The Distributed Computing Column

BY

Panagiota Fatourou

Department of Computer Science, University of Crete
P.O. Box 2208 GR-714 09 Heraklion, Crete, Greece
and
Institute of Computer Science (ICS)
Foundation for Research and Technology (FORTH)
N. Plastira 100. Vassilika Vouton
GR-700 13 Heraklion, Crete, Greece
faturu@csd.uoc.gr

Bulletin of the EATCS no 110, pp. 35–35, June 2013

Computing with Advice: when Knowledge Helps

Stefan Dobrev
Slovak Academy of Sciences, Bratislava, Slovakia.
Stefan.Dobrev@savba.sk

Rastislav Královič
Comenius University, Bratislava, Slovakia.
kralovic@dcs.fmph.uniba.sk

Richard Královič
ETH Zurich, Switzerland.
Google Zurich, Switzerland.
riso@google.com

Abstract

In several areas of computer science the possibility and efficiency of the solution is determined by information that is not accessible to the algorithm. Traditionally, a qualitative approach to the study of this information has been pursued, in which the impact of enhancing the algorithm with various specific types of information has been studied. Recently, a number of authors have proposed a quantitative approach, where the amount of the added information is studied in relation with the improvement of the quality or efficiency of the solution. We survey several recent examples of this approach from the area of distributed and online computing.

1 Introduction

From a high-level point of view, computation is usually thought of as information processing: The input instance contains some implicit, hidden information, and the role of the algorithm is to make this information explicit, which usually

means to produce some specified form of output. While from the information-theoretic point of view, all the relevant information is contained in the input, there may be several reasons why some of this information is not available to the algorithm. First, it may not be possible at all to obtain the required information in the given model of computation, or the algorithm may have limited resources that prevent it from extracting the required information. The questions how the bounded resources relate to the possibility to extract information are among the most notoriously difficult problems of computer science.

There are some cases, however, where information is inaccessible to the algorithm not due to some limited computational resources, but from the nature of the setting. A prominent example is the area of distributed computing where the global state of the system is usually not known to the computing entities, yet it often plays a crucial role in the efficiency (or even feasibility) of the solution. Indeed, many works have been studying the impact of knowledge of the network topology on the efficiency and feasibility of various distributed tasks. Other pieces of information that influence the distributed algorithm are the knowledge of (some) identifiers, and, possibly, the knowledge of failure patterns. In Sections 2 and 3, we survey the results that analyze the impact of topology knowledge from a quantitative point of view: how much information has to be supplied to the computing entities in order to be able to solve tasks efficiently. A similar situation to the distributed computing comes to play in another field of computer science: online algorithms. Here, the algorithm must make irreversible decisions based only on partial knowledge about the input, and coping with the fact that the yet unknown remainder of the input may be crucial for the solution. Again, there is an extensive research concerning the augmentation of the algorithm with some a-priori information about the input, an approach known as semi-online algorithms. Here, too, recently effort has been made to analyze the additional information in a quantitative way, and we survey some of the recent results in Section 4.

Results from both areas show complex behavior of the relationship between the amount of the additional information and the increase of the solution quality: in some cases, there are trade-off relations, where increasing the knowledge gives better solutions, on the other hand, examples of threshold values are known, where adding more bits of advice does not help.

2 Message based distributed computing

Let us consider distributed systems consisting of independent entities connected in a network that can communicate by some form of exchange of messages. There

are two basic views on such systems, which are in essence equivalent, but yield themselves to different types of questions – either the active components are the nodes of the network, and messages are pieces of data send among them, or the active components are the messages (agents) that traverse the network, and the nodes passively provide resources for computation and communication. Typical problems solved in the message-based systems include communication tasks such as broadcasting, wake-up, leader election, or computational problems where some graph-theoretic objects are to be constructed, like, e. g., various spanners, colorings, independent or dominating sets, etc. On the other hand, typical problems tackled in the agent-based view include many variants of graph exploration, map drawing, agent rendezvous, and similar.

The quantitative study of the topological information in message based systems was introduced in the work of Fraigniaud, Ilcinkas, and Pelc [49], where a distinction has been shown between two similar problems: broadcasting, and wakeup. Both problems are considered in an asynchronous setting where nodes have distinct identities, messages are delivered between neighboring pairs of nodes, and in both problems there is an initiator that starts with a message that must be delivered to all other nodes. The distinction is that in the broadcast problem, control messages may be spontaneously sent among vertices from the beginning of the algorithm, whereas in the wakeup problem, only vertices that have previously received a message may send a message (except for the initiator). While it has been known that without any information about the topology, the wakeup requires $\Omega(m)$ messages on a graph with n vertices, and m edges [5], in specific topologies (e. g., [28, 35]) wakeup can be done using $O(n)$ messages. $O(n)$ messages are also sufficient when the network is equipped with the sense of direction [42]. In [49] the authors model the topological information in the following way: a-priori, each node knows its identity, and the local labeling of incident edges. Before the algorithm starts, each node v is provided with a binary string $f(v)$. The function $f : V \mapsto \{0,1\}^*$ is called an oracle, and $\sum_{v \in V} |f(v)|$ is its size. The smallest number of messages, over all oracles of a given size, exchanged by the algorithm is considered as a complexity measure. It is shown that an oracle of size $\Theta(n \log n)$ is needed to perform wakeup with linearly many messages, while linear broadcast can be accomplished with an oracle of size $\Theta(n)$.

The same notion of oracle size has been addressed for a number of other problems in the synchronous setting. Fusco and Pelc [52] consider wakeup in a rooted tree in the one-port model, where each node may send in each step only one message, and the aim is to minimize the number of steps. To evaluate the algorithm they use the competitive analysis. The competitive ratio is the ratio of the broadcasting time of the algorithm with oracle of size q, to the optimal (offline) algorithm with full topological knowledge. The main result shows that with linear-sized oracle,

the broadcasting can be done optimally, and for $\sqrt{n} \leq q \leq n$ the competitive ratio is between $\Omega(n^{1-\varepsilon}/q)$ and $O(n \log^2 n/q)$ for arbitrary small ε. However, advice smaller than $\sqrt{n}$ does not help, since for $q < \sqrt{n}$, the competitive ratio is $\Theta(\sqrt{n})$, the same as without any advice.

A similar situation (in the $\mathcal{LOCAL}$ model from [75],i. e.,, in a synchronous message-passing system with nodes that have unique identifiers) where adding advice does not help has been observed in [47], where the time needed for proper 3-coloring of cycles and trees is investigated. Without any advice, cycles and oriented trees can be 3-colored in time $O(\log^* n)$, and oracle of size $\Omega(n/\log^{(k)} n)$ for any constant k is needed to beat the $O(\log^* n)$ bound, where $\log^{(k)} n$ is the k-th iteration of the logarithm. Moreover, for unoriented trees, the same oracle size is needed for 3-coloring in time $\Theta(\log^* n)$; almost as much information as specifying the color for each node.

In the $\mathcal{LOCAL}$ model, the construction of minimum spanning tree (MST) has been studied as well: in [51], authors consider a setting where each node has access to the weights of incident links, and the goal is to find a distributed representation of a minimum spanning tree. Instead of the overall length of advice strings, they studied the maximum, over all nodes. They show that with constant advice in each vertex, the MST can be constructed in logarithmic time, whereas without any advice, $\Omega(\sqrt{n}/\log n)$ rounds are needed [76].

Broadcasting in radio networks has been considered in [57], where a trade-off between the size of advice and broadcasting time has been devised.

Finally, let us note that the above mentioned work is tightly connected with the study of informative labeling schemes (see, e. g., [20, 22, 45, 62, 63, 64] and references therein): here, the aim is to label vertices of the graph in such a way that it is possible to extract, based solely on the labels of a subset of vertices $V' \subseteq V$ some parameter concerning V' (e. g., if V' is any two-element set, and the parameter is distance, the scheme is called distance labelling scheme).

3 Agent based distributed computing

Alternatively to the model of distributed systems with active nodes that communicate by exchanging messages, one can consider systems where the nodes are passive, and the computation is driven by active messages (agents). The most studied problems in this setting comprise various variants of graph exploration: the agents have to collaboratively explore the network, with possible goals including visiting all vertices, drawing a map, etc. It is always assumed that the incident links in each node are locally distinguishable; in some cases, the nodes

may have also unique identifiers.

The problems related to graph exploration are presumably the oldest graph-theoretic problems (e. g., [39]). The first studied variants concerned a single agent with full topology knowledge, and were focused on the existence of various types of walks. Supposedly the first algorithm for traversing unknown graphs is due to Shannon [80]. In the late 70ties, attention has turned to the problem of a finite automaton navigating in an unknown graph (e. g., [3, 17, 18]), which developed into a series of results concerning the size of memory, and the number of moves of the agent needed for successful exploration (e. g., [48, 72, 73]). At the same time, extensive research has been conducted on teams of cooperating agents (see, e. g., [24, 37, 46]). If not explicitly mentioned, we shall consider undirected graphs (i. e., the agent can always return along the link it arrived). Directed graphs have been treated, e. g., in [2, 8, 25, 41]. Apart from the various variants of graph exploration, problems like rendezvous (e. g., [7, 19, 27, 66]) or black hole search (e. g., [23, 29, 30]) have been investigated.

When considering the additional information, and how it affects the exploration, one should note that the local labelling of the incident links is a potential source of information. On the one hand, if the agent has no means to locally distinguish the incident links in a node, the exploration process can no longer be deterministic, and the adversary may force the agent to traverse a single edge back and forth. Assigning local labels to the incident links in every node is a natural way to circumvent this problem; another approach that is used in some cases is to assign labels to nodes, and to allow the agent to see the labels of neighboring nodes. When using the model with local link identifiers, it is assumed that the labeling is chosen by an adversary, and the agent(s) must be able to perform the task under any labelling. A series of papers [31, 53, 56, 65, 82] investigates how the properly chosen labeling may help the algorithm. It is proven that a memoryless agent (e. g., one using a right-hand-on-the-wall rule) can perform a fast periodic exploration of the network when the local port labels are set appropriately.

The oracle-based approach where additional advice strings can be placed in the nodes was applied in [21], where it is proven that 2 bits in every node are sufficient for a finite automaton to explore all graphs, a task that is not possible without any information.

In the problem of drawing a map of an unlabeled graph, investigated in [26], the symmetry of the graph plays a crucial role: there is a graph invariant called multiplicity (μ), which in some sense expresses the symmetry properties of the graphs, such that for graphs with $\mu = 1$, oracle of size $\varphi(n)$ is sufficient for any function $\varphi = \omega(1)$. On the other hand, for graphs with $\mu > 1$, oracle size $\Theta(m \log \mu)$ is needed, where m is the number of edges. Again, without any information, the task

is not solvable.

In [50], the following problem has been investigated: the agent, starting from a node v, has to traverse all edges of an unknown tree. Obviously, with full information about the tree, this can be done in an optimal number of moves $\texttt{Opt} = 2(n-1) - ecc(v)$ where $ecc(v)$ is the eccentricity of the starting node, i. e., the longest distance from v to another vertex w. For an algorithm $\texttt{A}$, the authors consider the competitive ratio, i. e., the ratio $\text{cost}(\texttt{A})/\texttt{Opt}$. It can be seen that without any further information, the best possible ratio attainable is 2. In order to avoid problems with information given in the port labeling, they use an oracle of the form $f : T \mapsto \{0,1\}^*$, where T is an unlabeled tree T (i. e., the advice given to the agent by the oracle is a bit string that is the same for all labellings of a given tree). The main result shows a tight bound of $\log\log D$ bits in order to get competitive ratio better than 2.

As we already mentioned, there are alternatives to the local port labeling. In the so-called fixed graph scenario introduced by Kalyanasundaram and Pruhs in [59], the nodes have identifiers, and when the agent arrives at a node $v \in G$, it learns all incident edges, their endpoints, and, if the graph is weighted, their weights. While learning the endpoints of the incident edges is stronger than the typical exploration scenario, it does have a justification (see [59] and [69]); it also corresponds to the previously studied neighbourhood sense of direction [43].

In [32], the following problem was addressed from the point of view of advice size: the agent starts at a node v of an undirected labeled graph with n nodes, where each edge has a non-negative cost. The agent has no knowledge about the graph, and has to visit every node of the graph and return to v. The agent can move only along the edges, each time paying the respective edge cost. Clearly, the optimum corresponds to the minimum travelling salesman route (TSP) on the metric closure of the graph (since it is allowed to visit a node more than once). A simple and fast heuristic for the traditional offline setting which has been extensively studied is the greedy algorithm Nearest Neighbor (NN): Once at a node u, go to the closest yet unexplored node, and repeat the process until all nodes have been explored. This algorithm can be also performed by an agent, achieving a competitive ratio of $\Theta(\log n)$ ([78]), which is tight even on planar unit-weight graphs ([55]). Despite many partial results ([4, 59, 70, 69]), the main question, whether there exists a constant-competitive algorithm is still open: the best known lower bound on the competitive ratio is $5/2 - \varepsilon$ ([32]), and the upper bound for general graphs is $O(\log n)$. Moreover, in [32] the authors were able to obtain constant competitive ratio with an advice of size $O(n)$ (for optimality, $\Omega(n \log n)$ bits are needed). Reducing the advice of a constant-competitive algorithm to $o(n)$ bits remains an open problem.

The same concept of advice has been also applied to the graph searching problem in [71].

4 Online Computing

In distributed computing, the information that is not known to the algorithm is the information about the topology of the network. The term online is used for problems where the input comes in parts, and the algorithm must produce output in an incremental fashion, too. Formally, the input $\mathbf{x}$ is a sequence of requests $\mathbf{x} = (x_1, \ldots, x_n)$. The output $\mathbf{y}$ is a sequence of answers $\mathbf{y} = (y_1, \ldots, y_n)$ computed by the algorithm in such a way that each y_i is a function of $x_1, \ldots, x_i$ (for randomized algorithms, it is also a function of the random bits used so far). The goal is to maximize or minimize a cost function defined over the whole output $\mathbf{y}$. For an exposition to online algorithms, we refer the reader to [15].

An archetypal online problem is paging, where the algorithm has to maintain a buffer of k items (pages). The input is a sequence of pages; when a page is requested that is in the buffer, the page is served, and no output is produced. However, if the page is not in memory, the algorithm must select a victim that is removed from the buffer, and is replaced by the requested page; this is called page fault, and the algorithm pays a penalty of 1 for each fault.

The notion of a competitive ratio was introduced by Sleator and Tarjan [81]: a minimization (analogous definition is for maximization) algorithm $\mathtt{A}$ is called c-competitive, if it always produces an output where $\text{cost}(\mathtt{A}) \leq c \cdot \mathtt{Opt} + \alpha$ for some constant α (sometimes, it is required that $\alpha = 0$; this requirement is termed strong competitiveness). Note that in online problems the main concern is not the computational complexity, but the inherent loss of performance due to the unknown future.

Online computation has received considerable attention over the past decades as a natural way of modeling real-time processing of data. A classical result from [81] states that no deterministic paging algorithm can be better than k-competitive. In general, since the algorithm does not know the future input, and because it is compared to the offline optimum in the worst case, many problems have no good competitive algorithms. In order to make the situation less unfair for the algorithm, randomization is often employed. Here, the algorithm has additional access to a random string. In order to be c-competitive, it is sufficient that $\mathrm{E}[\text{cost}(\mathtt{A})] \leq c \cdot \mathtt{Opt} + \alpha$ where the expectation is taken over all random strings. Another well known result [40] states that for paging, randomization helps, since the randomized paging is $\Theta(\log k)$-competitive. The help of randomization in the

case of paging comes, intuitively, from the fact that a particular mistake creates a single page fault: the algorithm pays for it, but it is easy to correct it later. In other cases, many decisions are critical: Consider, e. g., a problem when vertices of a graph are revealed one by one, and the algorithm has to select the largest possible independent set from the resulting graph, while seeing, at each time step, the subgraph induced by the arrived vertices. Clearly, there are situations, where incorrectly selecting a particular vertex prohibits the algorithm from selecting any other vertex for the rest of the input. Hence, it is not difficult to construct a graph and a vertex arrival sequence, where the randomized algorithm selects an expected constant number of vertices, while the graph admits a linearly-sized independent set.

In a way similar to the area of distributed computing, many results have been proven about enhancing the algorithm with a particular type of information about the input. The method of access graphs [16, 58] restricts the sequence of requests to be a walk in an a-priori known graph. A similar approach using entropy has been taken in [74]. Lookahead (e. g., [1]) reveals to the algorithm some limited number of future requests. In many scheduling and graph problems, specific forms of the input sequence have been considered (graphs with certain parameters, jobs arriving in certain order, etc.).

The first attempt to analyze the impact of added information quantitatively was due to Halldórsson et al [54]. The authors considered the problem of finding the maximum independent set online, and introduced a model where the algorithm can maintain a set of solutions. The final solution produced by the algorithm is the best one from the set at the time of the last input request. If the algorithm is allowed to maintain $r(n)$ solutions, this model can be interpreted as running the algorithm with $\log r(n)$ bits of advice describing the particular input. The results of the paper show that when $r(n)$ is constant, the competitive ratio is $\Omega(n)$, i. e., constant advice does not help. However, when $r(n)$ is polynomial, the competitive ratio is $\Theta(n/\log n)$.

In [34], the authors start a systematic quantitative treatment of the problem-specific information. In the proposed model, the algorithm received, in each step, a (possibly empty) advice string, and the advice complexity was defined as the sum of lengths of these strings. However, this model suffered from the fact that information was encoded also in the empty requests, as pointed out in [38]. Here, the authors used an advice string of fixed length attached to each request, and studied the advice complexity of k-server and metrical task system problems. Using this approach, however, it is not possible to analyze information that is sublinear in the number of requests. Therefore, in the model from [13], the whole advice is given to the algorithm at the beginning as a single binary string. In this way, the

model is equivalent to both the model from [54], and the model from agent based systems where the advice is given to the agent. Moreover, it forms an analogy to the model of randomized algorithms – instead of a string of random bits and the expected outcome, the string of best possible bits and the corresponding outcome is considered. Hence, the comparison between randomization and advice is attractive. In the remainder of this section we shall consider the model from [13].

Obviously, there are two trivial ways to obtain an optimal algorithm: either to encode the whole input in the advice (recall that in the treatment of online problems, computational resources are usually disregarded), or to encode the whole output; hence the advice is upper bounded by the minimum of Kolmogorov complexity of those two. In certain cases, however, significantly lower advice is sufficient. A number of problems have been considered in this model, including paging [13], k-server [12], knapsack [14], set cover [61], metrical task systems [38] (the results from the paper hold in both models), buffer management [36], job shop scheduling [13], independent sets in various classes of graphs (general graphs [54], interval graphs [13], bipartite graphs [33]), and various variants of online coloring (bipartite graphs [9], paths [44], 3-colorable graphs [79], $L(2,1)$ coloring [10]).

In general, there are three questions that are usually asked about a problem:

- What advice is needed to get optimal solution?
- What advice is needed to get the competitive ratio of the best possible randomized algorithm?
- What is the relationship between the size of the advice and the competitive ratio?

Usually, the first question is the easiest one to answer, and it turns out that for many problems, large information is needed to be optimal. However, even this large information is sometimes smaller that the trivial bound. For the paging problem, e. g., $O(n)$ bits of advice are sufficient to obtain an optimal algorithm (see [13]) in the following way: with every page in the buffer, the algorithm stores a flag indicating whether the page will be used by a reference optimal algorithm before replacement. When a page fault occurs, the algorithm is safe to remove any page that will not be needed by the optimal solution. The new page is inserted into the cache, and the new flag is read from the advice. On the other hand, to encode the input or the output, $\Omega(n \log k)$ bits would be needed.

The comparison of advice and randomization is an interesting point, since the two approaches use different properties of the solution space: to have a randomized algorithm with good expected performance, many good witnesses for each input

instance are needed. On the other hand, for good performance of advice algorithms, only one witness is sufficient for a given instance, but the space of possible witnesses must be small. In general, it holds (see [12] for minimization problems; analogous statement holds for maximization) that if there is a randomized algorithm with expected worst case competitive ratio $E(n)$, then for any constant $\varepsilon > 0$ there is an algorithm with advice of $O(\log n + \log\log|\mathcal{I}(n)|)$ bits with competitive ratio $(1+\varepsilon)E(n)$, where $\mathcal{I}(n)$ is the set of all input instances consisting of n requests. However, in many cases significantly smaller advice is sufficient to be on par with randomization. For paging, e. g., $\log k$ bits (i. e., independent of n) of advice is sufficient to get competitive ratio $O(\log k)$ ([13]) which equals to the randomized competitive ratio. An interesting point is that an $O(\log k)$-competitive randomized algorithm can be obtained with $O(\log k)$ random bits ([60]), so the random bits and advice bits exhibit the same power in this case.

For an illustration, consider the k-server problem, which is a generalization of paging. In a metric space (finite or infinite) there are k servers located in some points of the space. Each request is some point x_i in the space. To fulfill the request, the algorithm must make sure that there is some server located on x_i; if it is not the case, some server must be moved there, paying the cost of the travelled distance. In the deterministic case, the competitive ratio is known to be $\Theta(k)$, and a famous conjecture states that the randomized competitive ratio is $\Theta(\log k)$. The closest in proving the conjecture is the breakthrough result from [6] that for a finite metric space with β points, there is a randomized k-server algorithm with expected competitive ratio $O(\log^2 k \log^3 \beta \log\log\beta)$. From that follows that there is an algorithm with advice $O(\log n + \log\log\beta)$ having the same competitive ratio. However, the algorithm from [12] runs in exponential time: it first simulates the randomized algorithm on all inputs and all possible random strings, and produces a dictionary of polynomial size. The advice is then a pointer to the dictionary. Existence of a polynomial-time algorithm for k-server that achieves competitive ratio $O(\log k)$ using $O(\log n)$ bits is an open problem.

The relationship between the size of the advice and the competitive ratio is a complex one. In some cases, a trade-off relation exists, where increasing the advice yields a better competitive ratio, as is, e. g., the case of constant competitive ratio of paging [13]. On the other hand, there are thresholds, where increasing the advice does not help, e. g., for simple knapsack [14], no algorithm using less than $\log n$ bits can be better than $(2-\varepsilon)$ competitive, but with $(3(\varepsilon+1)/\varepsilon)\log n + o(\log n)$ bits, competitive ratio $(1+\varepsilon)$ can be achieved for any constant ε.

Notable is also the approach from [11], where an artificial problem of string guessing is analyzed, and a reduction is used to prove lower bounds on the advice complexity of online set cover.

5 Conclusion

Recently, there have been several attempts to analyze the impact of the hidden information in a quantitative way. Although they are applied in different areas, they share a common framework: The algorithm is enhanced by some information about the unknown part of the input, which may be of any type, but of bounded size. This approach may deepen the understanding of the structure of the respective problems. Finally, we note that the term advice complexity has traditionally been used as a synonym for relativized complexity (i. e., a sequential computation where the Turing machine gets an advice that depends on the length of the input), which may cause some confusion. Also, we note similar approaches in the treatment of the problem of factorization ([68, 77]) where the number of queries to a yes/no oracle needed to determine the factors of a number was studied.

References

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

[2] S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000.

[3] H. Antelmann, L. Budach, and H.-A. Rollik. On universal traps. *Elektronische Informationsverarbeitung und Kybernetik*, 15(3):123–131, 1979.

[4] Y. Asahiro, E. Miyano, S. Miyazaki, and T. Yoshimuta. Weighted nearest neighbor algorithms for the graph exploration problem on cycles. *Information Processing Letters*, 110(3):93 – 98, 2010.

[5] B. Awerbuch, O. Goldreich, D. Peleg, and R. Vainish. A trade-off between information and communication in broadcast protocols. *J. ACM*, 37(2):238–256, 1990.

[6] N. Bansal, N. Buchbinder, A. Madry, and J. Naor. A polylogarithmic-competitive algorithm for the k-server problem. In R. Ostrovsky, editor, *FOCS*, pages 267–276. IEEE, 2011.

[7] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Rendezvous and election of mobile agents: Impact of sense of direction. *Theory Comput. Syst.*, 40(2):143–162, 2007.

[8] M. A. Bender and D. K. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *FOCS*, pages 75–85. IEEE Computer Society, 1994.

[9] M. P. Bianchi, H.-J. Böckenhauer, J. Hromkovič, and L. Keller. Online coloring of bipartite graphs with and without advice. In *Proc. of the 18th Annual International*

Conference on Computing and Combinatorics (COCOON 2012), volume 7434 of *Lecture Notes in Computer Science*, pages 519–530, 2012.

[10] M. P. Bianchi, H.-J. Böckenhauer, J. Hromkovič, S. Krug, and B. Steffen. On the advice complexity of the online $L(2,1)$-coloring problem on paths and cycles. In D. Du and G. Zhang, editors, *COCOON*, volume 7936 of *Lecture Notes in Computer Science*. Springer-Verlag, 2013. to appear.

[11] H.-J. Böckenhauer, J. Hromkovic, D. Komm, S. Krug, J. Smula, and A. Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. to appear, 2013.

[12] H.-J. Böckenhauer, D. Komm, R. Královič, and R. Královič. On the advice complexity of the k-server problem. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Proc. of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6755 of *Lecture Notes in Computer Science*, pages 207–218. Springer-Verlag, 2011.

[13] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In Y. Dong, D.-Z. Du, and O. H. Ibarra, editors, *Proc. of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, volume 5878 of *Lecture Notes in Computer Science*, pages 331–340. Springer-Verlag, 2009.

[14] H.-J. Böckenhauer, D. Komm, R. Královič, and P. Rossmanith. On the advice complexity of the knapsack problem. In D. Fernández-Baca, editor, *Proc. of the 10th Latin American Symposium on Theoretical Informatics (LATIN 2012)*, volume 7256 of *Lecture Notes in Computer Science*, pages 61–72. Springer-Verlag, 2012.

[15] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[16] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference (preliminary version). In C. Koutsougeras and J. S. Vitter, editors, *STOC*, pages 249–259. ACM, 1991.

[17] L. Budach. On the solution of the labyrinth problem for finite automata. *Elektronische Informationsverarbeitung und Kybernetik*, 11(10-12):661–672, 1975.

[18] L. Budach. Environments, labyrinths and automata. In *FCT*, pages 54–64, 1977.

[19] J. Chalopin, S. Das, and P. Widmayer. Rendezvous of mobile agents in directed graphs. In Lynch and Shvartsman [67], pages 282–296.

[20] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, Y. Vaxès, and Y. Xiang. Additive spanners and distance and routing labeling schemes for hyperbolic graphs. *Algorithmica*, 62(3-4):713–732, 2012.

[21] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-guided graph exploration by a finite automaton. *ACM Transactions on Algorithms*, 4(4), 2008.

[22] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Labeling schemes for tree representation. *Algorithmica*, 53(1):1–15, 2009.

[23] J. Czyzowicz, S. Dobrev, R. Královic, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In S. Kutten and J. Zerovnik, editors, *SIROCCO*, volume 5869 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2009.

[24] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro. Map construction of unknown graphs by multiple agents. *Theor. Comput. Sci.*, 385(1-3):34–48, 2007.

[25] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.

[26] D. Dereniowski and A. Pelc. Drawing maps with advice. In Lynch and Shvartsman [67], pages 328–342.

[27] A. Dessmark, P. Fraigniaud, D. R. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006.

[28] K. Diks, S. Dobrev, E. Kranakis, A. Pelc, and P. Ruzicka. Broadcasting in unlabeled hypercubes with a linear number of messages. *Inf. Process. Lett.*, 66(4):181–186, 1998.

[29] S. Dobrev, P. Flocchini, R. Kralovic, P. Ruzicka, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[30] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocols. *Distributed Computing*, 19(1), 2006.

[31] S. Dobrev, J. Jansson, K. Sadakane, and W.-K. Sung. Finding short right-hand-on-the-wall walks in graphs. In A. Pelc and M. Raynal, editors, *SIROCCO*, volume 3499 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2005.

[32] S. Dobrev, R. Královic, and E. Markou. Online graph exploration with advice. In G. Even and M. M. Halldórsson, editors, *SIROCCO*, volume 7355 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2012.

[33] S. Dobrev, R. Královič, and R. Královič. Independent set with advice: The impact of graph knowledge. In T. Erlebach and G. Persiano, editors, *WAOA*, Lecture Notes in Computer Science, page to appear. Springer, 2012.

[34] S. Dobrev, R. Královič, and D. Pardubská. Measuring the problem-relevant information in input. *RAIRO Theoretical Informatics and Applications*, 43(3):585–613, 2009.

[35] S. Dobrev and P. Ruzicka. Broadcasting on anonymous unoriented tori. In J. Hromkovic and O. Sýkora, editors, *WG*, volume 1517 of *Lecture Notes in Computer Science*, pages 50–62. Springer, 1998.

[36] R. Dorrigiv, M. He, and N. Zeh. On the advice complexity of buffer management. In K.-M. Chao, T. sheng Hsu, and D.-T. Lee, editors, *ISAAC*, volume 7676 of *Lecture Notes in Computer Science*, pages 136–145. Springer, 2012.

[37] M. Dynia, J. Lopuszanski, and C. Schindelhauer. Why robots need maps. In G. Prencipe and S. Zaks, editors, *SIROCCO*, volume 4474 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2007.

[38] Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.

[39] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Novi Commentarii Academiae Scientarium Imperialis Petropolitanque*, 7:9–28, 1758-59.

[40] 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.

[41] R. Fleischer and G. Trippen. Exploring an unknown graph efficiently. In G. S. Brodal and S. Leonardi, editors, *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 11–22. Springer, 2005.

[42] P. Flocchini, B. Mans, and N. Santoro. On the impact of sense of direction on message complexity. *Inf. Process. Lett.*, 63(1):23–31, 1997.

[43] P. Flocchini, B. Mans, and N. Santoro. Sense of direction in distributed computing. *Theor. Comput. Sci.*, 291(1):29–53, 2003.

[44] M. Forišek, L. Keller, and M. Steinová. Advice complexity of online coloring for paths. In *Proc. of the 6rd International Conference on Language and Automata Theory and Applications (LATA 2012)*, pages 228–239, 2012.

[45] P. Fraigniaud. Informative labeling schemes. In S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *ICALP (2)*, volume 6199 of *Lecture Notes in Computer Science*, page 1. Springer, 2010.

[46] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[47] P. Fraigniaud, C. Gavoille, D. Ilcinkas, and A. Pelc. Distributed computing with advice: information sensitivity of graph coloring. *Distributed Computing*, 21(6):395–403, 2009.

[48] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theor. Comput. Sci.*, 345(2-3):331–344, 2005.

[49] 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.

[50] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Tree exploration with advice. *Inf. Comput.*, 206(11):1276–1287, 2008.

[51] P. Fraigniaud, A. Korman, and E. Lebhar. Local mst computation with short advice. *Theory Comput. Syst.*, 47(4):920–933, 2010.

[52] E. G. Fusco and A. Pelc. Trade-offs between the size of advice and broadcasting time in trees. In F. Meyer auf der Heide and N. Shavit, editors, *SPAA*, pages 77–84. ACM, 2008.

[53] L. Gasieniec, R. Klasing, R. A. Martin, A. Navarra, and X. Zhang. Fast periodic graph exploration with constant memory. *J. Comput. Syst. Sci.*, 74(5):808–822, 2008.

[54] M. M. Halldórsson, K. Iwama, S. Miyazaki, and S. Taketomi. Online independent sets. *Theor. Comput. Sci.*, 289(2):953–962, 2002.

[55] C. A. Hurkens and G. J. Woeginger. On the nearest neighbor rule for the traveling salesman problem. *Operations Research Letters*, 32(1):1 – 4, 2004.

[56] D. Ilcinkas. Setting port numbers for fast graph exploration. *Theor. Comput. Sci.*, 401(1-3):236–242, 2008.

[57] D. Ilcinkas, D. R. Kowalski, and A. Pelc. Fast radio broadcasting with advice. *Theor. Comput. Sci.*, 411(14-15):1544–1557, 2010.

[58] S. Irani, A. R. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. In G. N. Frederickson, editor, *SODA*, pages 228–236. ACM/SIAM, 1992.

[59] B. Kalyanasundaram and K. R. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130(1):125 – 138, 1994.

[60] D. Komm and R. Královič. Advice complexity and barely random algorithms. In I. Černá, T. Gyimóthy, J. Hromkovič, K. G. Jeffery, R. Královič, M. Vukolic, and S. Wolf, editors, *Proc. of the 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2011)*, volume 6543 of *Lecture Notes in Computer Science*, pages 332–343. Springer-Verlag, 2011.

[61] D. Komm, R. Královič, and T. Mömke. On the advice complexity of the set cover problem. In E. A. Hirsch, J. Karhumäki, A. Lepistö, and M. Prilutskii, editors, *Proc. of the 7th Symposium on Computer Science in Russia (CSR 2012)*, volume 7353 of *Lecture Notes in Computer Science*, pages 241–252. Springer-Verlag, 2012.

[62] A. Korman. Labeling schemes for vertex connectivity. *ACM Transactions on Algorithms*, 6(2), 2010.

[63] A. Korman, S. Kutten, and D. Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

[64] A. Korman, D. Peleg, and Y. Rodeh. Constructing labeling schemes through universal matrices. *Algorithmica*, 57(4):641–652, 2010.

[65] A. Kosowski and A. Navarra. Graph decomposition for memoryless periodic exploration. *Algorithmica*, 63(1-2):26–38, 2012.

[66] D. R. Kowalski and A. Malinowski. How to meet in anonymous network. *Theor. Comput. Sci.*, 399(1-2):141–156, 2008.

[67] N. A. Lynch and A. A. Shvartsman, editors. *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*. Springer, 2010.

[68] U. M. Maurer. On the oracle complexity of factoring integers. *Computational Complexity*, 5(3/4):237–247, 1995.

[69] N. Megow, K. Mehlhorn, and P. Schweitzer. Online graph exploration: New results on old and new algorithms. In L. Aceto, M. Henzinger, and J. Sgall, editors, *ICALP (2)*, volume 6756 of *LNCS*, pages 478–489. Springer, 2011.

[70] S. Miyazaki, N. Morimoto, and Y. Okabe. The online graph exploration problem on restricted graphs. *IEICE Transactions on Information and Systems*, 92(9):1620–1627, 2009.

[71] N. Nisse and D. Soguet. Graph searching with advice. *Theoretical Computer Science*, 410(14):1307 – 1318, 2009.

[72] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *J. Algorithms*, 33(2):281–295, 1999.

[73] P. Panaite and A. Pelc. Impact of topographic information on graph exploration efficiency. *Networks*, 36(2):96–103, 2000.

[74] G. Pandurangan and E. Upfal. Can entropy characterize performance of online algorithms? In S. R. Kosaraju, editor, *SODA*, pages 727–734. ACM/SIAM, 2001.

[75] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000.

[76] D. Peleg and V. Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM J. Comput.*, 30(5):1427–1442, 2000.

[77] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[78] D. J. Rosenkrantz, R. E. Stearns, and P. M. L. II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[79] S. Seibert, A. Sprock, and W. Unger. Advice complexity of the online coloring problem. In *Proc. of the 8th International Conference on Algorithms and Complexity (CIAC 2013)*, volume to appear of *Lecture Notes in Computer Science*, page to appear. Springer-Verlag, 2013.

[80] C. E. Shannon. Presentation of a maze solving machine. In H. von Foerster, M. Mead, and H. L. Teuber, editors, *Cybernetics: Circular, Causal and Feedback Mechanisms in Biological and Social Systems, Transactions Eighth Conference, March 15–16, 1951, New York, NY*, pages 169–181, New York, NY, USA, 1951. Josiah Macy Jr. Foundation.

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

[82] M. Steinová. On the power of local orientations. In A. A. Shvartsman and P. Felber, editors, *SIROCCO*, volume 5058 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2008.