Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

NORBERT ASCHEUER MARTIN GRÖTSCHEL
SVEN O. KRUMKE JÖRG RAMBAU

Combinatorial Online Optimization

Preprint SC 98-24 (September 1998)

COMBINATORIAL ONLINE OPTIMIZATION

NORBERT ASCHEUER, MARTIN GRÖTSCHEL, SVEN O. KRUMKE, AND JÖRG RAMBAU

ABSTRACT. In "classical" optimization, all data of a problem instance are considered given. The standard theory and the usual algorithmic techniques apply to such cases only. Online optimization is different. Many decisions have to be made before all data are available. In addition, decisions once made cannot be changed. How should one act "best" in such an environment?

In this paper we survey online problems coming up in combinatorial optimization. We first outline theoretical concepts, such as competitiveness against various adversaries, to analyze online problems and algorithms. The focus, however, lies on real-world applications. We report, in particular, on theoretical investigations and our practical experience with problems arising in transportation and the automatic handling of material.

1. INTRODUCTION

It is a standard assumption in mathematics, computer science, and operations research that problem data are *given*. However, many aspects of life are *online*. Decisions have to be made without knowing future events relevant for the current choice. Online problems, such as vehicle routing and control, management of call centers, paging and caching in computer systems, foreign exchange and stock trading, have been around for a long time, but no theoretical framework existed for the analysis of online problems and algorithms. Correspondingly, only minor research efforts were spent on this topic.

The situation has changed considerably in the recent ten years. Both, the introduction of competitive analysis and the increasing economic importance of online problems have fostered this development. The first books on online computation, see [13, 15], and several survey articles, see [1, 2, 8, 22, 27] appeared recently and made the achievements and the weaknesses of the current theory known to a broader public.

This paper is not another comprehensive survey. It only briefly outlines some parts of the relevant theory. It concentrates on concrete applications of combinatorial online optimization, sketches the (rather poor) theoretical knowledge on these problems and describes what is done in practice for their solution. We report in particular on the control of elevators, the routing of stacker cranes, and the assignment of orders to commissioning vehicles.

2. Theory and Reality

We imagine to be in the following situation. We wish to optimize a certain objective function (in this paper we will always *minimize*) and have to make decisions before we know all data. We call such a scenario an *online problem*. We describe such a problem by a sequence of data (in online optimization one usually speaks of a sequence of *requests*) and assume that, after a request arrives, a decision has to be made before the next request appears. An algorithm acting this way is called *online algorithm*. It is, in general, also assumed that a decision cannot be revoked and that the number of requests coming up is unknown in the beginning. (Variants of these concepts will come up later.)

Let $C_{\mathsf{X}}(\sigma)$ denote the value of the objective function, called *cost* from now on, that algorithm X produces when acting on the request sequence σ. (For notational convenience we will assume throughout this paper that all values $C_{\mathsf{X}}(\sigma)$ are strictly positive.) If we have two algorithms A and ADV, both acting on σ, we can compare their results. We say that algorithm A is c*-competitive with respect to algorithm* ADV if there are real numbers $c \geq 1$ and b such that

$$C_{\mathsf{A}}(\sigma) \leq c \cdot C_{\mathsf{ADV}}(\sigma) + b$$

holds for all sequences of feasible requests σ. The typical situation here is that we would like to analyze the quality of an online algorithm A with respect to another algorithm, called *adversary* and therefore denoted by ADV. The adversary is not necessarily an online algorithm and may know the request sequence in advance. Even worse, ADV may produce the request sequence in such a way that ADV's solution is very good and A's very bad.

A worst possible adversary is an algorithm that computes an optimum solution. Making this precise needs some care. It is sometimes the case that a "natural" offline problem is associated with an online problem, i.e., that, ignoring the arrival times of the requests, the request sequence defines a standard optimization problem, the so-called *offline version* of the online problem. If OPT is an algorithm that solves this offline problem to optimality then an online algorithm A is said to be c*-competitive* if it is c-competitive with respect to its optimal offline adversary OPT.

However, it is not always clear what the offline version of an online problem is since the k-th request in a sequence may depend on what the online algorithm has done on a previous request. We will discuss this matter further in Section 6.

So far we have (implicitly) assumed that our algorithms act deterministically. In the context of online problems and algorithms it is natural to judge the quality of an algorithm by its expected performance and, moreover, to use randomized online algorithms. The concept of competitiveness can be defined for such settings analogously. Here it is also common to distinguish various adversaries with which the expected cost of a deterministic or randomized algorithm is compared.

An *oblivious* adversary is an adversary that acts independently of the decisions of the online algorithm while an *adaptive* adversary generates the next request depending on the answer of the online algorithm to the previous requests. The adaptive adversary may compute, in the end, an optimum offline solution—in this case it is called the *adaptive offline adversary*—but this adversary may also be required to act online and make a decision whenever a new request has been generated. This adversary is called *adaptive online*. The costs incurred by the oblivious and the adaptive offline adversary are of course the (expected) offline costs while, for the adaptive online adversary, there is no precise characterization of the cost generated. One simply takes what the algorithm has produced. One can prove that the competitiveness of an online algorithm with respect to an oblivious adversary is not larger than that against an adaptive online adversary and that this is not larger than that against an adaptive offline adversary.

We remark here that we have not made any requirement on the running time of either an online algorithm or an offline adversary. In fact, in many cases the offline versions of online problems are $\mathcal{NP}$-hard, and thus, we may compare a fast online heuristic with an algorithm that has exponential or worse running time. Competitive analysis ignores this aspect.

In practice, however, online algorithms are usually required to produce solutions "quickly". What "quickly" means depends on the application considered. The usual requirement is that an answer must be provided within a fixed time bound (which may be in the range of microseconds or even minutes if the process is somewhat slow). Whatever the bound is, the task is to find a "best possible" solution within the given time frame (and the available computing machinery). We call algorithms that are online and obey such additional requirements *real-time*. No (sufficiently interesting) concept for the performance evaluation of a real-time algorithm is known.

We will demonstrate that competitive analysis provides an overly pessimistic picture, i.e., an algorithm may not be competitive at all or may have a poor competitiveness factor c while its actual performance appears much better. Such an observation cries for *average case analysis*. However, this is a rather unexplored area of research. *Stochastic analysis* of an online problem is the next natural idea. The typical approach here is to view the given online problem as a stochastic program where the requests occur according to some distribution. An outcome of stochastic analysis would be an estimate of the average behavior of an algorithm that is based on solutions of the stochastic program or, even better, an estimate of the expected online cost of a given problem. A critical discussion of some aspects of this approach can, e.g., be found in Appendix B of [13].

Queuing theory addresses online problems as well. However, the focus of this approach is different. One considers, e.g., a given system (including the algorithms that control the system) and assumes that a sequence of requests arrives according to some distribution. Queuing theory tries to estimate important parameters of the system such as average service or waiting times. The goal is the description of the state and not the optimization of the process, see [3] among others.

If the theory of online algorithm looks so bleak and gives no guidance for practice what should be done in case of a real application? The answer of practitioners is: *simulation*. In fact, we have no better answer either and are actually using simulation to evaluate and compare online algorithms for combinatorial optimization problems that occur in practice. A discrete event oriented simulation program, called AMSEL [4], has been developed by N. Ascheuer that is particularly suited for a quick simulation analysis of logistics systems. All simulations discussed later have been executed with AMSEL.

There are (at least) two ways how simulation is used. For a system (such as the transportation system within a factory) to be designed, one sets up a simulation model that reflects the basic properties of the system to be built. Making various assumptions about the demands of this system one designs a number of "reasonable" or even "extreme" scenarios to see how the system behaves, whether waiting times, throughput times, etc. are acceptable, and if not, one modifies the system so that the expected standards can be achieved. Once such systems are built they often show different than the expected behavior. Then, instead of rebuilding or technically changing the system, one tries to optimize some or all of the components of the systems to achieve better results. Here online algorithms come into play. Typically several online algorithms are coded and, using real data of the past, their practical performance is analyzed by simulating the system where the new online algorithms replace the old procedures. Such comparisons, based on real system data, are used to judge whether the use of online algorithms pays and, if so, which algorithms to choose for the use in practice.

We regret that we still use this old fashion trial-and-error approach. It seems, however, that we have to, at least at present, since we know of no mathematical tools that could give us better guidance for real-world decisions.

3. Modeling Issues

We would like to outline, by means of a few examples, modeling issues that come up when online problems are analyzed mathematically.

Let us begin with one of the most studied problems in the "online world": *bin packing*. We assume here that we have a (potentially unlimited) number of bins of size a. Moreover, there is a sequence of items of size $a_i, 0 < a_i \leq a$, that have to be packed into the bins. The goal is to use as few bins as possible. (This problem in $\mathcal{NP}$-hard.)

In the online problem we have to pack an item into a bin as soon as it arrives (and before the size of the next item is known). See, among others, [18] for a survey on online bin packing. In the standard offline problem one assumes that the whole sequence $\sigma = (a_1, \ldots, a_n)$ of item sizes is given and that these have to be packed optimally. Let $C_{\mathsf{OPT}}(\sigma)$ denote the minimum number of bins needed. We would like to know how online bin packing algorithms behave with respect to this offline adversary. We first address lower bounds, i.e., we ask whether there is a constant c_l such that no online algorithm can guarantee a competitiveness of c_l or below.

Here is a simple observation. Let A be any online algorithm for the bin packing problem. Let us try to find a c_l such that

$$C_{\mathsf{A}}(\sigma) \leq c_l \cdot C_{\mathsf{OPT}}(\sigma)$$

cannot be guaranteed. We choose some positive number h. Let $a = 2h$ be the bin size, and consider the sequence $h - \epsilon, h - \epsilon, h + \epsilon, h + \epsilon$, (where $\epsilon > 0$ is an arbitrarily small constant) of four item sizes. A must put the first item into bin 1. If A puts item 2 into bin 2 then the adversary stops. A has used 2 bins, the optimum is 1 bin and hence $c_l \geq 2$. To achieve a better competitiveness A must put item 2 into bin 1. But then items 3 and 4 must go into bins 2 and 3. Hence, $C_{\mathsf{A}}(\sigma) = 3$ and $C_{\mathsf{OPT}}(\sigma) = 2$, which shows $c_l \geq 1.5$.

Taking now also the additive constant into account and using input sequences consisting of n items of size $h - \epsilon$ and n items of size $h + \epsilon$ one can similarly show that

$$C_{\mathsf{A}}(\sigma) \leq c_l \cdot C_{\mathsf{OPT}}(\sigma) + b$$

cannot be guaranteed for $c_l = \frac{4}{3}$ by any online algorithm whatever choice of the constant b is used. Using very clever refinements of this simple input sequence (and further ideas) this lower bound has been improved by various authors. The current best lower bound is $c_l = 1.540$, see [26].

Let us now look at the most trivial online algorithm for bin packing one can think of: NEXT FIT (NF). NF has an active bin (initially this is the first bin). NF tries to put a new item into the active bin. If it does not fit, NF closes this bin and puts the item into a new (now active) bin. Let $b_i, i = 1, \ldots, k$, denote the sum of the item sizes packed into bin i at termination. Then we have $b_i + b_{i+1} > a$ which immediately implies that $C_{\mathsf{NF}}(\sigma) \leq 2 \cdot C_{\mathsf{OPT}}(\sigma)$ for all sequences σ.

A little more elaborate are FIRST FIT (FF) and BEST FIT (BF). FF and BF keep all bins active; FF puts the next item into the first bin into which it fits, while BF puts the item into one of the bins into which it fits and that makes the bin as full as possible. A similarly trivial analysis shows that FF and BF are at least 2-competitive. With a lot of technical detail $c_l = 1.7$ can be established. In fact, a competitive ratio of at least 1.7 can be shown for any online algorithm that never opens a new bin unless the current item does not fit into any of the open bins. These algorithms can be refined. The current online champion (a variant of the modified HARMONIC algorithm) achieves $c_l = 1.5887$, see [25]. Hence, upper and lower bounds for the competitiveness of online algorithms for bin packing are quite close, but not identical.

In general, it is often not clear what a suitable offline problem is. In the case of bin packing one may have doubts whether always an unbounded number of bins can be kept available for the online algorithm to choose the best fitting bin, for instance. Therefore, the *bounded-space online bin packing problem* has received some attention. Here at most k (k given in the beginning) bins are kept active and—in contrast to the ordinary bin packing problem—one is allowed to repack the active bins. Lee and Lee [24] proved that, for any fixed k, no online algorithm can be better than 1.69103-competitive. Galambos and Woeginger [17] designed a (polynomial time) online algorithm that achieves this bound, in particular, for $k = 3$. Thus, for the bounded-space online bin packing problem, there is an algorithm that is optimal with respect to the achievable competitiveness. Moreover, the algorithm needs only three active bins for repacking. Keeping more bins open for repacking does not pay—at least within this theoretical framework.

Results of this type are rare. Moreover, in practice the mathematical models to consider are not so clear cut as suggested above. The typical issue is that there is a continuing process where, from time to time, decisions have to be made that influence various parameters of the process. The person controlling or in charge of the process is not "happy" with the performance of the system. The question is what can and what should be done, and if the decision process (the online algorithm controlling the system) has been changed, in what respects is the new solution better or worse than the old one?

We call this the "modeling issue" and will explain in the subsequent sections what we mean by this. The point is that, in general, there is not a single objective that one has to look at; practical online environments are complex with many alternatives for success or performance evaluation.

A second important issue, often overlooked, is the choice of the adversary with which the performance of an online algorithm should be compared. The point here is that the choice of an adversary that is too "strong" may produce a grossly pessimistic judgement of the quality of an online algorithm. We will discuss this issue in detail with respect to online stacker crane optimization where the asymmetric traveling salesman problem (ATSP), the ATSP with precedence constraints, and the ATSP with time windows are possible adversaries and provide rather different pictures of the performance of our online algorithm, see Section 6.

A third issue, already indicated above, is the fact that an online problem may not necessarily have a natural offline version. This fact displays again in our stacker crane application, see Section 6. Here a stacker crane moves objects within an automatic storage system. Moves requested may depend on the moves the stacker crane has executed previously. Thus, in this case, there is no offline sequence of requests that could be given beforehand.

4. Online Control of Elevator Systems

4.1. **Problem Description.** The automated pallet transportation system in the distribution center of Herlitz PBS AG [21] (one of Europe's largest producer and distributor of office supply) has to take care of the flow of pallets from/to the receiving docks, the production and commissioning departments, the automated storage system, and the delivery docks. This pallet transportation network connects nine factory floors. Among the building blocks for pallet transportation are two systems with five elevators. Each elevator can carry one pallet.

In principle, the control system of the transportation network has the task to move items as fast as possible without congestion, to observe time windows, due dates and other technical details. All this has to happen in real-time. The answer times have to be below one second. There is no clear objective that one could formulate mathematically. Nevertheless, it is important that each individual component of the transportation system works efficiently. We are currently investigating the elevator system. How should the elevators move in such an environment?

A typical approach would be to schedule the transportation requests in such a way that either the average or the maximum waiting time (or a combination of the two) is as short as possible. It is easy to show that, for these two objectives, there does not exist a c-competitive online elevator scheduling algorithm for any constant c. Hence, we decided to look at *minimum completion time* (i.e., the total time needed to serve all requests) for competitive analysis.

4.2. **Mathematical Model and Theoretical Results.** We will start with a more general problem than needed for our elevator application. An instance of the *online transportation problem* OLTP consists of an undirected graph $G = (V, E)$ with edge-weights $d(e)$ $(e \in E)$ and a distinguished origin vertex $o \in V$. We are also given an integer $C \geq 1$ which specifies the *capacity* of the server (elevator, vehicle, etc.). In all what follows here the capacity will be $C = 1$. Finally, one is given a sequence $\sigma = (r_1, \dots, r_m)$ of transportation requests.

In the OLTP each request is a triple $r_i = (t_i, a_i, b_i)$, where t_i is a real number, the time where request r_i becomes known, and $a_i \in V$ and $b_i \in V$ are the source and target of the transportation task. It is assumed that the requests in the sequence σ are ordered according to their release times, i.e., that $t_1 \leq \cdots \leq t_m$. The server can move at constant unit speed and is located at the start o at time 0.

Given a sequence σ of requests, a transportation schedule is said to be *valid* for σ, if the following conditions are satisfied: (a) The server starts its movement in the origin vertex o and, (b) each transportation request in σ is served, starting not earlier than the time it becomes known.

In the sequel let $C_{\mathsf{A}}(\sigma)$ denote the completion time of the server controlled by algorithm A on the input sequence σ. We also use OPT to denote the optimal offline algorithm.

Observation. *No deterministic online algorithm for OLTP has a competitive ratio better than 2 (with respect to completion time).*

Proof: The underlying graph $G = (V, E)$ for the instance of OLTP consists of a path of $2n + 1$ vertices with the origin being the middle vertex at distance of n from each end. More formally, we have that $V = \{ v_i \mid i = -n, \dots, n \}$ with $o := v_0$ and $E = \{ v_i v_{i+1} \mid i = -n, \dots, n-1 \}$. All edge-weights are equal to 1.

Assume that A is an algorithm with competitive ratio $c < 2$. We choose the number n so large that $2 \geq 2 - \frac{3}{n} > c$.

The first request given is $r_0 = (t_0 = 0, v_0, v_{-1})$. We now claim that at time $t_1 = n - 1$ the online-server cannot be strictly to the right of vertex v_1, i.e., on the path from v_1 to v_n but not on v_1. In fact, if the server were to the right, say at distance $\delta > 0$ to the right of vertex v_1, then we could add the request $r_1 = (n - 1, v_{-(n-1)}, v_{-n})$. The online server would then need time at least $1 + \delta + n$ to serve the request. This would result in a total time of $2n + \delta$. On the other hand, the offline server could serve r_0 starting at time $t_0 = 0$ and then continue to move to the left until it reaches vertex $v_{-(n-1)}$ at time $n - 1$ ready to serve the new request r_1. Thus, the offline server needs time n and thus $C_{\mathsf{A}}((r_0, r_1))/C_{\mathsf{OPT}}((r_0, r_1)) = (2n + \delta)/n > 2$, which means that A would not be 2-competitive.

We have seen that at time $t_1 = n-1$ the online server is to the left of vertex v_1. We now add the request $r'_2 = (n - 1, v_{n-2}, v_{n-1})$. The total time needed by the online server is then at least $n - 1 + (n - 2) = 2n - 3$.

On the other hand, the offline server serves the sequence (r_1, r'_2) by handling r_1 at time t_0 and then immediately moving to vertex v_{n-2} which it reaches at time $n - 1$ ready to serve r'_2. Thus, we have that $C_{\mathsf{OPT}}((r_1, r'_2)) = n$ and $C_{\mathsf{A}}((r_1, r'_2))/C_{\mathsf{OPT}}((r_1, r'_2)) = 2 - 3/n > c$. This contradicts the assumption that A is c-competitive with $c < 2$. □

A natural approach for the design of online algorithms is the following REPLAN-strategy. We formulate it here for the OLTP. However, the REPLAN-strategy is more of a general principle than of a specialized algorithm for a particular problem.

Strategy REPLAN: Whenever a new request arrives, the server completes the current transportation move (if it is performing one), then the server stops and does a *replan*: it computes a new shortest transportation sequence starting at the current position of the server and containing all yet unserved requests.

Observation. REPLAN *is a 3-competitive online-algorithm for OLTP.*

Proof: Let $\sigma = (r_1, \dots, r_m)$ be any sequence of requests. We consider the time t_m, when the last request $r_m = (t_m, a_m, b_m)$ becomes known. Let $s(t_m)$ denote the position of the REPLAN-server at time t_m.

Case 1: The server is not performing a carrying move at time t_m.

The REPLAN server computes an optimum transportation sequence S which starts at its current position $s(t_m)$ and serves all unserved requests. The time needed for S is at most $d(s(t_m), o) + C_{\mathsf{OPT}}(\sigma)$, since all the requests from σ are available for planning at time t_m and the server could move to the origin o in time $d(s(t_m), o)$ and from this point on follow the tour of the optimum offline server OPT. Thus, REPLAN will finish its work no later than $t_m + d(s(t_m), o) + C_{\mathsf{OPT}}(\sigma)$. Since the REPLAN-server has traveled to point $s(t_m)$ there must be a request $r = (t, a, b)$ in σ with $\max\{d(o, a), d(o, b)\} \geq d(o, s(t_m))$ from which we conclude that $d(s(t_m), o)) \leq C_{\mathsf{OPT}}(\sigma)$. Finally, we can use that the offline server can never complete its transportation before time t_m, i.e., when the last request becomes known. This gives us that $C_{\mathsf{REPLAN}}(\sigma) \leq 3 \cdot C_{\mathsf{OPT}}(\sigma)$.

Case 2: REPLAN is performing a carrying move from a to b at time t_m.

The server will finish its move at time $t_m + d(s(t_m), b)$. It is easy to see that the shortest schedule starting at b serving all yet unserved requests has length at most $C_{\mathsf{OPT}}(\sigma) - d(a, b) + d(b', o)$, where b' is the endpoint of the route taken by the optimum offline OPT. Thus,

$$\begin{aligned}
C_{\mathsf{REPLAN}}(\sigma) &\leq t_m + d(s(t_m), b) + C_{\mathsf{OPT}}(\sigma) - d(a, b) + d(b', o) \\
&\leq t_m + d(s(t_m), b) + C_{\mathsf{OPT}}(\sigma) - d(a, b) + d(b', o) + d(a, s(t_m)) \\
&\leq t_m + C_{\mathsf{OPT}}(\sigma) + d(b', o) \\
&\leq 3 \cdot C_{\mathsf{OPT}}(\sigma).
\end{aligned}$$

This completes the proof of the 3-competitiveness of REPLAN. □

No competitiveness results are known for elevators of capacity larger than one or systems of more than one elevator. Further investigations can be found in [10, 14, 19, 20].

4.3. **Simulation Results and Practical Impact.** For the OLTP we have set-up a parameterized simulation system based on AMSEL [4] (see Figure 4.3). Several algorithms were tested on random data, among them the simple FIRST FIT heuristic. This strategy always serves the closest available request first and hence does not organize at all the transportations in advance. This strategy is one of those used in the distribution center of Herlitz.

The performance was measured not only with respect to the completion time but also to the average and maximal flow time. This is the average resp.

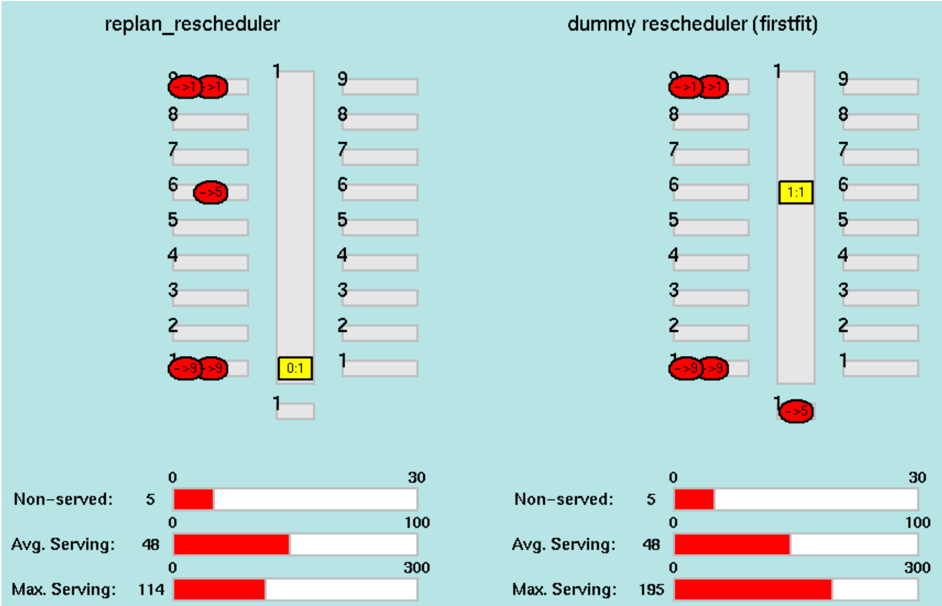

FIGURE 1. Screenshot of the simulation system for elevator systems

the maximum time between the time a request becomes known and its completion time.

Although REPLAN is competitive with respect to completion time and FIRST FIT is not known to be, both algorithms have almost identical completion times. REPLAN is considerably better than FIRST FIT with respect to maximum flow time (about 20% on the average) while FIRST FIT beats REPLAN slightly concerning average flow time (about 5%), see Table 4.3 for some simulation runs corroborating these observations.

Completion Time						
Algorithm	data set 1	data set 2	data set 3	data set 4	data set 5	average
FIRST FIT	16:03.27	16:02.41	16:00.21	16:01.24	16:00.40	**16:01.43**
REPLAN	16:03.11	16:02.47	16:01.05	16:01.17	16:00.40	**16:01.48**
Average Flow Time						
Algorithm	data set 1	data set 2	data set 3	data set 4	data set 5	average
FIRST FIT	75	67	90	69	83	**76.80**
REPLAN	73	70	98	71	86	**79.60**
Maximal Flow Time						
Algorithm	data set 1	data set 2	data set 3	data set 4	data set 5	average
FIRST FIT	855	459	676	601	683	**654.80**
REPLAN	439	509	727	466	597	**547.60**

TABLE 1. Simulation results for OLTP on random data

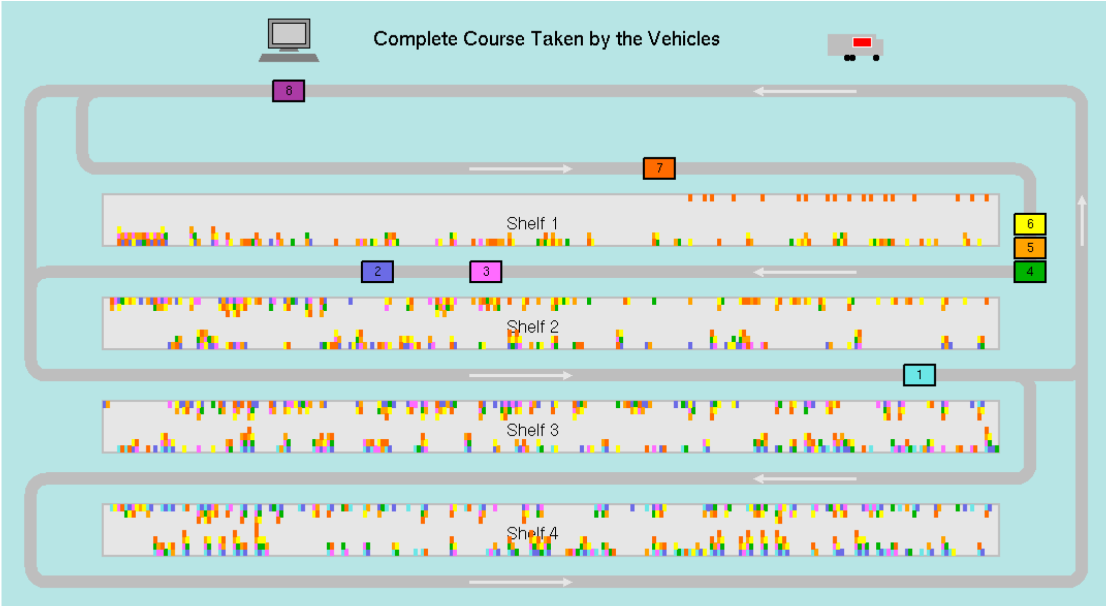

FIGURE 2. Screenshot of simulation system for commissioning mobiles

The next step in our project is to record real-world data from the Herlitz system in order to validate the simulation system and to find out whether any particular structure in the problem data leads to special results differing from those obtained by using random data.

5. ROUTING OF COMMISSIONING VEHICLES

5.1. **Problem Description.** In one of the departments of the Herlitz distribution center greeting cards are commissioned. The cards (about 4000 different types) are stored in four parallel shelving systems, see Figure 5.1. In accordance with the customers' orders, the different greeting cards are collected in boxes that are eventually shipped to the recipient. Order pickers on eight automated guided vehicles collect the orders from the storage systems, 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, see Figure 5.1. (Three vehicles line up in front of aisle 2 since they are not allowed to enter.)

At the loading zone (where vehicle 8 is located in Figure 5.1), each vehicle is assigned up to 19 orders from an order pool that changes whenever a software system (operating on a higher level) releases a new set of orders. A dispatcher decides when to send the vehicle onto the course. After leaving the loading zone the vehicles automatically stop at the positions where cards have to be picked from the shelf. Each stop position holds up to ten different types of cards.

There is no straightforward single objective for the optimization of the greeting card commissioning system. Each greeting card order is, in general, part of a larger order including, for example, orders for pencils, writing pads, staplers, and other stationary. All the parts of an order have to reach a specified destination in the delivery zone within a certain time window. The overall order processing is organized in a just-in-time fashion. For this reason the greeting cards commissioning has to observe certain due dates. On the average, more than two thirds of the orders processed during a day have been generated during this day.

The Herlitz system analysts observed severe deficiencies in their current scheduling system. Considerable congestion occurred resulting in a violation of due dates. Congestion was viewed as a consequence of an "unbalanced" vehicle load which caused a "fast" vehicle (carrying orders with only a few stop positions) to wait behind a "slow" one. Congestion was further increased by "human factors". Order pickers waiting in a queue left their vehicles to smoke a cigarette etc. and forgot to return in time. It was decided to come up with a new scheduling system that minimizes completion time, obeys the due dates and avoids congestion. (For details see [23].)

5.2. **Mathematical Model and Theoretical Results.** For the theoretical analysis it is necessary to provide a mathematical formulation of the problem under consideration. We have started analyzing the Herlitz system by considering the following Commissioning Vehicle Problem (CVP).

An instance of CVP consists of a set $L = \{1, 2, \ldots, N\}$ of possible stop positions, and a set of empty vehicles $v_1, \ldots, v_q$, each with capacity C. A request sequence $\sigma = (r_1, r_2, \ldots)$ consists of a chronologically ordered collection of sets of stop positions.

For a sequence of requests, a solution to the CVP is an assignment for every request r_i to a vehicle $v(r_i)$ so that the number of requests assigned to each vehicle does not exceed C. The objective is to minimize the total number of stop positions assigned to the vehicles. In Kamin [23] it was shown that the offline version of CVP is an $\mathcal{NP}$-hard problem. Explicit solving of the integer programming formulation in reasonable time turned out to be out of reach for commercial software packages.

In the online-situation we require that request r_i is permanently assigned to vehicle $v(r_i)$ before r_{i+1} becomes known and that the length of the sequence is unknown until the last request comes in.

Observation. *For every request sequence $\sigma = (r_1, \ldots, r_m)$, any assignment of requests to vehicles needs at most C times as many stops as an optimum offline assignment.*

Proof. Each request r_i requires $|r_i|$ stops. If I_v is the set of (at most C) requests assigned to a vehicle v then, clearly, v stops at at most $\sum_{r_i \in I_v} |r_i|$ positions. Hence, for every request sequence and any assignment, the total number of stops is bounded from above by $\sum_{i=1}^{m} |r_i|$. On the other hand, a vehicle v has to stop at at least $\max_{r_i \in I_v} |r_i|$ positions. Since C is the capacity of the vehicle, this value is at least as large as $\sum_{r_i \in I_v} \frac{|r_i|}{C}$. This holds, in particular, also for the optimum offline assignment. Summing up over all vehicles yields the observation. □

This result is due to Kamin [23], who also constructed a request sequence on which every online algorithm needs at least C times as many stop positions as the offline optimum. These two observations yield:

Corollary. *Every online algorithm for CVP is exactly* C*-competitive.* □

We view this as a truly strange result which displays the weakness of competitive analysis in certain situations. Competitive analysis is unable, in the case of CVP, to distinguish between different online algorithms. On the other hand, it is quite obvious that a BEST FIT algorithm will perform much better in practice than a "WORST FIT" algorithm or a random assignment.

5.3. **Actual Approach, Simulation Results, and Practical Impact.** Several heuristics that reduce the total number of stops and distribute them evenly among the vehicles were implemented. These are versions of the BEST FIT and the GREEDY algorithm, together with local exchange heuristics. The computation times of these algorithms are short enough to run in a real-time situation. The real-time requirements in this special problem, however, are by far not as strict as in the OLTP.

We implemented a 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:

- The number of stops of the commissioning vehicles could be reduced by about 6% on the average compared to the solution that has been used at Herlitz.
- A tour of a commissioning vehicle at Herlitz took about 70 minutes including about 5 minutes time spent in congestion. This idle time could be reduced to 40 seconds on the average.
- Due to a detailed congestion analysis and prediction, changes in the dispatching habits (in addition to minimizing the number of stops) the completion times could be reduced significantly. On the average, the work load of a day (two shifts of 8 hours) could be completed about 5 hours earlier.

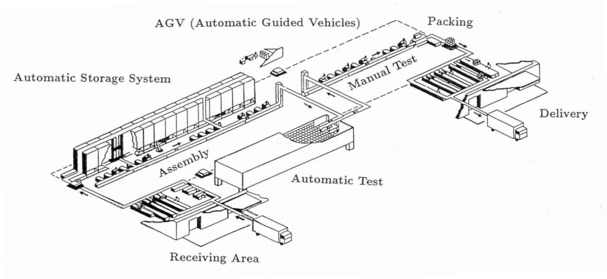

FIGURE 3. Sketch of the factory layout

- The number of vehicles (and therefore order pickers) could be reduced from eight to six without any negative impact on the system performance.

We conclude that our mix of heuristics—although not distinguished in the competitive analysis—whose choice was based on simulation experiments, was the basis for significant improvements in practice. The simulation results convinced Herlitz to test a prototype of the simulation program as a decision support tool for the dispatcher.

6. STACKER CRANE ROUTING IN AN AUTOMATIC STORAGE SYSTEM

6.1. **Problem Description.** Siemens Nixdorf Informationssysteme AG (SNI) maintains a production plant where all SNI personal computers and related products are assembled. All parts are stored in one of six automatic storage systems (AUSSs). The AUSSs serve as material buffers between the receiving area and the assembly lines located at each side of the AUSS. For each AUSS there is one stacker crane fulfilling transportation tasks between the receiving buffer, the storage locations, and the buffers for the assembly line. Figure 6 shows a sketch of the factory layout. (For a more detailed description of the layout, see [5].)

It is, of course, desired that the stacker crane works "efficiently". No problems arise at times of low work load where the stacker crane typically has to choose which of at most three jobs to perform first. However, there are times of heavy work load, e.g., after a system breakdown, when up to 60 transportation tasks have piled up. To get the production line working, the requested items have to be moved fast and it has to be made sure that certain jobs have to be executed within given time windows. After a detailed analysis it was decided to minimize the unloaded travel time of the stacker crane while observing certain priority rules. The stacker crane is supposed to work whenever a transportation task is available.

6.2. **Mathematical Model and Theoretical Results.** An offline version of the stacker crane problem that seems "natural" at a first glance is the asymmetric

	1	2	3	4	5
$\text{ATSP}^{\text{online}}$	11.50	14.66	11.33	10.00	11.92
ATSP	7.36	6.64	6.38	5.89	8.14
ATSP^{PR}	8.18	7.40	8.10	7.46	9.59
ATSP^{TW}	8.45	[8.6,11.3]	[9.2,11.3]	[9.7,10.0]	[10.4,11.4]
GAP :					
ATSP	56.2%	120.7%	77.5%	69.7%	46.4%
ATSP^{PR}	40.6%	98.1%	39.5%	34.0%	24.3%
ATSP^{TW}	36.1%	71.1%	22.6%	3.1%	14.9%

TABLE 2. Average unloaded travel time in seconds and gap for various adversaries

traveling salesman problem (ATSP). An instance of ATSP consists of a complete directed graph $D = (V, A)$. Each node in V corresponds to a transportation task, and the arc from v to w has weight corresponding to the travel time from the end point of task v to the starting point of task w. No polynomial time heuristic for the ATSP is known that has a constant quality guarantee. Similarly, no competitive online algorithm is known. (This contrasts with the situation for the symmetric traveling salesman problem, where, e.g., REPLAN is known to be 2.5-competitive [11, 12].) To overcome this gap we decided to invoke a-posteriori analysis which can be viewed as an instance-based competitive analysis.

We implemented several online algorithms (the AMSEL screenshot in Figure 6.3 shows the execution of three algorithms) and compared them to the optimum ATSP solution. The gaps were huge. Analyzing the ATSP solution we observed that some of the jobs created in the evening were executed in the morning and that some transportation tasks were performed before the items to be moved were actually available (as a result of a preceding move). This led us to consider more restricted versions of the ATSP. We introduced time windows and precedence constraints. (In fact, not much research had been spent on these ATSP versions, and significant work had to be invested to design and implement algorithms that can solve these problems to optimality for the range of instance sizes that come up in this application, see [5, 6, 7, 9]). Table 6.2 shows the gaps between our online solution and the three adversaries ATSP, ATSP with precedence constraints, and ATSP with time windows for some of the instances that came up at SNI. Clearly, a change of the offline adversary significantly changes our view of the performance of the online algorithm. Consider for instance problem 4 of Table 6.2. The online solution needs about 70% more time than the ATSP solution, 34% more time than the ATSP solution when precedence constraints are required and only 3.1% more time if time windows are observed where the latter deviation is only an upper

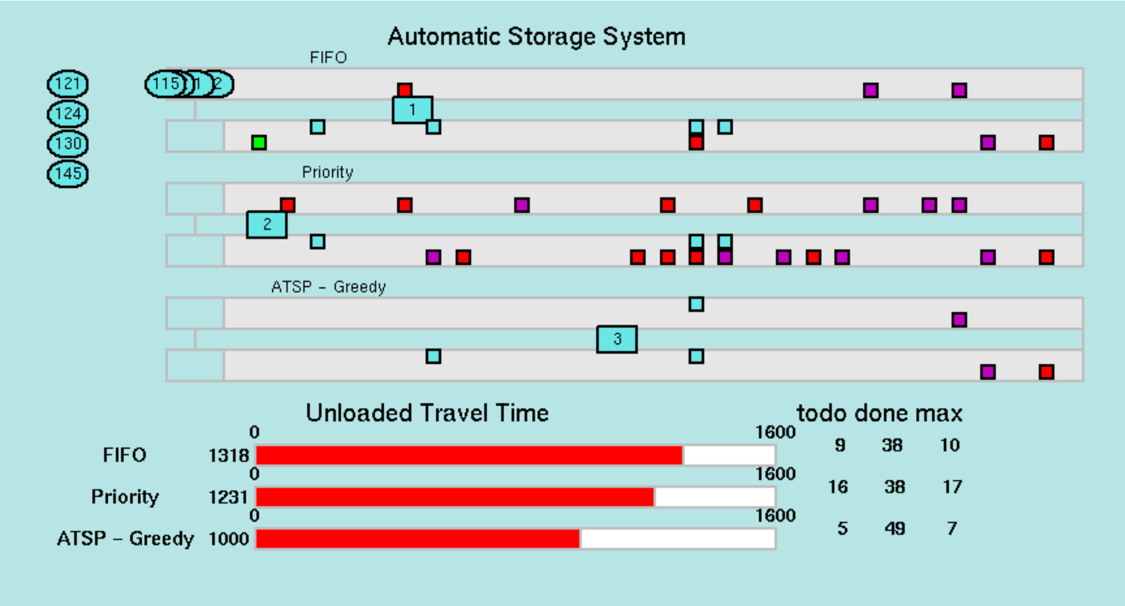

FIGURE 4. Screenshot of simulation system for automatic storage systems

bound on the gap since this instance could not be solved to optimality (9.7 is a lower and 10.0 an upper bound on the optimum ATSP^{TW} value).

The stacker crane problem displays a difficulty that frequently occurs in complex transportation problems. There is no "canonical" offline problem for the given online problem. For instance, the SNI logistics system is governed by certain rules that influence the request sequence for stacker crane moves, and in turn, the behavior of the stacker crane influences the other modules of the transportation system. For example, the input buffer of the AUSS is served by automatically guided vehicles. At most two items can be in the buffer. A vehicle may stand in front of the buffer carrying a further item for the AUSS. The transportation request for this item, however, is only generated when the item is moved into the buffer. On the other hand, if the waiting vehicle blocks other vehicles then a rule says that the vehicle must go away after 60 seconds. In general, it will make a turn and return to the buffer later on. This process shuffles the request sequence and leads to different generation times, time windows, etc. Therefore, none of our three offline adversaries is guaranteed to produce a sequence of stacker crane moves that can actually be executed in practice.

This short discussion indicates that finding an appropriate offline model of an online problem is an unresolved issue.

6.3. **Actual Approach, Simulation Results, and Practical Impact.** Here we describe what we did in practice. Although the problem is known to be $\mathcal{NP}$-hard there are codes available so that the REPLAN heuristics can be used in real-time situations. To be on the safe side, we implemented a three phases optimization process:

Phase 1: Perform cheapest insertion of the new task.
Phase 2: Run an iterative cheapest insertion on a random ordering of the current request sequence. Then pick the winner of Phase 1 and 2.
Phase 3: Solve the ATSP with the current transportation tasks to optimality (branch&bound from [16]) and replace the old sequence completely by the optimal one (REPLAN).

Phase 1 runs in time linear in the number of requests and is always completed. For the typical problem sizes that occur in our application (no. of requests is less than 60) this is done in fractions of a second. Even Phase 3 was always completed within a few seconds. If the stacker crane becomes idle, the optimization is interrupted, and the best sequence so far is passed to the stacker crane.

Although one can construct request sequences where REPLAN does not give the best result among all online algorithms we implemented, our simulation experience showed that REPLAN empirically provides the best results on average.

SNI provided data for one week of production. During this period, for one AUSS, each generated task and each move of the stacker crane was recorded. It turned out that in heavy load periods the times needed for unloaded moves could be reduced by approx. 30%.

As a result the optimization package was put in use, and the results were confirmed in everyday production. Even in the production environment the optimization process could almost always finish with Phase 3.

7. Conclusion

We hope that we could demonstrate that combinatorial online optimization is an exciting area of operations research with practical relevance and impact. The theory so far developed is far from being satisfactory and being able to guide decisions in practice. However, the use of theory-based problem specific heuristics can improve the performance of "online systems" in practice. We have discussed here only relatively simple modules of much more complex logistics systems. The analysis and performance improvement of large systems is the "real goal" and requires further research efforts. In particular, it is not really clear at present how complex systems should be modeled, what the real objectives, constraints, etc. and what the interdependencies between them are. To put it in a nutshell, we do not know how to answer the following two questions convincingly:

What is a good system?
What is a good system performance?

ACKNOWLEDGEMENTS

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

REFERENCES

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

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

[3] T. Altıok. *Performance Analysis of Manufacturing Systems*. Springer Series in Operations Research. Springer, 1996.

[4] *AMSEL—A Modeling and Simulation Environment Library*, 1997. Developed at the Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB).
See `http://www.zib.de/ascheuer/AMSEL`.

[5] N. Ascheuer. *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems*. PhD thesis, Tech. Univ. Berlin, 1995.
Avail. at URL `http://www.zib.de/ZIBbib/Publications`

[6] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric travelling salesman problem with time windows. Preprint SC 97-11, Konrad-Zuse-Zentrum Berlin, 1997.
Avail. at URL `http://www.zib.de/ZIBbib/Publications`

[7] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving ATSP with time windows by branch-and-cut. Technical report, ZIB Berlin, 1998. (In preparation).

[8] N. Ascheuer, M. Grötschel, N. Kamin, and J. Rambau. Combinatorial online optimization in practice. *OPTIMA*, 57:1–6, March 1998.

[9] 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, ZIB Berlin, 1997. Avail. at URL `http://www.zib.de/ZIBbib/Publications`

[10] N. Ascheuer, S.O. Krumke, and J. Rambau. Competitive scheduling of elevators. Technical report, ZIB Berlin, 1998. (Paper in preparation).

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

[14] E. Feuerstein and L. Stougie. On-line single server dial-a-ride problems. (In preparation), 1998.

[15] A. Fiat and G.J. Woeginger. *Online algorithms—The state of the art*. Lecture Note in Computer Science. Springer, 1998.

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

[17] G. Galambos and G.J. Woeginger. Repacking helps in bounded space on-line bin-packing. *Computing*, 49:329–338, 1993.

[18] G. Galambos and G.J. Woeginger. On-line bin-packing—a restricted survey. *ZOR—Mathematical Methods of Operations Research*, 42:25–45, 1995.

[19] D.J. Guan. Routing a vehicle of capacity greater than one. *Discrete Applied Mathematics*, 81(1):41–57, January 1998.

[20] D.J. Guan and X. Zhu. Multiple capacity vehicle routing on paths. Technical Report, Department of Applied Mathematics, National Sun Yat-Sen University, Kaoshiung, Taiwan, March 1997.

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

[22] 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.

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

[24] C.C. Lee and D.T. Lee. A simple on-line bin-packing algorithm. *J. Assoc. Comput. Mach.*, 32:562–572, 1985.

[25] M.B. Richey. Improved bounds for harmonic based bin packing algorithms. *Discrete Applied Mathematics*, 34:203–227, 1991.

[26] A. van Vliet. On the asymptotic worst case behavior of harmonic fit. *Journal of Algorithms*, 20:113–136, 1996.

[27] T. Winter and U. Zimmermann. Discrete online and real-time optimization. In *Proceedings of the 15th IFIP World Computer Congress, Budapest/Vienna, August31–September 4*, 1998. (To appear).

KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK (ZIB)
E-mail address: <surname>zib.de