3 Scientific Information

3.1 Summary

The project comprises basic research in the theoretical analysis of online problems. We shall consider the standard model of online computation, in which the algorithm is processing a sequence of inputs, and has to deliver a piece of output to each input (independently from the future inputs). The lack of information about the future input poses a new level of hardness, which is usually characterized by means of the competitive ratio.
In the proposed project, we plan to characterize the hardness of online problems using a generalization of this concept, taking into account the amount of the missing problem-specific information. The project is divided into two phases. First, we define a complexity measure that quantifies the amount of problem-specific information contained in the input. This measure is expressed as the number of bits that must be communicated between the algorithm and an oracle that sees the whole input, in order to solve the problem optimally. In the first phase, we shall investigate this measure and its various variants, and deliver lower and upper bounds on the amount of problem-specific information for a number of online problems. In the second phase, we shall use this measure to investigate the tradeoff relation between the amount of information about the future and the best achievable solution quality.
The amount of problem-specific information has not been considered before, and we believe that it may give better insight into fundamental properties of online computation. Moreover, this approach can lead to identifying some structural properties of inputs that may in turn lead to the design of more efficient algorithms. Also, there may be semi-online scenarios where the input is available, but has to be accessed sequentially by the algorithm.
The project fits into the scope of existing expertise in the research area of hardness of computational tasks.

3.2 Rationale for the proposed project and state of research

3.2.1 Rationale

The understanding of the hardness of online problems is both challenging from the point of view of theory, and important for practical applications. The amount of problem-specific information has not been considered before, and we believe that it may give better insight into fundamental properties of online computation. Moreover, this approach can lead to identifying some structural properties of inputs that may in turn lead to the design of more efficient algorithms. Also, there may be semi-online scenarios where the input is available, but has to be accessed sequentially by the algorithm (e.g. a series of orders sent to a remote robot). In this case, it may be worthwhile to equip each data item with some additional information to help guide the processing.

3.2.2 Introduction

One of the principal areas of interest in theoretical computer science is the understanding of various aspects of hardness of computational tasks. The most standard approach to representing a computational task is by means of a mapping from a set of input values to a set

of output values. Having this representation, it is natural to ask what formal models are appropriate to describe a given mapping, how complex the description is in a particular model, what resources are needed to implement it, etc. There are, however, many computational tasks in the real world that do not comply with this representation for a variety of reasons. For instance, numerous processes require that parts of output are delivered before the whole input is revealed, as is typical for processing of large (potentially even infinite) sequences of requests. This lack of information about future input poses a new level of hardness, and tasks operating in this fashion are usually termed "*online*". Many prominent examples of online problems are found in operating systems and client-server applications. As an illustration, one may consider a web server with caching: requests for specific web pages arrive to the server, and to each request the corresponding page must be offered before processing the next request. The fact that the future requests are not known makes it difficult to decide which pages should be stored in a fast internal cache of the server for future reference.

On line problems have been subject to extensive study since the sixties, with the main effort being put to the design and analysis of online algorithms that cope with the ignorance of future in the best possible way. This objective of overcoming the lack of information is usually measured by the degradation of the output quality incurred by the online setting, and the most standard way of expressing this degradation is the *competitive ratio*: the ratio of the output quality of a given online algorithm towards the optimal solution.

In the proposed project, we plan to characterize the hardness of online problems using a generalization of this concept which is based on another central notion of theoretical computer science – the notion of the complexity of an object. The central idea is that having a set of finite objects, e.g. boolean functions or strings of letters, some of them are more complex than others; the complex objects are more difficult to describe and hence they carry more information. Many ways of measuring this information carried by an object have been proposed, e.g. Kolmogorov complexity, communication complexity, Shannon's entropy, etc.

Our aim is to study the relationship between the amount of information in the unknown (future) part of the input and the best achievable solution quality for a given online problem. However, we need to restrict ourselves only to the information that is relevant to the problem (e.g. in the web server example the input contains large amounts of information concerning the content of the pages, users that requested them etc., which is irrelevant to the problem).

We propose to measure the amount of problem-specific information by the number of bits that must be given to the algorithm to perform optimally. The goal of the project is to find out how much information is needed to achieve a required solution quality in various online scenarios.

3.2.3 Scope of the project

The project comprises basic research in theoretical computer science. We shall consider the standard model of online computation, in which the algorithm is processing a sequence of inputs, and has to deliver a piece of output to each input (independently from the future inputs).

First, we modify the model by allowing the algorithm to communicate with an external entity that has all information about the input. We measure the amount of problem-relevant information by the number of bits the algorithm needs to communicate in order to achieve optimal performance. The first stage of the project will be devoted to tuning the formal model of this concept in order to mimic the intuition as much as possible, and to deliver tight

upper and lower bounds on the problem-specific information of a number of well understood online problems.

In the second stage of the project, we shall study the amount of information that needs to be communicated in order to achieve performance with a given worst case approximation guarantee. We may consider also modifications of the scenario (e.g. by introducing random sources) to see how these affect the solutions.

3.2.4 State of research

Competitive analysis

The first results dealing with online problems were published in the sixties in the context of searching algorithms [32]. A systematic study of online algorithms began in the late sixties [50], and it has received much attention over the years (see e.g. [24, 39, 74, 53]). Formally, an online algorithm processing an input sequence of requests $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$ produces an output sequence $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$ in such a way that each y_i is computed as a function of the prefix $\langle x_1, x_2, \ldots, x_i \rangle$. On the other hand, an algorithm computing the whole output sequence $\boldsymbol{y}$ from the entire input sequence $\boldsymbol{x}$ is termed "offline". The standard measure used for evaluating online algorithms is the competitive ratio [57, 69], i.e. the worst-case ratio of the solution quality of the given online algorithm to the optimal solution. An online algorithm is c-competitive, if there is a constant $c > 0$ such that, for each input sequence $\boldsymbol{x}$, $Q_{\mathcal{A}}(\boldsymbol{x}) \geq c \cdot Q_{OPT}(\boldsymbol{x})$ holds, where $Q_{\mathcal{A}}$ and Q_{OPT} denote the solution quality of the online algorithm and the best possible solution quality, respectively.

One of the most extensively studied and best understood online problem is the PAGING problem, originated in the context of memory management in operating systems. The virtual memory of a computer is divided into logical pages. At any time, K logical pages can reside in the physical memory. A paging algorithm is the part of the operating system responsible for maintaining the physical memory. If a program requests access to a logical page that is not currently in the physical memory, a *page fault* interrupt occurs and the paging algorithm has to transfer the requested page into physical memory, possibly replacing another one. Formally, the input for a paging algorithm is a sequence of integers (logical pages) $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$. The algorithm maintains a buffer (physical memory) $B = \{b_1, \ldots, b_K\}$ of K integers. Upon receiving an input x_i for which $x_i \notin B$, a page fault is generated, and the algorithm has to find some *victim* b_j that shall be replaced in the physical memory, i.e. $B := B \setminus \{b_j\} \cup \{x_i\}$. The quality is measured by the number of generated page faults.

A number of online algorithms for the paging problem has been proposed, two of the most notoriously known are the LRU (least recently used), where the victim is always the page that has been requested least recently, and FIFO (first in – first out), where the victim is the page that has been in the physical memory longest. Sleator and Tarjan have proven [77] that both LRU and FIFO are K-competitive. Moreover, it the same work it is proven that no deterministic online paging algorithm can have competitive ratio better than K. We just note that the optimal offline algorithm is due to Belady [33].

The previous example illustrates the flavor of typical results in the theory of online algorithms. On one hand, there are algorithms with guaranteed competitive ratio, on the other hand there are lower bounds stating that a better competitive ratio cannot be achieved.

There have been many other problems originating from different areas of computer science studied within this framework, and it is beyond the scope of this survey to give an exhaustive

overview. We only briefly mention some representative problems from selected application areas.

In the *data structures* setting, one of most prominent problems is the ListUpdate problem, where the task is to maintain a dictionary as an unsorted linear list. The requests may be of three kinds: access, insertion, and deletion of an item. All three requests are processed by scanning the list sequentially until either the requested item is found, or the end of the list is reached. The cost of this scan is proportional to the number of elements scanned. While processing a request sequence, a list update algorithm may rearrange the list. Immediately after an access or insertion, the requested item may be moved at no extra cost to any position closer to the front of the list. These exchanges are called free exchanges. Using free exchanges, the algorithm can lower the cost on subsequent requests. At any time, two adjacent items in the list may be exchanged at a cost of 1. These exchanges are called paid exchanges. For the list update problem, algorithms with competitive ratio 2 have been proposed in [23], and a matching lower bound was proven in an unpublished result of Karp and Raghavan.

As a typical problem for the *traffic management in communications*, we may consider the DiffServ problem, widely studied using competitive analysis (see [45, 66], and references therein). The setting involves a server processing an incoming stream of packets of various values. If the processing speed of the server is slower than the arrival rate, some packets must be dropped, ideally those least valuable. Usually it is assumed that the packets arrive in discrete time steps. In each step, a number of packets can arrive, one packet can be processed, and at most K packets can be stored in a buffer. Moreover, it is required that the packets are processed in FIFO manner. More formally, the input is formed by a sequence of items of various values that is partitioned into (possibly empty) requests. In each step the algorithm maintains an ordered buffer B of K items. Upon receiving a request x, the algorithm discards some elements from the sequence $\langle B, x\rangle$, keeping some subsequence of length at most $K + 1$. The first item of this sequence (if it is non-empty) is submitted, and the remainder of the sequence forms the new buffer. The process ends if there are no more requests and the buffer is empty. The quality of the solution is the sum of the values of all submitted elements. Lotker and Patt-Shamir [66] presented an optimal greedy offline algorithm. In [45], tight lower and upper bounds of approximately 1.281 on the competitive ratio were proven.

In the *distributed data management* area, many variants of migration and replication problems have been studied. In the FileAllocation problem, one considers a network of computers that want to access (i.e., read from or write to) a single file. There may be many copies of this file in the network, and for a read request any of them can be accessed. Data may be transmitted along the communication links, and each transmission incurs a unit cost. Copies of the file can be created or destroyed, however, they must be kept consistent, i.e., every write access must be followed by a number of data transfers that update (or destroy) each copy. In [31], a lower bound $\Omega(\log n)$ has been shown on the competitive ratio of any file allocation algorithm, and in [30] a matching upper bound has been established.

Other widely-studied areas include robot searching and navigation (e.g. [26, 38, 81, 37, 44, 63]), routing (e.g. [65, 27, 29]), graph problems like coloring (e.g. [52, 62, 67, 80]), matching (e.g. [61, 55, 59]), Steiner tree (e.g. [28, 36, 51, 82]) etc.

Randomization

All previously mentioned examples were of a deterministic setting, with the analysis being performed against the worst case. The advantage of this approach is that the obtained results hold in general without any a priori assumptions about the environment. On the other hand, there is also a number of drawbacks. The worst case scenario is often a very special situation that is quite unlikely to happen in reality. Moreover, often a single very unlikely case can degrade the performance of any algorithm to such extent that it is not possible to distinguish between algorithms that behave quite differently in real-life situations. As an example, both LRU and FIFO algorithms for the paging problem have optimal competitive ratio of K. Another example is the list update problem, where the optimal competitive ratio of 2 is achieved by a simple move-to-the-front algorithm, although it is by far outperformed in real life situations by more advanced algorithms.

The worst-case analysis of online problems can be viewed as a game between the algorithm and an adversary which selects the worst possible input. Many problems of the competitive analysis rise from the fact that this adversary is given too much power to choose the input without any restrictions. Hence, a number of ways to restrict the power of the adversary have been investigated, perhaps the most prominent one being randomization.

There are two ways of introducing randomness into the computation. In the *probabilistic analysis*, one would not consider the worst-case input, but a random input chosen from a given distribution. The advantage of this approach is that it limits the power of adversary, thus giving less pessimistic (and, hopefully, more realistic) results. The input distribution can be viewed as a form of information about the future input. In this setting, the paging problem (formulated in the context of designing interfaces between IP networks and connection-oriented networks) has been analyzed in [68]. In [58], the input distribution is considered to be generated by a Markov chain, and an optimal online algorithm in this setting is designed.

In *randomized online algorithms*, the worst-case input is considered, but the power of the adversary is counterbalanced by allowing the algorithm to access a source of random bits, and analyze the expected behavior of the algorithm on the input (over the choice of random bits). The rationale is that, since the algorithm is not deterministic, it is more difficult for the adversary to induce poor behavior by selecting a particular input. In this setting, Ben-David et al. [35] introduced three kinds of adversaries; the basic difference is that an adaptive adversary can react to the behavior of the algorithm, whereas an oblivious adversary has to generate the whole input in advance without knowing the random bits.

1. *Oblivious Adversary* The oblivious adversary has to generate a complete request sequence in advance, before any requests are served by the online algorithm. The adversary is charged the cost of the optimum offline algorithm for that sequence.

2. *Adaptive Online Adversary* This adversary may observe the online algorithm and generate the next request based on the algorithm's (randomized) answers to all previous requests. The adversary must serve each request online, i.e., without knowing the random choices made by the online algorithm on the present or any future request.

3. *Adaptive Offline Adversary* This adversary also generates a request sequence adaptively. However, it is charged the optimum offline cost for that sequence.

A randomized online algorithm $\mathcal{A}$ is called c-competitive against any oblivious adversary if there is a constant a such that for all request sequences $\boldsymbol{x}$ generated by an oblivious adversary, $E[Q_{\mathcal{A}}(\boldsymbol{x})] \geq c \cdot Q_{\mathcal{OPT}}(\boldsymbol{x}) + a$ The expectation is taken over the random choices made by $\mathcal{A}$.

In [35], the relative strengths of the adversaries have been investigated, with the conclusion that randomization does not help against the adaptive offline adversary, and at most quadratic improvement in the competitive ratio can be achieved against adaptive online adversary. Hence, most of the results deal with the oblivious adversary.

There is a fundamental relationship between the randomized algorithms against the oblivious adversary, and the probabilistic analysis of deterministic algorithms, which is an application of the Yao's minimax theorem [83]. In the context of online computation, it can be formulated as follows: Given an online problem, the competitive ratio of the best randomized online algorithm against any oblivious adversary is equal to the competitive ratio of the best deterministic online algorithm under a worst-case distribution. Hence, for proving lower bounds on the competitiveness of a randomized algorithm against the oblivious adversary, it is sufficient to consider the optimal deterministic algorithm against all possible input distributions.

As in the deterministic case, there are many results concerning many different problems. Here, we present just some examples that complement the results mentioned in the previous section. For the PAGING problem, Fiat et al. [46] proved a lower bound $\Omega(H_K)$ on the competitive ratio of the randomized online algorithms against oblivious adversaries, with tight upper bounds following in [21, 70]. For the LISTUPDATE problem, a 1.6-competitive algorithm against the oblivious adversary is presented in [25], and a lower bound of 1.5 is established in [78]. For the FILEALLOCATION problem, tight bounds are established in [31].

The following table summarizes the deterministic and randomized competitive ratios of the presented examples. It demonstrates that in some cases (e.g.PAGING) randomization

problem	competitive ratio	
	deterministic	randomized (oblivious adversary)
PAGING	K	$H_K \approx \log K$
LISTUPDATE	2	between 1.5 and 1.6
DIFFSERV	≈ 1.281	
FILEALLOCATION	$\Theta(\log n)$	$\Theta(\log n)$

brings an exponential improvement, whereas in other cases it does not help much.

Other approaches

The main disadvantage of the randomized approach is that it cannot give a-priori bounds, only bounds on the expected behavior. To overcome this, a number of alternative measures of hardness of online algorithms have been proposed, either tailored to some particular problems (e.g. loose competitiveness [84]), or usable in a more general setting. Among the more general models, many forms of *resource augmentation* have been studied (e.g. [43, 56, 75]). The common idea of these approaches is to counterbalance the lack of information about the input by granting more resources to the online algorithm (e.g. by comparing the optimal offline algorithm to an online algorithm that works k times faster). Another approach was to use a *look-ahead* where the online algorithm is allowed to see some limited number of future requests [22, 34, 56, 79]. The main problem with the look-ahead approach is that a look-ahead of constant size generally does not improve the worst case performance measured by the competitive ratio. Yet another approach is based on not comparing the online algorithms to offline ones, but to other online algorithms instead (e.g. Max/Max ratio [34], relative worst-order ratio [42]; see also [41]). Still another approach is to limit the power of the adversary as e.g. in the access graph model [40, 54], statistical adversary model [76], diffuse adversary

model [64], etc. Still another approach that shares some similarities with the method proposed in this project is based on entropy [72, 73]. Finally, a somewhat similar approach of measuring the complexity of a problem by the amount of additional information that has to be added to the algorithm to solve it has been recently pursued in the context of distributed computing by Fraigniaud, Gavoille, Ilcinkas, and Pelc [47, 49, 48].

3.2.5 Proposed model

Our approach is a generalization of the ideas that have been implicitly present in previous work: e.g. the assumption about known distribution of inputs is a form of information that is additionally given to the algorithm. Also, adding the relevant information can be seen as a form of resource augmentation.

The idea how to measure the relevant information about the input is inspired by the communication complexity research. We consider, in addition to the algorithm itself, an oracle that sees the whole input and knows the algorithm. When computing the i-th output y_i, the algorithm not only sees the sequence $\langle x_1, x_2, \ldots, x_i \rangle$, but can also communicate with the oracle. We require that the algorithm always computes an optimal solution, and define the advice complexity of the algorithm-oracle pair ($B(\mathcal{A}, \mathcal{O})$) as the number of bits communicated between the algorithm and the oracle, normalized per request, i.e.,

$$B(\mathcal{A}, \mathcal{O}) = \limsup_{n \mapsto \infty} \max_{|\boldsymbol{x}|=n} \frac{B(\mathcal{A}, \mathcal{O})(\boldsymbol{x})}{n}$$

where $B(\mathcal{A}, \mathcal{O})(\boldsymbol{x})$ is the number of bits communicated on the input sequence $\boldsymbol{x}$, and n is the length of the input sequence.

The amount of problem-specific information contained in the input is the advice complexity of an online problem, and it is the minimum advice complexity over all oracle–algorithm pairs that together solve the problem, i.e.

$$B(\mathcal{P}) = \min_{(\mathcal{A}, \mathcal{O})} \mathcal{B}(\mathcal{A}, \mathcal{O})$$

where the minimum is taken over all $(\mathcal{A}, \mathcal{O})$ such that $\forall \boldsymbol{x} : Q_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x}) = Q_{OPT}(\boldsymbol{x})$

Note that there are two ways to achieve trivial upper bounds on advice complexity: (1) the oracle can send, in some compressed way, the whole input to the algorithm, which then can proceed as an optimal offline algorithm, and (2) the oracle can tell the algorithm exactly what to do in each step. However, both these approaches can be far from optimum. In the first case all information about the future input is communicated, although it may not be relevant. In the second case, the power of the online algorithm is completely ignored. Indeed, an online algorithm may be able to process large parts of the input optimally without any advice, requiring only occasional help from the oracle.

As another motivating example, consider the well known SkiRental problem [60] that has been used to model various scenarios e.g. in power management of notebooks: A skier can rent a set of skis for one day for a unit price, or buy them for a fixed price c. However, it is only the morning of each day when it becomes clear if the skier wants to continue skiing or not. A classical result of the competitive analysis [60] shows that the optimal worst case performance is achieved by first renting the skis for $c - 1$ days, and then buying them, which gives a competitive ratio of $2 - 1/c$. Hence, the unknown future part of the input carries information that prevents any algorithm from behaving better than twice the

optimum. However, the structure of the problem is very simple, as a single bit (indicating whether to buy the skis or not) of information is sufficient to achieve optimal performance. For the PAGING problem, this amount of hidden information is much higher.

3.2.6 State of our research

The project fits into the scope of existing expertise in the research area of hardness of computational tasks. In the last five years we published several papers devoted to finding high quality solutions to different optimization problems [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 17, 18, 19, 20], and wrote a textbook [15] on this topic. We focused on online problems especially in [17, 18] where we consider the job shop scheduling problem *unit–*J_m. The input of the problem is a set of d jobs that have to be executed on a collection of m machines in such a way that each job has to be processed once on each of the machines. Every job consists of a permutation of tasks for all machines. The execution of any task on its corresponding machine takes exactly one time unit. The objective is to minimize the overall completion time, called makespan. We deliver lower and upper bounds of $m + o(m)$ on the makespan for $d = o(m^{1/2})$, and we present a randomized on-line approximation algorithm for *unit–*$J_m$with the best known approximation ratio for this case. Moreover, we present a deterministic approximation algorithm for *unit–*$J_m$that works in quadratic time for constant d and has an approximation ratio of $1 + 2^d/\lfloor\sqrt{m}\,\rfloor$ for $d \leq 2\log_2 m$.

The project shall continue in cooperation with the ongoing research in [12], where first ideas about measuring the amount of knowledge of future requests were introduced, and two modes of interaction with the oracle have been proposed. In the *helper* mode, the algorithm itself cannot activate the oracle; instead, the oracle oversees the progress of the algorithm, and occasionally sends some pieces of advice. In the *answerer* mode the oracle remains passive, and the algorithm may, in any particular step, ask for advice.

We have analyzed the PAGING and DIFFSERV problems, and delivered upper and lower bounds on their advice complexity as seen in the following table:

	competitive ratio	helper	answerer
PAGING	K [77]	$(0.1775, 0.2056)$	$(0.4591, 0.5 + \varepsilon)$
DIFFSERV	≈ 1.281[45]	$\frac{1}{K}$	$\left(\frac{\log K}{2K}, \frac{\log K}{K}\right)$

3.3 Detailed research plan

3.3.1 First phase

In the first phase of the project, we shall consider variants of the proposed model, especially in two aspects: delimiting and bounding the communication. In the present model, the bit strings sent by the oracle are in a non-delimited form. The algorithm works in a synchronous setting where, in the i-th step, it receives the i-th input request x_i, and possibly some advice a_i, based on which it produces the output y_i. In a manner usual in synchronous distributed algorithms (see e.g. [71] and references therein) we count the number of bits communicated between the oracle and the algorithm, relying upon the timing mechanism for delimiting both input and advice sequences. Requiring that both the input requests and the oracle advises come in a self-delimited form, would alter our upper bounds by a factor of at most 4; the exact values, however, are unknown.

Another feature of the model is that the size of a single advice is unbounded in the sense that it may depend on the size of input, a feature that is actually used in our proofs. If we limit the size of the advice to be constant we would obtain a model more suitable for potentially infinite online processes.

After we tune the model to best fit our needs we shall analyze a number of problems from the point of view of advice complexity.

3.3.2 Second phase

In the second phase, we shall investigate the trade-off between the amount of added information about the input, and the solution quality in the problems studied in the first phase. There are two possible questions to ask: "*What competitive ratio can be achieved, if the communication with the oracle is limited to c bits per request?*" and, complementary, "*How much information has to be transmitted in order to achieve the competitive ratio k?*".

3.3.3 Timetable

In the first year, we consider still to deal with the development of suitable models for measuring the amount of information about the future requests, in the tradeoff with the achievable quality of solution. We consider also to think about the efficiency of the online procedures with respect to this tradeoff. The research in the first year will focus on the extremal scenario, where an almost optimal or even an optimal solution has to be found, and one has to find out the amount of knowledge about the future input that is necessary and sufficient for this purpose. In the last two years, we will consider the general scenario, investigating the tradeoff between the growth of the solution quality and the increase of the information amount.

3.3.4 Personnel

Prof. Dr. Juraj Hromkovič, Dr. Hans-Joachim Böckenhauer, Tobias Mömke PhD student, are involved in the project from our group. We consider to intensively cooperate with the groups of Prof. Dr. Rastislav Královič (Comenius University, Bratislava) and Prof. Dr. Georg Schnitger (University of Frankfurt a. M.) in this project. We have a few excellent candidates for PhD on this project and we consider to choose an appropriate one later.

3.4 Available resources for realization of the project

All the equipment required for the performance of the proposed project is present in the Information Technology and Education department.

3.5 List of own relevant publications

[1] H.-J. Böckenhauer and D. Bongartz. Protein folding in the HP model on grid lattices with diagonals (extended abstract). In *Proc. of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS 2004)*, volume 3153 of *Lecture Notes in Computer Science*, pages 227–238. Springer-Verlag, 2004.

[2] H.-J. Böckenhauer and D. Bongartz. Protein folding in the HP model on grid lattices with diagonals. *Discrete Applied Mathematics*, 155(2):230–256, 2007.

[3] H.-J. Böckenhauer, D. Bongartz, J. Hromkovič, R. Klasing, G. Proietti, S. Seibert, and W. Unger. On the hardness of constructing minimal 2-connected spanning subgraphs in complete graphs with sharpened triangle inequality. In *Proc. FSTTCS'02*, volume 2556 of *Lecture Notes in Computer Science*, pages 59–70. Springer-Verlag, 2002.

[4] H.-J. Böckenhauer, D. Bongartz, J. Hromkovič, R. Klasing, G. Proietti, S. Seibert, and W. Unger. k-edge-connectivity problems with sharpened triangle inequality. In *Proc. CIAC 2003*, volume 2653 of *Lecture Notes in Computer Science*, pages 189–200. Springer-Verlag, 2003.

[5] H.-J. Böckenhauer, D. Bongartz, J. Hromkovič, R. Klasing, G. Proietti, S. Seibert, and W. 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.

[6] H.-J. Böckenhauer, L. Forlizzi, J. Hromkovič, J. Kneis, J. Kupke, G. Proietti, and P. Widmayer. Reusing optimal TSP solutions for locally modified input instances. In *Proc. IFIP TCS*, 2006.

[7] H.-J. Böckenhauer, J. Hromkovič, R. Klasing, S. Seibert, and W. Unger. Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. In *Algorithms and Complexity, Proc. CIAC 2000*, volume 1767 of *Lecture Notes in Computer Science*, pages 72–86. Springer-Verlag, 2000.

[8] H.-J. Böckenhauer, J. Hromkovič, R. Klasing, S. Seibert, and W. Unger. Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. *Theoretical Computer Science*, 285:3–24, 2002.

[9] H.-J. Böckenhauer, J. Hromkovič, J. Kneis, and J. Kupke. On the approximation hardness of some generalizations of TSP. In *Proc. SWAT'06*, Lecture Notes in Computer Science, pages 184–195. Springer-Verlag, 2006.

[10] H.-J. Böckenhauer, J. Hromkovič, J. Kneis, and J. Kupke. On the parametrized approximability of TSP with deadlines. *Theory of Computing Systems*, to appear.

[11] H.-J. Böckenhauer, J. Hromkovič, and S. Seibert. Stability of approximation. In T. F. Gonzalez, editor, *Handbook on Approximation Algorithms and Metaheuristics*, chapter 31. Chapman & Hall/CRC, 2007.

[12] S. Dobrev, R. Královič, and D. Pardubská. How much information about the future is needed? *submitted manuscript*, 2007.

[13] L. Forlizzi, J. Hromkovič, G. Proietti, and S. Seibert. On the stability of approximation for Hamiltonian path problems. In *Proc. SOFSEM 2005*, Lecture Notes in Computer Science, pages 147–156. Springer-Verlag, 2005.

[14] L. Forlizzi, J. Hromkovič, G. Proietti, and S. Seibert. On the stability of approximation for Hamiltonian path problems. *Algorithmic Operations Research*, 1:20–34, 2006.

[15] J. Hromkovič. *Algorithmics for Hard Problems (Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics)*. Springer-Verlag, second corrected edition, 2004.

[16] J. Hromkovič. Stability of approximation in discrete optimization. In *World Computer Congress 2004, Proc. IFIP TCS*, pages 3–18. Kluwer, 2004.

[17] J. Hromkovič, T. Mömke, K. Steinhöfel, and P. Widmayer. Job shop scheduling with unit length tasks: Bounds and algorithms. *Algorithmic Operations Research*, 2, 2007.

[18] J. Hromkovič, K. Steinhöfel, and P. Widmayer. Job shop scheduling with unit length tasks: Bounds and algorithms. In *Proc. ICTCS'01*, volume 2202 of *Lecture Notes in Computer Science*, pages 90–106. Springer-Verlag, 2001.

[19] T. Mömke. On the power of randomization for job shop scheduling with k-units length tasks. In L. Matyska, editor, *Proc. 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2006)*, pages 121–128. Brno University of Technology, 2006.

[20] S. Seibert and W. Unger. A 1.5-approximation of the minimal Manhattan network problem. In *Proc. ISAAC 2005*, pages 246–255, 2005.

3.6 Relevant publications of other authors

[21] D. Achlioptas, M. Chrobak, and J. Noga. Competive analysis of randomized paging algorithms. In J. Díaz and M. J. Serna, editors, *ESA*, volume 1136 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 1996.

[22] S. Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, 1997.

[23] S. Albers. Improved randomized on-line algorithms for the list update problem. *SIAM Journal on Computing*, 27:670–681, 1998.

[24] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1-2):3–26, 2003.

[25] S. Albers, B. von Stengel, and R. Werchner. A combined bit and timestamp algorithm for the list update problem. *Information Processing Letters*, 56:135–139, 1995.

[26] D. Angluin, J. Westbrook, and W. Zhu. Robot navigation with range queries. In *Proc. 28th ACM Symposium on the Theory of Computing*, pages 469–478, 1996.

[27] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Annual ACM Symp. on the Theory of Computing*, pages 623–631, 1993.

[28] B. Awerbuch and Y. A. and. Y. Bartal. On-line generalized steiner tree problem. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 68–74, 1996.

[29] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *34th IEEE Symp. on Foundations of Computer Science*, pages 32–40, 1993.

[30] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. *Inf. Comput.*, 185(1):1–40, 2003.

[31] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management (extended abstract). In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 39–50, New York, NY, USA, 1992. ACM Press.

[32] A. Beck. On the linear search problem. *Israel J. Math.*, 2:221228, 1964.

[33] L. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

[34] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.

[35] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11:2–14, 1994.

[36] P. Berman and C. Coulston. Online algorithms for steiner tree problems. In *Proc. 29th Annual ACM Symposium on Theory of Computing*, pages 344–353, 1997.

[37] M. Betke, R. Rivest, and M. Singh. Piecemeal learning of an unknown environment. In *Proc. 5th Conference on Computational Learning Theory*, pages 277–286, 1993.

[38] A. Blum, P. Raghavan, and B. Scheiber. Navigating in unfamiliar geometric terrain. In *Proc. 23th ACM Symposium on the Theory of Computing*, pages 494–504, 1991.

[39] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, New York, NY, USA, 1998.

[40] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. In *Selected papers of the 23rd annual ACM symposium on Theory of computing*, pages 244–258, Orlando, FL, USA, 1995. Academic Press, Inc.

[41] J. Boyar, M. R. Ehmsen, and K. S. Larsen. Theoretical evidence for the superiority of lru-2 over lru for the paging problem. In T. Erlebach and C. Kaklamanis, editors, *WAOA*, volume 4368 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 2006.

[42] J. Boyar and L. M. Favrholdt. The relative worst order ratio for online algorithms. *ACM Trans. Algorithms*, 3(2):22, 2007.

[43] J. Boyar, K. S. Larsen, and M. N. Nielsen. The accommodating function: A generalization of the competitive ratio. *SIAM J. Comput.*, 31(1):233–258, 2001.

[44] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment. *Journal of the ACM*, 45:215–245, 1998.

[45] M. Englert and M. Westermann. Lower and upper bounds on fifo buffer management in qos switches. In Y. Azar and T. Erlebach, editors, *ESA*, volume 4168 of *Lecture Notes in Computer Science*, pages 352–363. Springer, 2006.

[46] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.

[47] P. Fraigniaud, C. Gavoille, D. Ilcinkas, and A. Pelc. Distributed computing with advice: Information sensitivity of graph coloring. In 34^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of Lecture Notes in Computer Science. Springer, July 2007.

[48] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Oracle size: a new measure of difficulty for communication tasks. In E. Ruppert and D. Malkhi, editors, *PODC*, pages 179–187. ACM, 2006.

[49] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Tree exploration with an oracle. In R. Kralovic and P. Urzyczyn, editors, *MFCS*, volume 4162 of *Lecture Notes in Computer Science*, pages 24–37. Springer, 2006.

[50] R. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 46:1563–1581, 1966.

[51] M. Imase and B. Waxman. Dynamic steiner tree problems. *SIAM Journal on Discrete Mathematics*, 4:349–384, 1991.

[52] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11:53–62, 1994.

[53] S. Irani and A. Karlin. Online computation. In *Approximation Algorithms for NP-complete Problems*, pages 521–559. PWS Publishing Company, 1997.

[54] S. Irani, A. R. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 228–236, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.

[55] B. Kalyanasundaram and K. Pruhs. Online wieghted matching. *Journal of Algorithms*, 14:139–155, 1993.

[56] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance [scheduling problems]. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 214, Washington, DC, USA, 1995. IEEE Computer Society.

[57] A. Karlin, M. Manasse, L. Rudolph, and D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[58] A. Karlin, S. Phillips, and P. Raghavan. Markov paging. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, pages 208–217, 1992.

[59] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for online bipartite matching. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 352–358, 1990.

[60] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1*, pages 416–429, Amsterdam, The Netherlands, The Netherlands, 1992. North-Holland Publishing Co.

[61] S. Khuller, S. Mitchell, and V. Vazirani. On-line weighted bipartite matching. In *Proc. 18th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 510 of *Springer LNCS*, pages 728–738, 1991.

[62] H. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM Journal on Discrete Mathematics*, 1:526–530, 1988.

[63] J. Kleinberg. On-line search in a simple polygon. In *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 8–15, 1994.

[64] E. Koutsoupias and C. H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):394–400, 2000.

[65] S. Leonardi. On-line network routing. In A. Fiat and G. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Springer LNCS*, pages 242–267, 1998.

[66] Z. Lotker and B. Patt-Shamir. Nearly optimal fifo buffer management for diffserv. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 134–143, New York, NY, USA, 2002. ACM Press.

[67] L. Lovasz, M. Saks, and M. Trotter. An online graph coloring algorithm with sublinear performance ratio. *Discrete Mathematics*, 75:319–325, 1989.

[68] C. Lund, S. Phillips, and N. Reingold. Ip over connection-oriented networks and distributional paging. In *Proc.of the 35th IEEE Symposium on Foundations of Computer Science*, pages 424–435, 1994.

[69] M. Manasse, L. McGeoch, and D. Sleator. Competitive algorithms for on-line problems. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 322–333, New York, NY, USA, 1988. ACM Press.

[70] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

[71] U.-M. O'Reilly and N. Santoro. The expressiveness of silence: Tight bounds for synchronous communication of information using bits and silence. In E. W. Mayr, editor, *WG*, volume 657 of *Lecture Notes in Computer Science*, pages 321–332. Springer, 1992.

[72] G. Pandurangan and E. Upfal. Can entropy characterize performance of online algorithms?. In *SODA*, pages 727–734, 2001.

[73] G. Pandurangan and E. Upfal. Entropy-based bounds for online algorithms. *ACM Transactions on Algorithms*, 3(1), 2007.

[74] Phillips and Westbrook. On-line algorithms: Competitive analysis and beyond. In *Algorithms and Theory of Computation Handbook*. CRC Press, 1999.

[75] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 140–149, New York, NY, USA, 1997. ACM Press.

[76] P. Raghavan. A statistical adversary for on-line algorithms. volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 79–85, 1992.

[77] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, Feb. 1985.

[78] B. Teia. A lower bound for randomized list update algorithms. *Information Processing Letters*, 47:5–9, 1993.

[79] E. Torng. A unified analysis of paging and caching. *Algorithmica*, 20(2):175–200, 1998.

[80] S. Vishwanathan. Randomized online graph coloring. *Journal of Algorithms*, 13:657–669, 1992.

[81] V.J.Lumensky and A.A.Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[82] J. Westbrook and D. Yan. Lazy and greedy: On-line algorithms for steiner problems. In *Proc. Workshop on Algorithms and Data Structures*, volume 709 of *Springer LNCS*, pages 622–633, 1993.

[83] A.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 17th Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[84] N. E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.

3.7 Significance of the project to ETH

To develop algorithms for hard computational problems and to measure the hardness of tasks with respect to their computational complexity is the central topic of research in the computer science fundamentals. ETH is one of the leading institutions in the world in algorithmics. This research proposal fits very well into this part of the image of computer science at ETH. Additionally, the results of this project can be useful in applications, which are intensively investigated in operation research, too.

3.8 Explanation for the material costs

No material is requested within the scope of the project.

3.9 Further information

none