{aligned}
n(\log n)^{\beta \cdot c^{d-1}} &= n^{O(1)} && \text{which implies} \\
\beta \cdot c^{d-1} &= \log_{\log n} n^{O(1)} = \frac{O(\log n)}{\log \log n} && \text{and thus} \\
c &= \Big(\frac{O(\log n)}{O(\sqrt{d})^{d-1} \log \log n} \Big)^{1/(d-1)} .
\end{aligned}
$$

We can formulate this in terms of stability of approximation by choosing L_I to be all 1-dimensional instances, L all d-dimensional instances for $d \geq 1$ and specifying the distance function h as follows. Let $d(x)$ be the number of dimensions used in instance x. Then $h(x) = d(x) - 1$. We conclude that the Euclidean TSP is $f(r)$-quasistable according to h with

$$
f(r) = 1 + 1/\Big(\frac{O(\log n)}{O(\sqrt{\lfloor r+1 \rfloor})^{\lfloor r+1 \rfloor - 1} \log \log n} \Big)^{1/(\lfloor r+1 \rfloor - 1)} .
$$

7.2 Doubling spaces

In the previous section, the only reason to restrict our view to the Euclidean space was that Arora's algorithm uses its properties. The dimensionality behaves orthogonally to the restrictions imposed by the Euclidean space. To this end, we now focus on a more general setting. Given an arbitrary metric space $\mathbb{M}$ with a distance function dist, let B_ℓ denote a ball in this space with radius ℓ according to dist.[4] Then the doubling dimension of $\mathbb{M}$ is the minimal number d such that for every ℓ, every ball B_ℓ can be covered by 2^d balls $B_{\ell/2}$ of radius $\ell/2$. Such a metric space is called a *doubling space* of dimension d. As an example, the doubling dimension of a 2-dimensional space with ℓ_1-norm is 2, since each

[4] Note that here we do not talk about the relaxation of a class of instances, but about the standard notation of balls in a metric space.

square can be covered by $2^2 = 4$ squares of half the radius. Doubling spaces of dimension $O(d)$ generalize the Euclidean space with d dimensions, and are significantly more general (cf. [6, 22, 28, 42] and references therein).

As an example of a stability result according to the dimension of a doubling space we use the Maximum Scatter TSP (MSTSP). In the MSTSP, we are given a TSP instance. Now, the goal is to find a tour such that the *smallest* cost over all edges in the tour is *maximized*. In other words, the objective is to only use long jumps. The idea is that the order of points visited is scattered as much as possible within the considered graph. A solution to the MSTSP is required, for instance, when drilling holes and the work piece is heated up in the process. Then one should first process parts of the work piece far away from the last hole in order to prevent overheating.

Kozma and Mömke [36] showed that there is a PTAS for the MSTSP in doubling spaces of dimension d at most $O(\log \log n)$, where n is the number of vertices. More precisely, the result is a $(1+\varepsilon)$-approximation algorithm running in time $\tilde{O}\big(n^3 + 2^{O(K \log K)}\big)$, where $K \leq \left(\frac{13}{\varepsilon}\right)^d$. We want to determine ε such that

$$\tilde{O}\big(n^3 + 2^{O(K \log K)}\big) = \log^c n \cdot \big(n^3 + 2^{O(K \log K)}\big) = n^{O(1)}\,. \tag{1}$$

The factor $\log^c n$ and the term n^3 can be hidden in the exponent $O(1)$ of the right-hand side of (1). We obtain $2^{O(K \log K)} = n^{O(1)}$. We further simplify the equation and obtain

$$K \log K = (13/\varepsilon)^d \log(13/\varepsilon)^d = O(\log n)\,.$$

For sufficiently small ε then $1/\varepsilon^{d-1/2} < \log n$ and therefore

$$\varepsilon > (\log n)^{-1/(d-1/2)}\,.$$

The remaining steps are analogous to our analysis of the Euclidean space. It is sensible to define L_I to be the one-dimensional instances and to use the distance function $h(x) = d(x) - 1$. Then the algorithm is $f(r)$-quasistable according to h with

$$f(r) = 1 + (\log n)^{-1/(\lfloor r+1 \rfloor - 1/2)}\,.$$

8 Online Algorithms

The concept of stability of approximation is not restricted to approximation algorithms. It naturally translates to concepts such as online algorithms.

The source of hardness in online algorithms comes from incomplete knowledge rather than from time restrictions: we (*i.e.*, the algorithm) have to take decisions before knowing the entire instance.

Formally, an online problem U is defined as follows. An instance of U is given as a sequence of requests $r_1, r_2, \ldots$. An online algorithm A for U has to provide a sequence of answers $a_1, a_2, \ldots$. The answer a_i depends on the requests $r_1, r_2, \ldots, r_i$ and the answers $a_1, a_2, \ldots, a_{i-1}$.

In order to focus on the essence of the problem, the algorithm does not have a time restriction. It is sufficient to provide a computable function that determines the answers.

There are several models of online computation. One of the most used and most reasonable models is to assume an oblivious adversary. This model provides a measure of worst-case complexity because we assume an adversary that knows our algorithm and tries to provide a worst possible request sequence for it. The adversary, however, is oblivious which means that he has to fix the request upfront, without knowing the algorithm's answers. (This does not change anything for deterministic online algorithms, but it provides a huge advantage for randomized online algorithms.)

Online problems are typically optimization problems and as in approximation algorithms, we would like to have a guaranteed quality of the computed solution. Unlike in approximation algorithms, however, the need to compromise quality in online algorithms is unconditional: we do not depend on conjectures such as $\mathsf{P} \neq \mathsf{NP}$. The analog to the approximation ratio in online algorithms is called *competitive ratio*.

An algorithm is *c-competitive*, if there is a constant $\alpha > 0$ such that for each sequence of requests with cost opt of an optimal offline solution, the cost alg of the solution computed by the algorithm is at most $\alpha + c \cdot \mathsf{opt}$.

We define stability for online algorithms analogous to Section 2, but we replace the approximation ratio by the competitive ratio. We note that unless we require the online algorithm to run in polynomial time, the requirement of the distance function h to be polynomial time computable is not essential for online algorithms. We therefore only require h to be a computable function.

8.1 Online Scheduling

We continue with the online view on the problems that we analyzed in Section 4. Again, we want to minimize the flow-time on multiple machines. We have formally defined the offline version of this problem in Section 4. The difference to the offline version is that now, the jobs arrive in an online fashion. Let $j_1, j_2, \ldots, j_n$ be the jobs of a problem instance. We then order the jobs by increasing release time. The i-th request reveals the i-th job j_i. Before a new job j_{i+1} (or the end of the input) is revealed, the algorithm has to fix the machine on which j_i will be processed.

Leonardi and Raz [37, 38] showed a tight bound $\Theta(\log P)$ on the competitive ratio of this problem. Recall that P is the maximal processing time of a single job. Note that the upper bound matches the offline upper bound. For online algorithms, however, the lower bound has changed.

Again, P leads to a natural distance function in the context of stability of approximation. Recall that the set L contains all instances, without restricting the maximum processing time of a job and that L_I induces the subproblem with unit processing times. For an instance x we define $h(x) = P(x) - 1$, where $P(x)$ is the maximum processing time over all jobs in x.

Since the competitive ratio increases monotonously with the distance from L_I, also the conditions for $B_{r,h}$ are satisfied. We conclude that the SRPT algorithm is stable according to h.

Unlike in the case of offline algorithms, however, this result does not translate to the version of the problem where each job can only be assigned to a specific subset of machines. Garg and Kumar [26] showed that in this case, the competitive ratio is unbounded, independent of P. Therefore, for this version of the problem there is no algorithm that is stable according to h.

8.2 Online Matching

The k-server problem is one of the central problems studied in online computation. We are given a metric space $\mathbb{M}$ and a set S of k servers placed on various locations in $\mathbb{M}$. The initial positions of the servers are known. The online requests are points in $\mathbb{M}$ and have to be answered with a server from S, which then has to move to the requested point. The cost of moving a server equals the distance in $\mathbb{M}$ and the goal is to minimize the overall distance traveled by servers.

The k-server problem formalizes natural problems such as for instance real time delivery: the servers are delivery cars and the metric space is determined by clients within a street network. In such a network, it is natural to consider a capacitated version of the k-server problem. We assign a capacity to the servers: each delivery car can serve at most ℓ different clients.

If $\ell = 1$, the capacitated k-server problem boils down to a bipartite matching between the servers and the clients: each server is matched to at most one client and each client is served by exactly one server. The resulting problem is known as online bipartite matching.

Raghvendra [41] presented an algorithm for online bipartite matching which in the following we call RM-algorithm (for "robust matching," following the notation of the authors). Nayyar and Raghvendra [40] afterwards obtained a better analysis based on the parameter $\mu_{\mathbb{M}}(S)$ which is defined as the maximum ratio of the traveling salesman tour and the diameter over all subsets of S (or, more precisely, the graphs induced by S). To use the parameter, we rely on our relaxation of stability for online algorithms in which we defined the distance function to be only computable and did not require polynomial time. The meaning of the parameter $\mu_{\mathbb{M}}(S)$ becomes clear when we consider concrete classes of metric spaces $\mathbb{M}$. If $\mathbb{M}$ is the space determined by the shortest path metric of a line, $\mu_{\mathbb{M}}(S) = 2$. If $\mathbb{M}$ is a doubling space of dimension d, $\mu_{\mathbb{M}}(S) = O(n^{1-1/d})$.

The result of Nayyar and Raghvendra is that the RM-algorithm has a competitive ratio of $O(\mu_{\mathbb{M}}(S) \log^2 n)$, while every algorithm has a competitive ratio of $\Omega(\mu_{\mathbb{M}}(S))$. To analyze the stability of the RM-algorithm, we now fix the needed parameters.

We set L to the set of all feasible instances independent of the value of $\mu_{\mathbb{M}}(S)$. One can check that for an arbitrary metric space on three points the value of $\mu_{\mathbb{M}}(S)$ is at least 2, matching the value for a shortest path metric of a line. Since the latter class of instances is non-trivial, it is natural to set L_I to be all instances with $\mu_{\mathbb{M}}(S) \leq 2$. Let S_x be the set of servers located in the metric

space $\mathbb{M}_x$ for an instance $x \in L$. We then define $h(x) = \max\{0, \mu_{\mathbb{M}_x}(S_x) - 2\}$, which satisfies all required conditions. Since the upper bound on the competitive ratio is non-constant, we do not have an analysis that states that the algorithm is stable according to h. However, we can claim that it is f-quasistable according to h for the function f with $f(n) = O(\mu_{\mathbb{M}}(S) \log^2 n)$.

9 Conclusion

We have seen that stability of approximation is a natural concept that appears in many circumstances. We can draw several conclusions from our investigation.

(i) Stability of approximation can be seen as a parameterized version of approximation algorithms. If the approximation ratio monotonously increases with a given parameter, this parameter is a natural base for a stability distance function. Conversely, for stability of approximation results, we could also state the algorithms as algorithms for general problem instances parameterized on the distance function.

(ii) Stability of approximation has similarities with the study of time complexity. The relation becomes clear when considering polynomial time approximation schemes. In this survey, we determined the impact of stability on the error ε. An alternative way to handle a PTAS would be to fix a specific running time and to determine the error ε depending on the stability distance function. We can see the situation as a pareto curve with the stability distance on one axis and the running time on another axis.

(iii) The concept of stability is not restricted to approximation algorithms. For example, it directly translates to stability of competitiveness in online algorithms. This insight motivates to further investigate the scope of stability in further contexts.

To summarize, we observe that classifying the hardness of the instances of the investigated algorithmic problems is a successful approach that is important in modern algorithmic research.

Acknowledgment. We thank the anonymous referees for their careful reading and valuable comments, which helped to improve the presentation of the paper.

References

1. Thomas Andreae. On the traveling salesman problem restricted to inputs satisfying a relaxed triangle inequality. *Networks*, 38(2):59–67, 2001.
2. Thomas Andreae and Hans-Jürgen Bandelt. Performance guarantees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics*, 8(1):1–16, 1995.
3. Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

4. Sanjeev Arora. Approximation schemes for NP-hard geometric optimization problems: a survey. *Mathematical Programming*, 97(1-2):43–69, 2003.
5. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proc. 30th Annual ACM Symposium on the Theory of Computing (STOC 1998)*, pages 106–113, 1998.
6. Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme. *SIAM Journal on Computing*, 45(4):1563–1581, 2016.
7. Michael A. Bender and Chandra Chekuri. Performance guarantees for the TSP with a parameterized triangle inequality. *Information Processing Letters*, 73(1-2):17–21, 2000.
8. Hans-Joachim Böckenhauer, Dirk Bongartz, Juraj Hromkovič, Ralf Klasing, Guido Proietti, Sebastian Seibert, and Walter Unger. On the hardness of constructing minimal 2-connected spanning subgraphs in complete graphs with sharpened triangle inequality. *Theoretical Computer Science*, 326(1-3):137–153, 2004.
9. Hans-Joachim Böckenhauer, Dirk Bongartz, Juraj Hromkovič, Ralf Klasing, Guido Proietti, Sebastian Seibert, and Walter Unger. On k-connectivity problems with sharpened triangle inequality. *Journal of Discrete Algorithms*, 6(4):605–617, 2008.
10. Hans-Joachim Böckenhauer, Juraj Hromkovič, Ralf Klasing, Sebastian Seibert, and Walter Unger. Approximation algorithms for the TSP with sharpened triangle inequality. *Information Processing Letters*, 75(3):133–138, 2000.
11. Hans-Joachim Böckenhauer, Juraj Hromkovič, Ralf Klasing, Sebastian Seibert, and Walter Unger. An improved lower bound on the approximability of metric TSP and approximation algorithms for the TSP with sharpened triangle inequality. In *Proc. 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2000)*, volume 1770 of *Lecture Notes in Computer Science*, pages 382–394. Springer-Verlag, 2000.
12. Hans-Joachim Böckenhauer, Juraj Hromkovič, Ralf Klasing, Sebastian Seibert, and Walter Unger. Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. *Theoretical Computer Science*, 285(1):3–24, 2002.
13. Hans-Joachim Böckenhauer and Sebastian Seibert. Improved lower bounds on the approximability of the traveling salesman problem. *RAIRO - Theoretical Informatics and Applications*, 34(3):213–255, 2000.
14. Hans-Joachim Böckenhauer, Sebastian Seibert, and Juraj Hromkovič. Stability of approximation. In *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007.
15. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.
16. Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.
17. Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.
18. Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for Euclidean Steiner forest. *ACM Transactions on Algorithms*, 11(3):19:1–19:20, 2015.
19. Li-Hsuan Chen, Dun-Wei Cheng, Sun-Yuan Hsieh, Ling-Ju Hung, Ralf Klasing, Chia-Wei Lee, and Bang Ye Wu. Approximability and inapproximability of the star p-hub center problem with parameterized triangle inequality. *Journal of Computer and System Sciences*, 92:92–112, 2018.

20. Li-Hsuan Chen, Sun-Yuan Hsieh, Ling-Ju Hung, and Ralf Klasing. The approximability of the p-hub center problem with parameterized triangle inequality. In *Proc. 23rd International Computing and Combinatorics Conference (COCOON 2017)*, volume 10392 of *Lecture Notes in Computer Science*, pages 112–123. Springer Verlag, 2017.
21. Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.
22. Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete & Computational Geometry*, 22(1):63–93, 1999.
23. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
24. Guoli Ding and Stan Dziobiak. On 3-connected graphs of path-width at most three. *SIAM Journal on Discrete Mathematics*, 27(3):1514–1526, 2013.
25. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer-Verlag London, 2013.
26. Naveen Garg and Amit Kumar. Minimizing average flow-time : Upper and lower bounds. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pages 603–613. IEEE Computer Society, 2007.
27. Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.
28. Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 534–543. IEEE Computer Society, 2003.
29. Magnús M. Halldórsson and Jaikumar Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, 1997.
30. Juraj Hromkovič. Stability of approximation algorithms and the knapsack problem. In *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 238–249. Springer, 1999.
31. Juraj Hromkovič. *Algorithmics for Hard Problems - Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics, Second Edition*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.
32. Juraj Hromkovič. Stability of approximation in discrete optimization. In *IFIP TCS*, volume 155 of *IFIP*, pages 3–18. Kluwer/Springer, 2004.
33. Juraj Hromkovič. Algorithmics - Is there hope for a unified theory? In *CSR*, volume 6072 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2010.
34. Henry A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.
35. Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem. *SIAM Journal on Computing*, 37(3):757–782, 2007.
36. László Kozma and Tobias Mömke. Maximum scatter TSP in doubling metrics. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 143–153, 2017.
37. Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. In *Proc. 29th Annual ACM Symposium on the Theory of Computing (STOC 1997)*, pages 110–119. ACM, 1997.

38. Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. *Journal of Computer and System Sciences*, 73(6):875–891, 2007.
39. Tobias Mömke. An improved approximation algorithm for the traveling salesman problem with relaxed triangle inequality. *Information Processing Letters*, 115(11):866–871, 2015.
40. Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 505–515. IEEE Computer Society, 2017.
41. Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *APPROX-RANDOM*, volume 60 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
42. Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC 2004)*, pages 281–290, 2004.
43. Luca Trevisan. When Hamming meets Euclid: The approximability of geometric TSP and steiner tree. *SIAM Journal on Computing*, 30(2):475–485, 2000.
44. David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.