CHAPTER 5

Online-algorithms for reactive vehicle scheduling

G. Schmidt[a] and E. Jacob[b]

[a]*Prof. Dr. Günter Schmidt, Rechts– und Wirtschaftswissenschaftliche Fakultät, Lehrstuhl für Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik II, Universität des Saarbrücken, FRG*

[b]Dipl. -Kff. Evelina Jacob, *Rechts– und Wirtschaftswissenschaftliche Fakultät, Lehrstuhl für Betr iebswirtschaftslehre, insbesondere Wirtschaftsinformatik II, Universität des Saarbrücken, FRG*

Abstract

We investigate the question if online-algorithms can be used in a framework for knowledge-based reactive vehicle scheduling. Appropriate models are presented and corresponding research issues are discussed.

Keyword Codes: G.1.2; H.1.0
Keywords: dynamic scheduling, sequential solution approach; task-systems

1. Introduction

Problems in the domain of reactive vehicle scheduling are very often solved by human decision makers using some type of heuristic. On the other hand there exist algorithmic approaches which could support the problem solving process. With this paper we want to show how online-algorithms can be applied to reactive vehicle scheduling within a knowledge-based solution framework. We start with a short discussion of two major issues of vehicle scheduling which are based on reactive and predictive problem instances. Some reactive problem formulations can be solved by online-algorithms if they belong to certain problem classes. We give a survey on online-problem classes related to vehicle scheduling problems. Finally we investigate detailed problems and show how they can be reduced to models for which online-algorithms exist.

R. Kerr et al. (eds.), *Artificial Intelligence in Reactive Scheduling*

2. Reactive and Predictive Vehicle Scheduling

Vehicle scheduling is concerned with customer orders which have to be fulfilled using a set of vehicles. A customer order is defined by a set of goods which have to be transported from a given supply point to a given demand point. The objective of vehicle scheduling is to fulfil all customer orders with minimum cost. Depending on the practical application a number of additional constraints might be of importance, e.g. time windows for delivery, restricted loading capacity of the vehicles, etc.. A tour for a vehicle consists of the sequence of all customer delivery points. Due to increasing traffic on roads vehicle scheduling is a major problem for many enterprises.

Vehicle scheduling is a special scheduling problem. A survey on scheduling problems can be found e.g. in [BESW94]. Scheduling problems have two components: a predictive and a reactive one. Predictive scheduling means to generate a schedule in advance. It is assumed that the problem instance is well-known and does not change over time. In this respect the problem definition is deterministic and static. Predictive scheduling is carried out before the system starts operation. In case of vehicle scheduling that means predictive scheduling is carried out before vehicles start to work on customer orders. Reactive scheduling has to consider changes of the problem instance during operation and is concerned with a great volume of permanently changing data. This data has to be processed online. Sometimes predictive scheduling is also called static scheduling and reactive scheduling is called dynamic scheduling [Psa88].

How predictive and reactive scheduling can be integrated is shown in [Sch92]. Predictive scheduling is carried out by an offline-planning (OFP) modul and reactive scheduling by an online-control (ONC) modul. The OFP modul consists of an analysis, a construction and an evaluation component. First, the problem instance is analysed (A) in terms of objectives, constraints and further characteristics. In order to do this the first step for (A) is to describe the scheduling situation as detailed as possible. In a second step from this description a specific model has to be chosen from a set of possible scheduling models. The analysis component (A) is based upon knowledge-based approaches such as those used for problems like classification.

The problem analysis defines the parameters for the construction (C) phase. From the basic modul obtained in (A), a solution for the scheduling problem is generated by (C) using some generic algorithms. The result is a complete schedule that has then to be evaluated by (E). Here the question has to be answered if the solution is implementable in the sense that it meets business objectives and fulfils all constraints coming from the application. If the evaluation is satifactory to the user, the proposed solution will be implemented. If not, the process will repeat itself until the proposed solution delivers a desirable outcome or no more improvements appear to be possible in reasonable time.

The construction component (C) of the ACE loop generates solutions for OFP. It bases its solution upon exact and heuristic problem solving methods. Unfortunately, with this approach we only can solve static representations of quite general problems. The dynamics can at best be only approximately represented. In order to obtain the necessary answers for a dynamic process, the evaluation component (E) builds up descriptive models in the form of queuing

networks at aggregated levels or simulation on a specific level. With these models one can evaluate the various outcomes and from this if necessary new requirements for problem solution are set up.

Having generated a feasible and satisfactory predictive schedule the ONC module will be called. This module takes the OFP schedule and translates its requirements to an ONC strategy, which will be followed as long as the scheduling problem remains within the setting investigated in the analysis phase of OFP. If temporary disturbances occur, a time dependent strategy in the form of an ad-hoc decision must be devised. If the interruption continues for such a long time that a new schedule needs to be generated, the system will return to the OFP module and seek for an alternative strategy on the basis of a new problem instance with new requirements and possibly different objectives within the ACE loop. Again a new ONC strategy has to be found which will then be followed until again major disturbances occur.

Using the above approach to combine predictive and reactive scheduling it is assumed that the results for predictive scheduling are used as a guideline for the reactive part. In order to carry out some predictive vehicle scheduling some customer orders have to be given before the vehicles start operation. Sometimes this cannot be assumed; in this case customer orders arrive online during operation. In the remainder of this paper we will concentrate on the latter problem formulation for reactive vehicle scheduling.

We assume that there is a set of vehicles which have to serve given customer orders which are not known in advance. With this assumption predictive vehicle scheduling is of minor importance. A customer order is defined by a supply point and a demand point. The supply point represents the geographical location where some goods are loaded on the vehicle. The demand point represents the location where this goods are to be delivered. Each vehicle has to serve a set of customer orders starting from a given position of the vehicle and then going to the demand point visiting the supply point before. The objective is to serve all customer orders with minimum cost. In the next section we analyse how online-algorithms can be used to solve this kind of problem.

3. Online-Algorithms

Online-algorithms schedule the operation of a system having only local information. Being in a certain state the system has to carry out a given task. After processing this task the system enters a new state and the online-algorithm can take a decision on changing this state. Only information about the current task is available; future tasks are not known in advance. All processing and decision making are carried out sequentially. With this approach only suboptimal solutions can be guaranteed. With respect to [Alb93] and [BD91] we define an online-algorithm as a sequential solution approach where tasks have to be processed in the sequence of their appearance. No information is given about the number of tasks and the kind of each individual task. Online-algorithms differ from offline-alorithms such that for an offline-algorithm the number of tasks and the relevant information of each individual task is known in advance. With offline-algorithms optimal solutions can be generated.

The performance of an online-algorithm can be analysed by comparing its solution with the solution of an optimal offline-algorithm [ST85]. This evaluation is called competitive analysis [KMRS88]. Let the sequence of appearing tasks be called σ. For each σ corresponding cost are generated depending on the states of the system. Let $K_{on}(\sigma)$ be the cost which are generated by an online-algorithm for the task sequence σ and let $K_{off}(\sigma)$ be the minimum cost which are generated by an optimal offline-algorithm for σ. An online-algorithm has the performance c if there is a constant k such that the cost of the task sequence σ is:

$$K_{on}(\sigma) \leq c \cdot K_{off}(\sigma) + k.$$

In the literature different online-problem classes for online-algorithms are investigated. The most important classes are:

- Task-Systems,
- k- Server-Problems and
- List- Update-Problems.

The most general formulation of online-problems are task-systems [BLS92]. They consist of a set $S = \{1,2, ..., n\}$ of states, i.e. the system can comprise $n = |S|$ states. The cost of state changes are given by a n x n cost matrix d: S x S ->IR. The cost matrix fulfils the triangle inequality and all cost are positive. If there is no state change no cost for the system will be assumed. Let d_{ik} be the cost of the change from state i to state k. A task-system is called metric if for all states i,k the equality $d_{ik} = d_{ki}$ holds. It is assumed that the task-system can operate on an arbitrary set of tasks $T = \{T_1, T_2, ..., T_m\}$. For each operation there exist a processing cost depending on the state of the system. The processing cost c: S x T ->IR are influenced by the task and the state of the system. For each task T_j it exists a n-dimensional vector $(T_j(1), T_j(2), ..., T_j(n))$. Each $T_j(i)$ defines the cost of processing of the task T_j in state i. At each point of time the task-system is in only one of n possible states. Before starting processing it is assumed that the system is in an initial state s_{t-1} at time t-1. The online-algorithm decides at time t whether a state change has to be carried out or not. After this a task appears and at time t+1 the system is starting operation on this task. The cost consist of the cost for state changes and the cost for processing being in some state. For times t+2, ..., t* there is an arbitrary but finite number of future state changes and tasks which have to be processed. To calculate the total cost from the decisions of the online-algorithm all individual cost of decision and processing have to be sumed up. It is known that there exist online-algorithms for task-systems which have a competitive performance of (2n-1) [BLS92], [CDRS90].

The k-server-problem which is introduced by Manasse et al. [MMS88]. It consists of scheduling k-mobile servers which cover vertices in a complete directed n vertex graph G. The edge lengths are non negative and fulfil the triangle inequalitiy. A task sequence consists of vertices of G to be served. In response to each task a server must be moved to the requested vertex, unless a server is already present. The goal is to minimize the total distance travelled by all servers. The k-server-problem is called symmetric if the distance from vertex i to vertex j equals the distance from vertex j to i for all vertices i and j. Otherwise it is called

asymmetric. A detailed description of the k-server-problem is given in [CDRS90]. It is known that for this problem class there exist simple online-algorithms of competitive performance k.

The list-update-problem consists of maintaining a set of items as an unsorted linear list [IRWS91]. A list of n items is given. Each task is an access to an item in the list. The list-update algorithm starts at the front of the list and searches linear through the items until the desired item is found. Accessing the i-th item in the list induces the cost of one. At any time a list-update algorithm may exchange adjacent items in the list. Immediately after an access the requested item may be moved at no extra cost to any position closer the front of the list. These exchanges are called free changes. All other exchanges have unit cost and are called paid exchanges. The goal is to serve the request sequence such that the total cost is as small as possible. The best online-algorithms can reach the competitive performance of 2 [ST85].

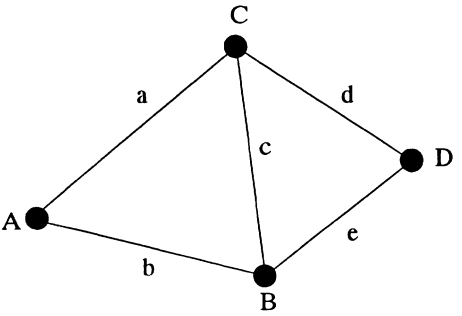
In the next section we will show how reactive vehicle scheduling can be reduced to task-systems and k-server-problems, respectively.

4. Models for Reactive Vehicle Scheduling

In the following we show that online-algorithms can be applied to reactive vehicle scheduling problems. Let G be a weighted graph with $G = \{V, E, w\}$. $V = \{1, 2, ..., n\}$ represents all geographical locations of a road map, E represent all connections of the locations. Let w be a cost function $w: V \times V \rightarrow \mathbb{R}$, i.e. each edge from E is given a non negative weight which represents the travel time between two corresponding locations. There are three explicit nodes in the graph; these are starting point, supply point and demand point. A customer order is specified by the supply and the demand point. The starting point is the actual position of the vehicle. Before working on a new customer order the current customer order has to be processed. We want to carry out a tour for each vehicle such that the sum of the cost is minimized and all customer orders are served.

Now we will formulate the vehicle scheduling problem in terms of task system problems. Let G be a graph as defined above. Let us assume that the weights of the edges do not change over time, i.e. there is constant traffic flow on the road net. Now states, tasks, state changing and task processing cost have to be defined. A state is represented by the node of the graph where the vehicle is located when a customer order arrives, i.e. S = V. A state change is equivalent to a change of the location from node i to node k. The cost of the changes can be calculated from the given cost function w finding the shortest path between two nodes which represent the state change. A task T_j is a customer order to transport some goods from a given supply node to a given demand node starting from the node which represents the current state of the system. For each customer order a vector with n entries is given. Each entry describes the cost for carrying out the order depending on the state of the system. Again the processing cost are calculated by the shortest path from the current state visiting the supply point and terminating at the demand point. The set of possible customer orders is related to the set of the nodes of the graph, i.e. each node of the graph can be a supply and a demand point.

To demonstrate the modelling approach we want to consider the following example problem. The road network is shown in the following figure.



We have a set S of four states which are A, B, C, and D. They represent all possible starting nodes for a vehicle in the graph. The cost matrix w^S of state changes is given in the following table. We assume that all weights are positive and the triangle inequality holds.

Matrix w^S		j =			
		A	B	C	D
i =	A	0	b	a	P*)
	B	b	0	c	e
	C	a	c	0	d
	D	P*)	e	d	0

*) P = min {(b+e), (a+d)}

Each entry of the matrix represents state changing cost between all possible states of the system. For the calculation of the delivery cost we simplify the exposition. We assume that for each customer order supply point and demand point are at the same location. Now the cost vector $T_j(i)$ of each task T_j is given by:

$$T_A(i) = (0, b, a, P)$$
$$T_B(i) = (b, 0, c, e)$$
$$T_C(i) = (a, c, 0, d)$$
$$T_D(i) = (P, e, d, 0)$$

The entries of each vector $T_j(i)$ represent the cost of travelling to the demand point j being in any possible starting point (state) i.

Now consider an online-algorithm at work. Starting with time zero at each even point of time the algorithm has to make a decison on state changes and at each odd point of time the

algorithm has to work on some incoming customer order. Let B be the state at point of time 0 and T_D, T_B and T_C the sequence of tasks arriving at points of time one, three and five. Being in state B task T_D is to be processed. The processing cost are e and the resulting state is D. Now the online-algorithm is taking a decision on a state change from D to C with state changing cost d. Now task T_B has to be carried out with processing cost c. The resulting state is now B and we want to assume that the online-algorithm does not take any further decisions on state changes. Now task T_C has to be processed with cost c. Summing up all state changing and processing cost we have $(e + d + 2 \cdot c)$ cost.

We did assume that there is only one vehicle for serving all customer orders. If we enlarge the problem that we have a fixed number of vehicles for serving the customers then we end up with the k-server-problem formulated above. In this case the decision is not taken by an individual vehicle but there is a general dispatcher for scheduling the vehicles. The situation still remains deterministic and static as far as the traffic flow on the road net is concerned. In order to introduce some dynamic traffic characteristics we have to introduce weights of the edges of the graph which depend on time. With this assumption we can model changing traffic flow as it occurs in cases of queues or traffic jam on the roads. In order to solve this kind of dynamic problem we have to consider a vector for each task which does not depend only on the considered states but also on the considered time.

At the end of this section we want to make a few comments on quite general reactive vehicle scheduling problems. If some customer orders are known in advance the predictive scheduling problem can be solved by an offline-algorithm. From this offline-algorithm a partial tour is known for each vehicle. Coming now to the reactive situation means that each vehicle is trying to follow its predictive tour as close as possible as long as there is no change in the problem instance. The online-problem is formulated in the same sense as the offline-problem now with potentially changing problem parameters as far as customer orders and weights of nodes are concerned. With this assumption two kinds of tasks exist; some are known in advance and some arrive in the future. It remains open how an online-problem formulation as this one falls within the scope of online-algorithms.

5. Conclusions

Using the assumption of a static traffic flow system it could be shown that reactive vehicle scheduling can be formulated as an online-problem which can be solved by online-algorithms. The problem formulation which is used makes it possible to apply models for task-systems and k-server-problems. A competitive analysis of the performance of online-algorithms for these problem classes with n states shows that algorithms with the performance of (2n-1) or better exist.

As far as knowledge based reactive vehicle scheduling is concerned it has to be investigated which algorithms are used by the human problem solver and whether these approaches are within the competitive performance value which can be achieved for these problem classes from a theoretical point of view. On the other hand the analysis of online-algorithms gives also insights for the human problem solver to come to better results. It remains to investigate

how the knowledge for the problem solution of the human decision maker can be harmonised with theoretical consideration for the design of online-algorithms.

In order to improve the performance of online-algorithms they can be modified in two ways. Either certain features of lookahead have to be incorperated or a randomization of the algorithms should be used. The possibilities of lookahead and randomized algorithms are sketched in [BBKTW90] and [Spi77]. Further it should be investigated how dynamic scheduling algorithms following the ideas presented in [PY91] can help to solve vehicle scheduling problems. For a real life application of vehicle scheduling some dynamic features are represented by a probabilistic queueing model and a dynamic cost function. With this traffic flow can be varied from normal to congested [FPMJ94]. The vehicle scheduling system which contains predictive and reactive components has been implemented as a prototype.

6. References

[Alb93] Albers, S. : The Influence of Lookahead in Competitive On-Line Algorithms, Dissertation, Universität des Saarlandes, 1993

[BBKTW90] Ben-David, S., Borodin, A., Karp, R. M., Tardos, G., Wigderson, A.: On the power of randomization in on-line algorithms. In Proc. 22nd Annual ACM Symposium on Theory of Computing, 379-386, 1990.

[BD91] Ben-David, S., Dichterman E. : Derandomizing Online Algorithms (Extended Abstract), Technion - Israel Institute of Technology, Computer Science Department, Technical Report #691, October 1991

[BESW94] Blacewicz, J., Ecker, K., Schmidt, G., Weglarz, J.: Scheduling in Computer and Manufacturing Systems, Springer-Verlag, Second revised Edition, 1994

[BLS92] Borodin, A., Linial, N., Saks, M. E.: An optimal Online Algorithm for Metrical Task Systems, Journal of the Association for Computing Machinery, Vol. 39, No. 4, 745-763, October 1992

[CDRS90] Coppersmith, D., Doyle, P., Raghavan, P., Snir, M.: Random Walks on Weighted Graphs, and Applications to On-line Algorithms, in 22nd STOC, 369-378, May 1990

[FMPJ94] Fischer, K., Müller, J. P., Pischel, M., Jacob, E.: Modeling Traffic Jams in a Logistics Simulation Environment, Proceedings of the 1994 European Simulation Multiconference, ESM94, Barcelona, 883-887, Spain 1994

[IRWS91] Irani, S., Reingold, N., Westbrook, J., Sleator, D.D.: Randomized competitive Algorithms for the List Update Problem. In Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms, 251-260, 1991

[KMRS88] Karlin, A. R., Manasse, M. S., Rudolph, L., Sleator, D. D.: Competitive snoopy Caching, Algorithmica, 3(1):79-119, 1988

[MMS88] Manasse, M. S., McGeoch, L. A:, Sleator, D. D.: Competitive Algorithms for On-line Problems, in Proc. 20th Annual ACM Symposium on theory of Computing, 322-333, 1988

[Psa88] Psaraftis, H., N.: Dynamic Vehicle Routing Problems, in: Golden, B. L., Assad, A. (eds.), Vehicle Routing: Models and Studies, Elsevier Science Publisher B. V. (North-Holland), 223-248, 1988

[PY91] Papadimitriou, Chr. H., Yannakakis, M.: Shortest path without a map. Theoretical Computer Science 84, Elsevier Science Publisher B. V. (North-Holland), 127 -150, 1991

[Sch92] Schmidt, G.: A Decision Support System for Production Scheduling, Journal for Decision Systems 1 (2-3), 243-260, 1992

[Spi77] Spirn, J. R.: Program Behavior: Models and Measurements. Elsevier, New York 1977.

[ST85] Sleator, D. D., Tarjan, R. E. : Amortized efficiency of List Update and Paging rules. Communication of the ACM, 28, 202-208, 1985