Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

NORBERT ASCHEUER MARTIN GRÖTSCHEL
NICOLA KAMIN JÖRG RAMBAU

Combinatorial Online Optimization in Practice

Preprint SC 98-07 (February 1998)

Combinatorial Online Optimization in Practice

Norbert Ascheuer * Martin Grötschel * Nicola Kamin † Jörg Rambau *

Berlin, February 25, 1998

Abstract

This paper gives a short introduction into combinatorial online optimization. It explains a few evaluation concepts of online algorithms, such as competitiveness, and discusses limitations in their application to real–world problems. The main focus, however, is a survey of combinatorial online problems coming up in practice, in particular, in large scale material flow and flexible manufacturing systems.

Keywords: Online optimization, combinatorial optimization, real–world problems

1 Introduction

In classical optimization, called *offline optimization* here, it is assumed that all input data of an instance are available before solution algorithms are applied. In many applications this is not realistic. Decisions have to be made before all data are known. Such situations are often termed *online*. They arise, in particular, in processes that are continuously running for a longer period of time. For instance, in material flow systems of companies transportation tasks arise throughout the day and decisions have to be made before all jobs have been generated.

Online optimization is the task of finding "good", or "cheap", or "economic" decisions in online situations. An *online algorithm* provides such decisions. In practical applications online algorithms are typically subject to additional constraints. For example, they have to answer in real time, or must process a job (or request) within a given time frame, sometimes they have access to limited computing resources only.

Competitiveness. A common concept to evaluate online algorithms is *competitiveness*. Here an online algorithm has to act as follows: it always has to serve a request of a sequence before the next request (or the next k different requests in a model with look–ahead) becomes visible. The idea behind the variants of this notion is the following: compare the solution of the online algorithm under consideration with the solution that some adversary would produce on the same set of data.

The easiest case is the offline adversary, i.e., to him the complete sequence of requests is known in advance. For an algorithm X let $C_X(\sigma)$ denote the cost X produces on input

*Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Takustr. 7, D–14195 Berlin Dahlem

†Konrad-Zuse-Zentrum für Informationstechnik Berlin(ZIB), Takustr. 7, D–14195 Berlin Dahlem,and Herlitz PBS AG, Berliner Str. 27, D–13507 Berlin

sequence σ. We assume for notational convenience that $C_X(\sigma)$ is positive for all σ and that we want to minimize. A deterministic online algorithm A is *c–competitive* if for any sequence σ of requests and any offline algorithm S

$$C_A(\sigma) \leq c \cdot C_S(\sigma) + a$$

holds for real numbers c, a, both not depending on σ.

The goal is to find online algorithms that are competitive in an optimal way, i.e., so that no other online algorithm can have a better performance ratio with the adversary. This concept is applicable both to deterministic and randomized algorithms. It allows for provable statements about the performance of an online algorithm. It has been the subject of many investigations concerning (mainly elementary) online problems. A further advantage of this concept is that no information about the distribution of the input data is needed in order to make exact statements. See, among others, Albers [3, 4], Goemans [18], Irani and Karlin [20], Motwani, Raghavan [24], and Ottmann et al. [25] for more information.

The competitiveness ratio is usually a pessimistic measure, because the adversary is supposed to be a bad person trying to fool the algorithm by designing a particularly difficult sequence of requests. Moreover, competitive analysis is based on some hard restrictions to the model: one assumes that no earlier than after a request is served the next request becomes available. In practice, however, there is often a dynamically growing and shrinking pool of requests of unknown size visible to the algorithm.

Sometimes competitive analysis provides absolutely no insight into the quality of an online algorithm. For instance, for a version of the greeting card commissioning problem to be discussed later, one can prove that all (reasonable) online algorithms have the same competitiveness factor K (the common capacity of the vehicles of the system), see Kamin [22].

Furthermore, the competitive analysis is not applicable if the decisions of the online algorithm have direct impact on the sequence of future requests.

Stochastic Optimization. Stochastic optimization employs a model that seems to be close to what we might call "reasonable acting under incomplete information". The decisions in the online algorithm are made based on the optimal solution of a *stochastic program* that has to be solved beforehand. Usually, this is a linear program where the objective function is the expectation of the cost function in the decision and request variables.

This can only be done under the assumption that the request data has a certain distribution. Making decisions beforehand relying on statistical data is rather an offline model of an online problem because the distribution allows for computing the expectations before the actual requests occur.[1] Stochastic analysis is—as the name might suggest anyway—focusing on the average behavior of the algorithm. See Prékopa [26] and Kall, Wallace [21] for more background and applications in this area.

Since stochastic analysis requires an idea about what the distribution of the incoming requests may look like, it is not applicable if one cannot find any structure in the input data. And: if the probability is not concentrated at the expectation then the decisions that correspond to solutions optimizing the expectation of the cost function may bear too large a risk of failure. Usually, there is no guarantee that the algorithm works well for any sequence of requests that might occur.

[1] There are also models that perform a certain number of alternating observation and optimization steps in order to adjust the information about the distribution. These models are, however, hard to evaluate in practice.

Simulation. This is the path practice follows. A coarse mathematical model of the online problem is developed and implemented in the form of a simulation model. Several online algorithms are coded and experiments with real–world data are run on a computer to gain insight into the practical performance. In particular, experiments are made to analyze high load and failure situations. Usually those online algorithms are selected for use in practice that exhibit good average performance and can somehow cope with "catastrophes".

A–posteriori Analysis. This approach uses history. The actual sequences that came up in the past are recorded and competitive analysis is made based on the gathered data only. This evaluation is often used to tune the online algorithms so that they perform better on these known input sequences. One hopes that the old input sequences are good representations of the typical data and that, thus, the modified algorithms will show improved performance also on future input sequences.

2 Application to real–world problems

Besides the theoretical attractiveness of online optimization problems there is a broad variety of real–world problems that can be modelled as an online problem. In the sequel we outline problems we encountered in joint projects with industry that were aimed to optimize the internal material flow within a flexible manufacturing system and a distribution center. We just like to mention that, among others, there exist further applications in computer science, vehicle routing, scheduling, and telecommunication that will not be discussed in this paper.

2.1 Online Optimization of a Flexible Manufacturing System (FMS)

Siemens Nixdorf Informationssysteme AG (SNI) maintains a production plant where all their personal computers (PCs) and related products are assembled. See Figure 1 for a sketch of the material flow within this FMS. Parts out of which the PCs are assembled (PCB, floppy disk, cables, etc.) enter the FMS at the receiving area in normed containers. They are brought by automatic guided vehicles into one of in total six automatic storage systems (AUSS). The AUSS serve as material buffer between the receiving area and the assembly lines located at each side of the AUSS. After assembly the PCs enter a test area where for up to 24 hours test programs are run in order to check the full functionality of the PCs. After a manual test the PCs are packed and delivered.

Optimization problems. A profound analysis of the system showed that it offers a variety of optimization problems, some of them of an online character. These mainly are:

- scheduling of transportation tasks within the AUSS,
- assignment of containers to storage locations,
- routing and scheduling of the AGV,
- assignment of locations in the test area,
- retrievals of PCs from the test area.

In the following the first question will be discussed in more detail. Discussions of the other topics can be found in [1, 6, 23]

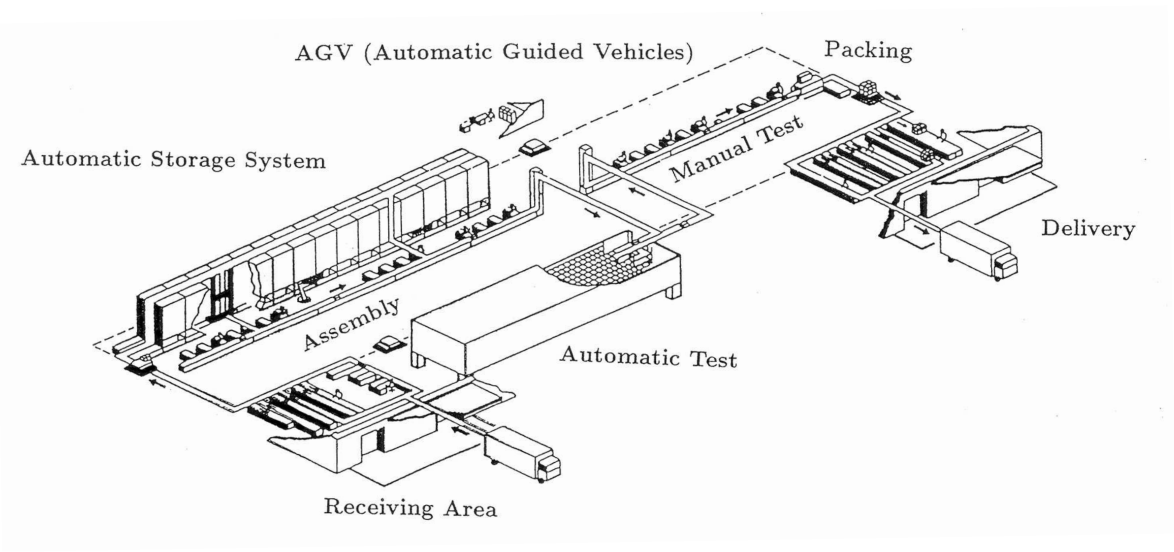

Figure 1 Sketch of the factory layout

2.1.1 Stacker Crane Routing in the Automatic Storage Systems

The automatic storage systems (AUSS) are single–aisled with storage locations on both sides of the aisle. In the lower part there are buffer places where containers are provided to the assembly line. A single stacker crane has to fulfill all transportation tasks (jobs). So far, to each task (storage, retrieval, buffer–refill, etc.) a certain priority was assigned that was only dependent on the type of the task. Within one priority class jobs were sequenced due to a FIFO–rule. Although easy to implement this strategy resulted in a high percentage of unloaded travel time.

Since every algorithm has to process all the jobs we can only control the unloaded moves of the stacker crane. We suggested to sequence the tasks in such a way such that the total time needed for the unloaded moves between the jobs is minimized. This can be modeled as an *asymmetric travelling salesman problem* (ATSP) where each unperformed job and the one that is currently processed by the stacker crane is represented by a node in a complete digraph $D = (V, A)$. W.l.o.g. we assume that the current job corresponds to node 1. Each arc $(i, j) \in A, j \neq 1$, represents the unloaded move between jobs i and j. This arc is weighted with the time needed for the unloaded move from the endpoint of job i to the starting point of job j. To all arcs $(i, 1), i \in V \setminus \{1\}$, we associate weight 0. Now, an optimal tour through the nodes of $D = (V, A)$ corresponds to a sequence of the transportation tasks with minimal total unloaded travel time.

This is an online problem since not all transportation tasks (resp. nodes for the ATSP) are known in advance. They are generated during the production period and neither generation time nor start– and end–coordinates are known in advance, i.e., we have to solve an *online–ATSP*. A detailed discussion of this topic can be found in Ascheuer [6]. See, e.g, Ausiello et al. [11, 12] for competitiveness results on the online ATSP.

Solution approach. We decided to simply ignore tasks that might be generated in the future and to solve a "static" ATSP as soon as a new job is generated. In order to avoid that the stacker crane has to wait until we have finished our calculations we have implemented a

3–phases process. Whenever a new job is generated we run the following optimization process:

Phase 1: Simple insertion heuristic.
Try to insert the new node as cheap as possible into the current sequence.

Phase 2: Run a more sophisticated heuristic.
We have chosen a random insertion heuristic.

Phase 3: Solve the ATSP to optimality.
This is done using a branch&bound–implementation of Fischetti and Toth [16].

Phase 1 runs in $\mathcal{O}(n)$ time and is always completed. For the typical problem sizes that occur in our application ($n \leq 60$) this is done in fractions of a second. Even phase 3 was always completed within a few seconds.

After the completion of each phase a sequence is available that can be improved by one of the subsequent phases. In case that during the execution of phase 2 or 3 a new job is generated the whole process is stopped and restarted. If the stacker crane has finished a task and asks for a new one the process is interrupted as well and the best sequence so far is passed to the control system of the stacker crane.

We tested several heuristics to be used in phase 2 (cf. [2]). It is easy to construct examples where it does not always lead to the best solution if each ATSP is solved to optimality. The use of this strategy might construct sequences that are "not good" with respect to the nodes generated in the future. Nevertheless, phase 3 empirically gives the best results on the average.

Computational results. SNI provided data for one week of production. During this period on one AUSS each generated task and each move of the stacker crane was recorded. That data was used to validate the simulation model. Based on the SNI–data we compared several strategies for sequencing the jobs within the simulation environment.

Extensive computational tests showed that it was possible to reduce the times needed for unloaded moves by approx. 30 % in heavy load periods. As a result this optimization package was put in use on five AUSS and the results were confirmed in everyday production. It showed that even in the production environment the optimization process could always finish with phase 3.

Quality of the online solutions. A scientific question that arises is how good are the solutions in comparison to an optimal offline solution. To evaluate the quality we performed an *a–posteriori analysis*, i.e., we determined how we would have sequenced the tasks if we had known which tasks were generated.

To this end we "collected" all jobs over a certain time period and sequenced them optimally. First note that the jobs cannot be sequenced earlier than they are generated. Moreover, the completion of the jobs cannot wait too long as, e.g., the production might be delayed. Thus, to each job a time window is associated and we only allow to visit a node within its time window (ATSP–TW) (see, among others, Desrochers et al. [14] and Desrosiers et al. [15]). The ATSP–TW is a difficult combinatorial optimization problem where it is even strongly $\mathcal{NP}$–complete to find a feasible solution [17, 27]. We have developed a branch&cut–approach for the ATSP–TW [7, 8] and a relaxation, namely the ATSP with precedence constraints [10]. The optimal solutions to these problems yield a lower bound to an optimal online strategy, if the same sequence of nodes is generated. Computational tests based on the production data from SNI showed that there is still an online optimality gap of 3–70 %, approx. 30 % on the average.

We like to point out one important restriction of this a–posteriori analysis. This analysis is based on the fact that the same sequence of nodes is generated, independent of the way the jobs are performed. This is not necessarily the case for this application as the completion of a certain job may have an influence on other generated tasks. For example, consider the case that a container delivered to the assembly line (task A) contains parts that are not usable (e.g., they are broken). As a result the workers generate a retrieval task B and order new parts (task C). The sooner task A is performed the earlier tasks B and C are generated and the earlier the time window for B and C will become active. Thus, there is no well defined *optimal offline solution* to which we can compare the online solution. As a consequence, competitive analysis cannot be applied.

2.2 Online–Optimization of a distribution center

The Herlitz PBS AG [19] is the main manufacturing firm for office supplies in Germany. They maintain their Europe–wide distribution center in Falkensee, close to Berlin. A joint project is aimed to efficiently manage their complete internal material flow. In a first phase we have optimized one commissioning area. We are currently working on the optimization of the whole pallet transportation system consisting of a system of roller conveyors, ten elevator systems and 18 automatic storage systems.

Online optimization questions arise in the following areas:

- routing of pallets,
- efficient control of elevator systems,
- routing within the automatic storage systems,
- routing of commissioning vehicles.

2.2.1 Commissioning of greeting cards

In this section we discuss one question in further detail, namely the efficient commissioning of greeting cards.

Description of the system. The cards are stored in four parallel shelving systems (see Figure 2). In accordance with the customers' orders, the different greeting cards have to be collected in boxes to be shipped to the customers. Order pickers on eight automated guided vehicles collect the orders from the storage systems while following a circular course. The vehicles are unable to pass each other. Moreover, due to security reasons, only two vehicles are allowed to be in the middle aisles at the same time, whereas three are allowed in the first and last aisle.

At the loading zone each vehicle is "loaded" with up to 19 orders. Afterwards a dispatcher decides when to send the vehicle onto the course. After leaving this area the vehicles automatically stop at a position where cards have to be picked from the shelf. Signal lights indicate the position from where and to which box the cards are picked.

The management was unhappy with the system since frequently vehicles ran into congestions and orders were completed late. For example, suppose that there is a vehicle that

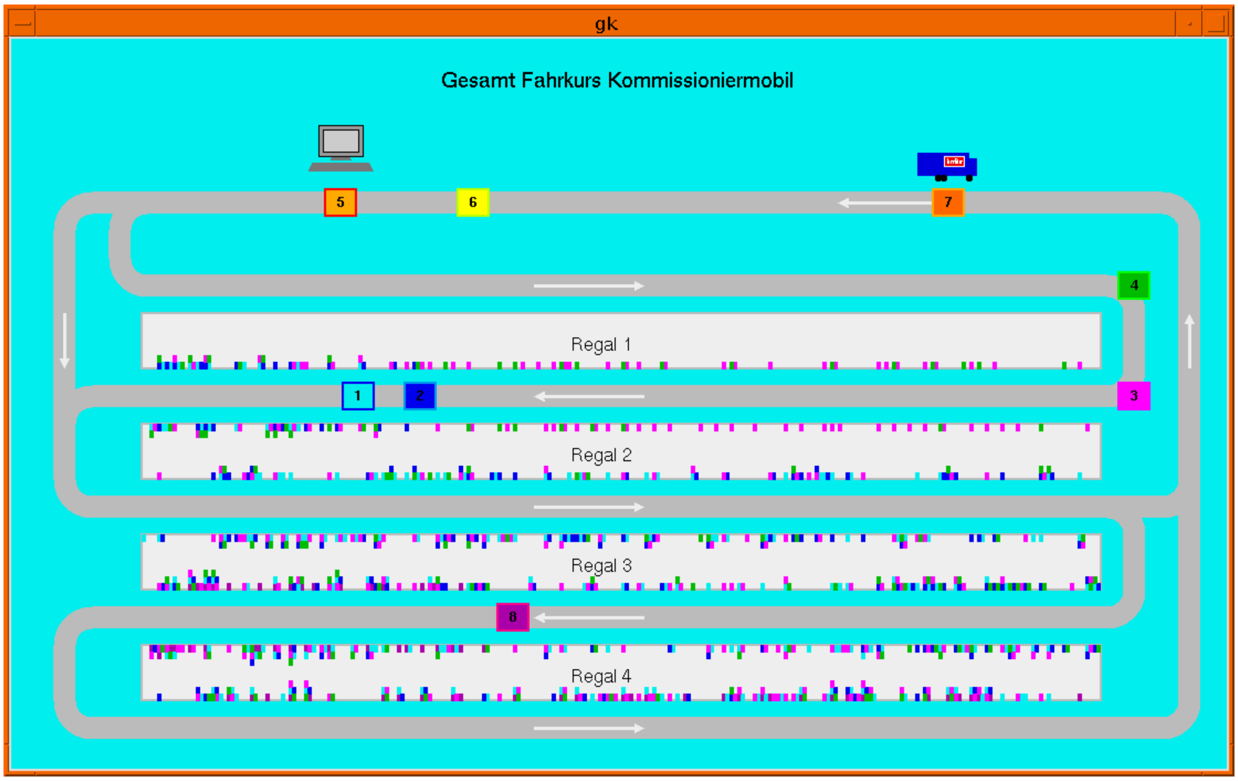

Figure 2 Commissioning area for greeting cards *(screenshot from the simulation program)*

requires a lot of stops and the subsequent one only has a few stops. In case the dispatcher sends them onto the course immediately the fast vehicle will catch up with the slow one and will frequently run into congestions. As a consequence the order pickers of fast vehicles often left the AGV to smoke a cigarette, etc., which resulted in further congestions.

Modelling. We suggested to assign the orders to the vehicles in such a way that whenever a vehicle stops the order picker can collect as many cards as possible. Or alternatively, for a given set of orders minimize the total number of stops to fulfill these orders. In this way it is possible to avoid some time–consuming deceleration, fine adjustment, and acceleration phases for the vehicles. Besides minimizing the total number of stops, we aim at reducing the time vehicles spend in congestion. This can be modelled as a mixed integer program. First computational test showed that for some data sets provided by Herlitz it took several hours of CPU–time just to solve the linear relaxations of the MIP. Thus, an exact solution approach was unsuitable for a deployment in the distribution center.

It could be shown that already the problem of minimizing the total number of stops is $\mathcal{NP}$–hard [22]. Therefore, we implemented several heuristics that reduce the total number of stops required for the vehicles and evenly distribute these stops among them. We used variants of greedy– and best–fit–algorithms with an additional 2–exchange improvement heuristic. In addition, we used a coarse simulation to determine the best starting time for each vehicle. By this optimization–simulation approach predictable congestions are shifted to the loading zone, where the order pickers can either have a break or can be assigned other tasks.

Results. We implemented a very detailed simulation model for the whole commissioning area in which we compared our approach to the one used so far. Herlitz provided production data from a period of about six weeks, which were the basis for the comparison. The main results are the following:

- A significant improvement with respect to the completion times of the orders can be achieved.
- The number of vehicles can be reduced from eight to six without any negative impact on the system performance.
- Congestions can more or less be avoided completely. Vehicles run into congestions only for a few seconds.

More details can be found in [9, 22]. A prototype of the simulation approach is currently tested by the support team of Herlitz for its use as a decision support tool for the dispatcher.

2.3 Challenges

Conveyor modules in large scale transportation systems. In connection with another cooperation with Herlitz AG, Berlin, we are analyzing the following problem: the automated pallet transportation system in a large dispatch building of Herlitz in Falkensee has to take care of a congestion–free flow of pallets from/to ware–input, commissioning departments, shelf system, and ware–output. Among the building blocks for pallet transportation the following seem to be the most complex ones:

- the automated shelf systems,
- the automated elevator systems.

Modules of these types are found in many automated transportation systems.

One would like to describe how different modules of such a system must be controlled in order to work together well. The common practice is to run very simple heuristics with emphasis on avoiding congestion.

Prior to the investigation of the interplay between the modules it is necessary to understand the modules themselves. While for the automated shelf systems we can use our experience from the above mentioned project with Siemens Nixdorf, the elevator control problem is not well understood so far. This is even the case in very elementary settings, let alone real–world layouts with additional restrictions to the flow from/to the elevators.

A generic simulation environment for elevator systems based on the event based simulation library AMSEL [5] was designed in order to test heuristic approaches to the problem (see Figure 3).

Conceptual problems. The competitive analysis is a mathematical performance measure of online algorithms. It has the advantage of not being dependent on the knowledge of the probability distribution of the requests. However, the online model it relies on is too restrictive for many real–world problems. For an elevator system, e.g., there are usually many requests available at the same time, and the elevator has the opportunity to make an offline schedule based on the known information. Moreover, not yet processed requests may be rescheduled

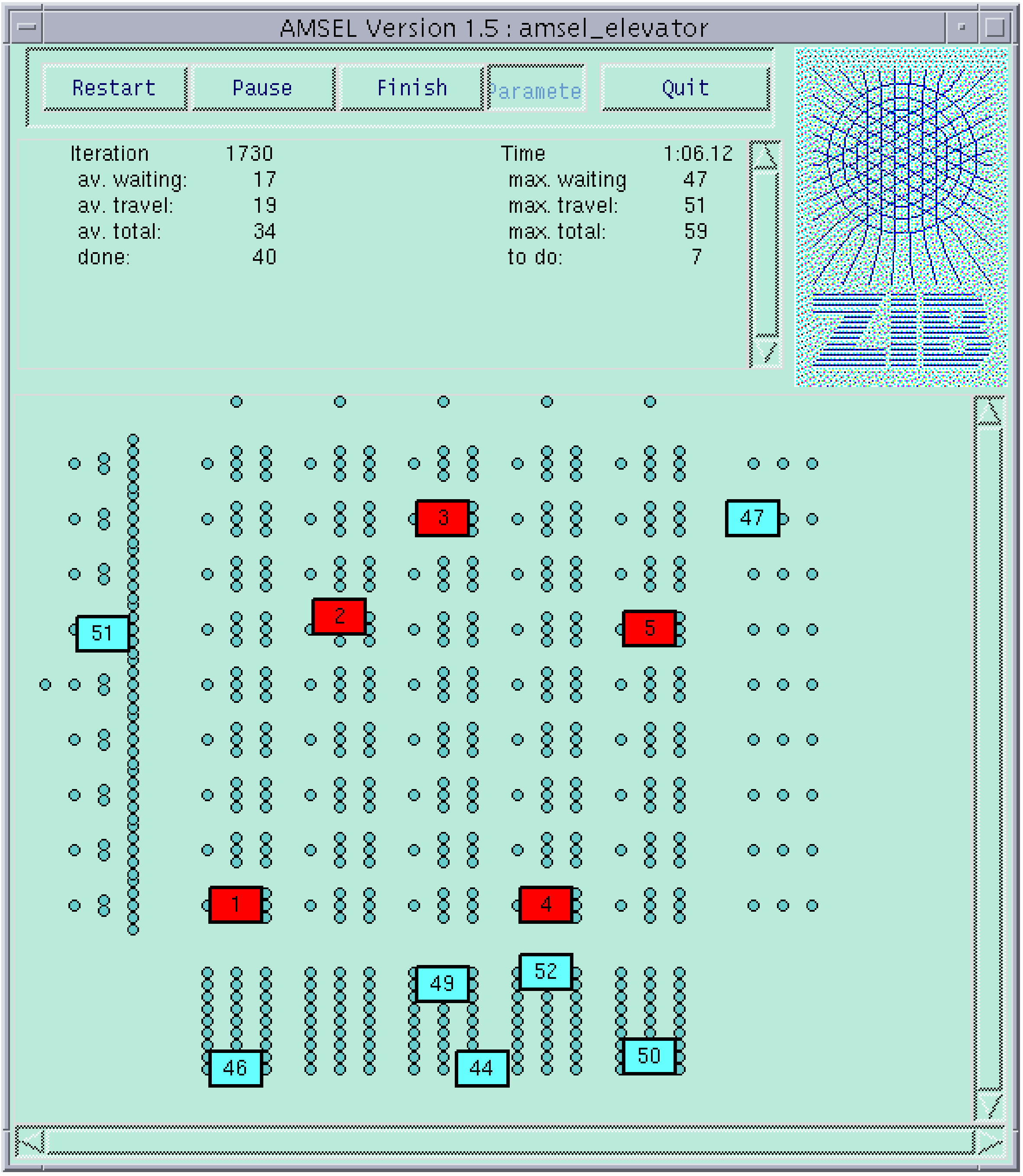

Figure 3: A snapshot of the animated simulation of an elevator system.

by the algorithm. This *dynamic look–ahead* should be integrated into a generalization of competitive analysis.

A very hard problem occurs if there is no corresponding offline problem at hand. In these cases even the definition of what should be an optimal solution to the online control problem is problematic. In control theory one computes an optimal control at each point in time. Usually one cannot ensure that these local optima combine to a globally "optimal" solution if the problem is discrete because the objectives are not continuously dependent on the decisions.

3 Conclusion and outlook

Online problems come up almost everywhere in industrial production, logistics etc. We have shown this by means of a few practically relevant examples. But online optimization problems have not yet received too much attention in the mathematical programming community yet, in particular in combinatorial optimization.

The field lacks "good" mathematical concepts for decision support. From a practical point of view, competitive analysis and the like too rarely yield results that can guide decisions about the selection of online algorithms to be used. Simulation experiments are still the state of the art.

Nevertheless, using the tools that have been developed in combinatorial optimization over the years, i.e., by combining and modifying various heuristic and exact approaches for associated offline problems, it is still possible to improve considerably on what is currently done in practice, as our examples show.

Acknowledgements

The research was supported by the *Deutsche Forschungsgemeinschaft (DFG)* within the research cluster *Echtzeit–Optimierung großer Systeme*.

References

[1] A. Abdel-Aziz Abdel-Hamid. *Combinatorial Optimization Problems Arising in the Design and Management of an Automatic Storage System.* PhD thesis, Technische Universität Berlin, 1994.

[2] A. Abdel-Aziz Abdel-Hamid, N. Ascheuer, and M. Grötschel. Order picking in an automatic warehouse: Solving online asymmetric TSPs. Technical Report SC 98–08, Konrad–Zuse–Zentrum Berlin, 1998, (in preparation).

[3] S. Albers. Competitive online algorithms. BRICS Lecture Series LS-96-2, 1996.

[4] S. Albers. Competitive online algorithms. *OPTIMA*, 54:2–8, June 1997.

[5] *AMSEL – A Modelling and Simulation Environment Library*, 1997. Developed at the Konrad-Zuse-Zentrum Berlin. [1].

[1] See `http://www.zib.de/ascheuer/AMSEL`

[6] N. Ascheuer. *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems.* PhD thesis[2], Technische Universität Berlin, 1995.

[7] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric travelling salesman problem with time windows. Preprint SC 97-11[3], Konrad-Zuse-Zentrum Berlin, 1997.

[8] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving ATSP with time windows by branch-and-cut. Technical Report, Konrad–Zuse–Zentrum Berlin, 1998. (In preparation).

[9] N. Ascheuer, M. Grötschel, and N. Kamin. A combined optimization–simulation approach for a system of automatically guided vehicles. Technical Report, Konrad–Zuse–Zentrum Berlin, 1998. (In preparation).

[10] N. Ascheuer, M. Jünger, and G. Reinelt. A branch & cut algorithm for the asymmetric Hamiltonian path problem with precedence constraints. Technical Report SC 97–70, Konrad–Zuse–Zentrum Berlin, 1997.

[11] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Serving requests with on–line routing. In *Proceedings of the 4th Scandinavian Workshop on Algorithm Theory*, 1994.

[12] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Competitive algorithms for the on-line traveling salesman. In *Workshop on Algorithms and Data Structures*, 1995.

[13] M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors. *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science.* Elsevier Sci. B.V., Amsterdam, 1995.

[14] M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh, and F. Soumis. Vehicle routing with time windows: Optimization and approximation. In B.L. Golden and A.A. Assad, editors, *Vehicle routing: Methods and studies*, pages 65–84. North–Holland, 1988.

[15] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. *Time Constrained Routing and Scheduling*, chapter 2, pages 35–139. Volume 8 of Ball et al. [13], 1995.

[16] M. Fischetti and P. Toth. An additive bounding procedure for the asymmetric TSP. *Mathematical Programming*, 53:173–197, 1992.

[17] M.R. Garey and D.S. Johnson. Two–processor scheduling with start–times and deadlines. *SIAM Journal on Computing*, 6:416–426, 1977.

[18] M.X. Goemans. On-line algorithms. Lecture Notes, September 1994.

[19] Herlitz PBS AG. Information available via WWW at URL `http://www.herlitz.de`.

[20] S. Irani and A.R. Karlin. Online computation. In D.S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, chapter 13, pages 521–564. PWS Publishing Company, 1997.

[2] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`

[3] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`

[21] P. Kall and S.W. Wallace. *Stochastic Programming.* John Wiley, 1994.

[22] N. Kamin. *On-Line Optimization of Order Picking in an Automated Warehouse.* PhD thesis, Technische Universität Berlin, 1998.

[23] T. Krippner and H. Matejka. On-line Optimierung eines Testregallagers. Modellierung und Vergleich verschiedener Heuristiken. Master's thesis, Universität Augsburg, Germany, 1993.

[24] R. Motwani and P. Raghavan. *Randomized Algorithms*, chapter 13. Cambridge University Press, 1995.

[25] T. Ottmann, S. Schuierer, and C. Hipke. Kompetitive Analyse für Online–Algorithmen — Eine kommentierte Bibliographie (Teil I und Teil II). Technical Report, Institut für Informatik, Albert–Ludwigs–Universität Freiburg, August 1994.

[26] A. Prékopa. *Stochastic Programming*, volume 324 of *Mathematics and Its Applications.* Kluwer Academic Publishers, Dordrecht/Boston/London, 1995.

[27] M.W.P. Savelsbergh. Local search for routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.