[Schö07a] Schönberger, J.; Kopfer, H.: Autonomous Decision Model Adaptation and the Vehicle Routing problem with Time Windows and Uncertain Demand. In: Windt, K.; Hülsmann, M. (eds.): Understanding Autonomous Cooperation and Control in Logistics. Springer, Berlin Heidelberg, 2007

1 Autonomous Decision Model Adaptation and the Vehicle Routing Problem with Time Windows and Uncertain Demand

Jörn Schönberger, Herbert Kopfer

Collaborative Research Centre 637, Chair of Logistics, University of Bremen

1.1 Introduction

The instruction of resources in logistic systems in order to ensure an effective as well as efficient usage is a very sophisticated task. At lot of data and requirements have to be considered simultaneously. For this reason computerized decision support (Makowski 1994) is strongly recommended (Bramel and Simchi-Levi 1997; Crainic and Laporte 1998).

A prerequisite for the application of computerized methods to tackle logistics decision problems is the representation of the current decision situation in a formalized fashion, a so-called decision model, normally described in terms of mathematical expressions (Williams 1999). Such a model, often of optimisation type, is than tackled by, typically heuristic, algorithms (Ibaraki et al. 2005; Michalewicz and Fogel 2004) in order to derive one (best possible) solution that is the instruction predicting the future activities in the logistic process execution system.

If the decision problem in the real world changes, the existing problem model becomes void and a re-modelling is required. Additional knowledge about the current system state and performance enters the model in order to propagate the problem changes to the used decision support system. However, this topic has received only minor attention so far in the scientific literature although it is of very high practical relevance and importance.

In this contribution, we investigate generic procedures and rules for an automatic feedback controlled adaptation of decision models for a variant of the well-known Vehicle Routing Problem with Time Windows. The

considered problem differs from the generic problem because the customer sites, which require a visit, emerge successively over time so that a plan revision becomes necessary. In Subsection 1.2 we present the considered decision problem in more detail. Subsection 1.3 introduces the algorithmic framework for an autonomous adaptation of the decision model and in Subsection 1.4 we prove the framework's general applicability within numerical simulation experiments.

1.2 The Vehicle Routing Problem with Time Windows and Uncertain Demand

This section is about the investigated decision problem. The problem is non-stochastic, e.g. requests are released consecutively but we do not know anything about their arrival times. In Subsection 1.2.1 we survey the scientific literature related to the problem considered here. Subsection 1.2.2 outlines the problem informally. The life cycle model of a request is presented in Subsection 1.2.3 and the decision problem that requires a solving whenever at least one additional request arrives is stated in 1.2.4. The construction of artificial test cases developed for a numerical simulation of selected problem instances is subject of Subsection 1.2.5.

1.2.1 Literature

Gendreau and Potvin (1998) survey vehicle routing and scheduling problems with incomplete planning data. Psaraftis (1988) and Psaraftis (1995) discuss the differences between vehicle routing and scheduling problems with deterministic and with probabilistic or incomplete planning data.

Jensen (2001) understands *robust planning* as the generation of plans that maintain their high or even optimal quality after subsequent modifications. He defines *flexible planning* as the generation of plans whose quality does not significantly decrease after the execution of algorithmic re-scheduling and alterations of the so far used plans.

Jaillet (1998), Jaillet and Odoni (1988) as well as Bianchi et al. (2005) propose a robust transport scheduling approach. They construct optimal *a-priori-routes*. Such a route has a minimal expected length among all possible routes through the potential customer sites. However, this approach assumes that probability distributions of the actual demand at the customer sites are known. As soon as a vehicle has visited a customer site and the corresponding demand becomes sure it has to be decided whether a replen-

ishment visit at a depot has to be executed before the next customer (again with uncertain demand) is met.

Flexible planning approaches do not require any knowledge about future events. An existing plan is updated consecutively and reactively. Sequenced planning problem instances P_i are solved one after another. Such a sequence of decision problems P_1, P_2,… is called an *online planning problem* according to Fiat and Woeginger (1998). A survey of online vehicle routing and scheduling problems is provided by Krumke (2001). Special cases are addressed by Ausiello et al. (2001). Theoretical results for online repairmen dispatching strategies are found in Bertsimas and van Ryzin (1989) as well as Irani et al. (2004).

Slater (2002) as well as Gayialis and Tatsiopoulos (2004) propose dispatching systems for transport planning tasks. Ghiani et al. (2003), Gendreau et al. (1999), Fleischmann et al. (2004), Séguin et al. (1997) and Gutenschwager et al. (2004) investigate dispatching systems in which decisions have to be derived in real time without any delay.

Gendreau and Potvin (1998) give a survey of applications for vehicle routing type problems requiring a re-planning. Brotcorne et al. (2003) report about an application of operations research methods to a relocation problem in medical rescue service. Chen and Xu (2006) as well as Savelsbergh and Sol (1999) investigate sophisticated algorithms for the repeated plan update in real world transport applications.

1.2.2 Informal Problem Description

Similar to the vehicle routing problem with time windows (VRPTW), we are looking for a decision support system that generates automatically a set of route for the available vehicles so that they fulfil customer orders and then travels back to a depot. Time windows restrict the intervals in which a customer order can be served. The problem we are investigating in this contribution comes along with three generalisations compared with the generic VRPTW version:

1. (SOFT TIME WINDOWS) A customer site is allowed to be visited after the corresponding time window has been closed. However, lateness will produce additional penalty costs to be paid to the customer. The amount pen(δ) to be paid increases linearly with the temporal distance δ from the arrival time to the latest allowed arrival time but is limited to a certain amount PENMAX.

2. (SUBCONTRACTION) Each customer request can be subcontracted. In such a situation the considered company, that maintains the fleet to be routed, orders another logistics service provider (LSP) to fulfil a particular request. The LSP receives a certain amount for this service but ensures that the request is fulfilled within the specified time window.
3. (UNCERTAIN DEMAND) Only a subset of all requests to be fulfilled is known to the planning authority at the time when the subcontraction is decided and the routes for the own vehicles are generated. Whenever one or more additional requests become known, it has to be decided whether these requests are subcontracted or not. In the latter case, the additional requests are integrated into the so far existing routes. For some of the requests, so far expected to be served by an own vehicle, subcontraction can become more attractive now. This is the result of a postponement of these requests in order to serve a recently released request in time. Attention has to be paid that a once subcontracted request cannot be reintegrated into the route of an own vehicle because the contract with the ordered LSP is binding.

We refer to this decision problem as the vehicle routing problem with time windows and uncertain demand (VRPTWUD). The goal is to find a transportation plan (Crainic and Laporte 1997) that describes which requests are served internally or externally by LSPs and how the requests are served by the own vehicles.

The SOFT TIME WINDOWS property allows a more flexible route generation because minor window violations are penalized only slightly. Due to the SUBCONTRACTION property, requests not fitting with the remaining portfolio do not have to be considered in the route generation so that a more advantageous request consolidation is achieved. However, the UNCERTAIN DEMAND issue requires a transport plan adaptation every time when additional requests are released.

Although each particular request in allowed to be late, there is a general guideline that predicts a global punctuality. More detailed, for a given transportation plan update time t, the percentage of the requests served in time has to be larger than p^{target}. We consider only those requests completed within the last t^- time units and whose completion is scheduled within the next t^+ time units. This means, only recent service quality information are used because the relevant consideration time window $[t-t^-;t+t^+]$ moves with ongoing time.

The goal of the planning support to be developed is to establish a planning system that allows the generation and repeated update as well as adaptation of flexible transportation plans for the field teams including decisions about externalization of selected requests. The flexibility is important because the customer requests are received successively and their arrival times cannot be predicted or forecasted so that only a reactive transport plan revision is realizable. Furthermore, in order to maintain the flexibility of the transport plans even in situations with an extreme workload, it is allowed to violate the agreed time windows but the corresponding customers are paid compensation.

1.2.3 Online Request State Update

In order to consider the successively arriving additional requests, we propose to update the existing transportation plan reactively after the additional requests become known (Schönberger and Kopfer 2007).

Let t_i denote the i-th time when additional requests become available and let $R^+(t_i)$ represent the set of additional requests, released at t_i. After the last transportation plan update at time t_{i-1}, several requests have been completed. These requests are stored in the set $R^C(t_{i-1},t_i)$. Then the request stock $R(t_i)$ at time t_i is determined by $R(t_i) := R(t_{i-1}) + R^+(t_i) - R^C(t_{i-1},t_i)$.

The life of a single request r consists of a sequence of states to which r belongs. Initially, when r enters the transportation system it is known but not yet scheduled (F). If r is assigned to an own vehicle for execution it is labelled by (I) or by (E) in case that r is assigned to an external service partner. A request whose completion work at the corresponding customer site has been started but not yet finished is labelled as (S). The final stage (C) of r indicates that r is completed.

Every time a transportation plan update becomes necessary, the current states of known requests from $R(t_i)$ are updated. The state (F) is assigned to all new requests from $R^+(t_i)$. For all requests contained in $R^C(t_{i-1},t_i)$, their state is updated from (I) or (E) to (C) and requests whose on-site execution have been started but not yet completed receive the new state (S) that replaces their former state (I) or (E). Now, the scheduling algorithm is started that carries out the necessary transportation plan updates. From the updated transportation plan the information about the intended type of request execution of all requests labelled as (F) or (I) is taken. The state of an (I)-labelled requests is updated to (E) if it has been decided to out source this request. Otherwise, the state of this request remains unchanged. Finally, all (F)-labels of externalized requests are replaced by (E)-labels

for subcontracted requests and (I)-labels replace the (F) labels for the remaining requests from $R^{+}(t_i)$.

1.2.4 Statement of the Scheduling Problem

The decision whether a request should be assigned to an own team or given to an external partner cannot be solved uniquely for each request. A complex decision problem must be solved every time the currently valid transportation plan has to be updated, considering simultaneously all assignable requests, which are labelled by (I) or (F). It has to be decided for all these requests whether they are definitively subcontracted and given to a service partner for execution or if they should be assigned for the first time to one of the available own vehicles represented by the elements of set V(t). In order to find the minimal cost assignment, we propose the following optimisation model.

Let $\Omega(t)$ denote the set of all possible request sequences $p=(p_1,\ldots,p_{n(p)})$ representing the order in which the contained customer requests, selected from R(t), are visited. The vehicle v selected for request r in the last transportation plan is denoted as $\Psi(r)$. If r is labelled as (I) then $\Psi(r)\in V(t)$, otherwise $\Psi(r)=\{\}$.

We assume that each $p\in\Omega(t)$ holds for the following two properties:

- The final entry $p_{n(p)}$ of p refers to the depot to which all vehicles return.
- If the first entry p_1 refers to a request labelled currently as (S) then the departing time from p_1 cannot precede the finishing time of this request.

The following two binary decision variable sets are used to code the necessary decisions. The variable u_p is set to 1 if and only if sequence $p\in\Omega(t)$ is selected for vehicle $v\in V(t)$. Furthermore, y_r is set to 1 if and only if request $r\in R(t)$ is subcontracted.

We are looking then for instantiations of the above decision variables that minimizes the costs $C(\{x_{pv}\},\{y_r\})$ but considering that

1. Each vehicle is assigned to exactly one (maybe an empty) path from $\Omega(t)$.
2. Each request is contained in at most one of the selected paths.
3. A request r labelled by (S) cannot be assigned to another vehicle as $\Psi(r)$.

4. If request r is labelled by (E) then $y_r=1$.
5. If vehicle v is assigned to p then p_1 must correspond to the current location of vehicle v.

We desist from giving the formal mathematical statement of the above five constraints since we do not need them in the remaining presentation.

The objective function $C(\{x_{pv}\},\{y_r\})$ calculates the costs associated to the instantiations of the two decision variable sets. It is the sum of the travel costs for the own deployed vehicles plus the sum for subcontraction fees and penalties to be paid for late arrivals at customer sites. Therefore, it denotes the costs for the associated transportation plan.

1.2.5 Artificial Test Cases

In order to evaluate different dispatching approaches and to control the severeness of the observed scenario, we have derived a set of artificial test instances. Each instance is defined by a special instantiation of a set of parameters. The adjustment of these parameters models different scenarios.

Two different kinds of routing scenarios with successively arriving requests are reported in the scientific literature. In the first scenario type, the number of demands that are released during a specific time interval remains unchanged. It is possible to adapt the available resources in such a situation so that all additional demands can be served in time. For this reason, such a scenario is called a *balanced scenario*. Examples can be found in Pankratz (2002), Lackner (2004) and Mitrović-Minić et al. (2004). In case that the number of additionally released demands during a specific time interval varies, the scenario is denoted as a *peak scenario*. Here, it is hardly possible to adapt the available resources in advance. Gutenschwager et al. (2004), Sandvoss (2004) as well as Hiller et al. (2005) deal with real world examples that do not allow a parameterization and classification for scientific analysis purposes.

In order to determine a competitive and comparable tariff for calculating the fare to be paid to an LSP for subcontraction, we compare the travelled and the demanded distances in the best-known VRPTW solutions for the Solomon benchmark Problem (Solomon, 1987) as described in detail in Schönberger (2005). For each request, the amount of the subcontraction costs is calculated and assigned to the particular request.

To simulate peak scenarios we first generate a balanced stream of incoming customer demands over the complete observation time period. A second stream is generated for a part of the observation period. Both streams

are than overlaid so that during the period in which the second stream is alive, the balanced stream is interrupted and a higher number of requests must be scheduled.

The balanced stream of incoming demands for the observation period $[0,T_{max}]$ is generated by successively drawing requests from the Solomon instance P. At time $t^{rel}=0$, n_0 demands are drawn randomly from P. Then, the release time is updated by $t^{rel}:= t^{rel} +\Delta_t$. For this new release time, n demands are drawn from P at random. For each selected demand r, its release time is set to t^{rel}. The original service time window $[e_r,l_r]$ of r is replaced by $[t^{rel}+e_r,\ t^{rel}+l_r]$. Additional demands are generated as long as $t^{rel}\leq T_{max}$.

The second stream of demands is released to simulate a peak of demands. For the first generated release times $0, \Delta_t ,2\Delta_t ,\ldots, n_1\Delta_t$ no demands are released. For the next n_2 release times $(n_1+1)\Delta_t,\ldots,(n_1+n_2)\cdot\Delta_t$ Δ_m demands are specified as described above. For the remaining release times, no additional demands are given.

All vehicles specified in P can be used.

Consequently, each scenario is described by the triple (P, d^{peak}, Δ_m). In this investigation, we use the four Solomon cases R103, R104, R107 and R108 to generate request sets. Furthermore, it is $n_0=n=50$ and $\Delta t=100$ time units. The peak duration has been set to $d^{peak}=200$ time units and the peak high is fixed at $\Delta_m=100$ additional request.

1.3 Model-Based Planning in Dynamic Environments

This section is dedicated to the presentation of an algorithmic framework realizing an automatic, autonomously controlled adaptation of a generic decision model to a particular situation. In Subsection 1.3.1, we compile the open issues that contradict the realization of an automatic adaptation of a decision model. Subsection 1.3.2 describes generic approaches for modifying an optimisation model. Subject of Subsection 1.3.3 is the proposal of an iterative procedure that controls the adaptation of a logistic process exploiting a feedback-triggered adaptation and in 1.3.4 explicit adaptation rules are presented for the VRPTWUD.

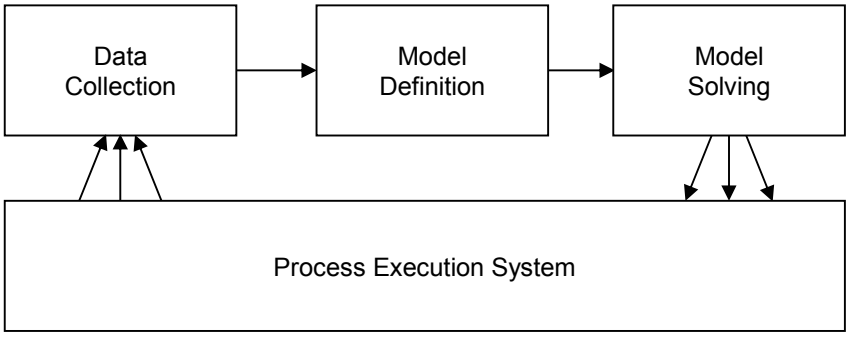

Fig. 1: Model-based replanning

1.3.1 Model-Based Replanning

The re-active planning of logistic processes in a changing environment for adapting a process to a new situation requires the repeated execution of the three basic steps data collection, model statement and model solution. Fig. 1 represents the steps to be executed. As soon as relevant changes are detected that affect the so far executed process, the available data are collected and prepared. From these data, a new decision model is set up and, next, this model is solved. Finally, the new process (the solution of the recently solved decision problem) is broadcasted back into the logistic system for execution.

The re-start of the planning cycle is a response to a modification in the underlying real world decision problem. Clearly, these modifications have to be considered in the compilation and solving of the new decision model. Up to now, some technical as well as conceptual challenges have to be overcome before the autonomous and appropriate redefinition and solving of a decision model can be exploited to the largest possible extend. The collection of the required problem data and the re-setup of the decision model as well as its solving are faced with some deficiencies that have not been solved satisfactorily yet.

Data Collection. The technical availability of the data is high but a lot of effort and intelligence has to be spent in order to get helpful, consistent and reliable as well as complete data for the next instance of a decision model in an online decision problem. In a deterministic environment, there is enough time to discuss the adequacy of the data and to look for missing data but if the changing environment predicts a rapid and reliable reaction on environment changes and requires the revision of former decisions then

a strong automatic data pre-processing support is unconditional necessary. Contributions from artificial intelligence and/or ideas methods borrowed from Data Mining should be incorporated to calibrate, to complete the collected data, and to prepare the setup of the next decision model.

Automatic Adaptation of the Planning Goal. The objective belonging to the model of the next optimisation problem instance requires an automatic, feedback-triggered adaptation with respect to the congruence with superior objectives that predict the development of the logistic system over a longer time horizon. Consequently, an analysis of the currently collected data with respect to the current system performance is necessary in order to decide what the next goal to be followed will be and which data are required for the definition of the objective function.

Vectors of Data. Beside the consideration of current problem data, it is necessary to consider both its development direction and the velocity of change as well. Therefore, additional data are necessary in order to assess and describe the system evolution adequately.

Data Extraction and Data Interpolation. Data provided by information technology-based services have to undergo a substantial pre-processing before these data can be used for the setup of the next model. Redundant data have to be eliminated and missing data must be integrated.

These three items describe special requirements for the collection and preparation of the modelling of a decision situation in an online scenario.

Model Building. In order to allow the automatic re-definition of an adequate decision model, the so far existing straightforward techniques require some extensions.

Flexible Representation of Decision Alternatives. The setup of a particular decision model is currently compromised by inflexible representations of the decision alternatives. Here, future research efforts should be spent to the development of more flexible and adaptable representation methods so that a modified decision problem can be coded easily.

Automatic Adaptation of the Decision Alternative Evaluation. It is necessary to re-think the worthiness of a certain decision alternative after the problem under consideration has changed. Often, the usefulness of a certain alternative is given only if some assumptions are met (enough resources, enough time, …). If these assumptions become void, the worth of a particular decision alternative runs into danger to alter.

Automatic Feedback-Controlled Adaptation of the Search Space. The search space of an optimisation problem instance represents the decision

alternatives currently available. In order to identify adequate solutions that support the strive for the fulfilment of longer term planning goals, it is necessary to prune some alternatives that are currently feasible but, on the long run, to not lead to the intended system development. Furthermore, additional solution alternatives should be allowed if the existing solution alternatives do not comply with the current situation.

Model Solving. The derivation of a solution (proposal) is left to automatic software procedures (algorithms). They have been applied successfully to problems in static environments for several decades. In order to use the observed findings in scenarios with varying system environments, the software procedures must undergo some specific modifications in order to apply them successfully to process adaptations.

Autonomous Re-Parameterization and Re-Configuration. In order to enable the software to deal with quantitative as well as with qualitative differences in problem instances it is unconditional necessary to equip the software with a problem interpreter to analyse the current decision problem. Furthermore, depending of the results of the problem analyse, the software has to decide autonomously about the adjustment of their search parameters as well as their hardware usage.

Simultaneous Addressing of Feasibility Recovery and Update Improvement. Decision software for process updating in systems with uncertain problem data has to consider simultaneously the recovery from event-based infeasibility as well as the improvement or even optimisation of the updated processes in order to achieve highest process quality and reliability.

Limitation of Decision Times. In order to provide the logistic system with an adapted process proposal after a process disruption event the update time has to be kept as short as possible. In case that the pre-specified update answer time is very short (or even close to zero), it is tried to provide a feasible update first and to improve it afterwards if time is still available (Gutenschwager et al. 2004).

The application of software decision support and planning algorithms is unconditional necessary to cope with the complexity of recent decision problems. The availability of an adequate final decision model is a prerequisite for the successful application of automatic software procedures. An enlargement of the quality of the provided data is currently subject of different research disciplines, e.g. data mining (Clifton et al. 2002). Furthermore, certain researchers (Holzer 2003) also address the automatic reconfiguration and the speed-up of decision algorithms. However, the ex-

ploitation of feedback-information from the underlying process execution system for the adaptation of formal decision (e.g. optimisation) models to the currently observed system state and to the currently waiting decision problem is not yet subject of any scientific work. In the remainder of this contribution, we propose some generic ideas to target this topic.

1.3.2 Generic Approaches for Optimisation Model Adaptation

Any formal optimisation problem consists of a description of the search space that includes all feasible solutions and of an objective function that assigns a numerical or vector value to each element of the search space. One or both of these components can be subject of modifications in order to adjust and adapt an existing generic optimisation model according to externally given rules.

Modification of the Objective Function. The main reason for defining an objective function is to evaluate each solution alternative in order to distinguish different solutions as well as to rank them. An automatic solving procedure exploits the objective function and uses the objective function value(s) as a feedback when

a) comparing different branches for further exploration of the search space and selecting one branch to be searched next or

b) pruning some branches from a further exploration due to a reliable estimation about an unsatisfying solution quality to be found in this branch.

For this reason, a modification of the objective function can be interpreted as an adaptation of the search direction of the applied search algorithm. The main goals of this adaptation kind are (i) guiding the search process away from solutions that are currently unattractive and (ii) allowing the search algorithm to find adequate solutions quicker.

Adaptation of the Search Space. The search space can be modified by excluding (pruning) solution alternatives from the search space defined in a given model or adding additional solutions to the proposed set of solution alternatives. The pruning of solutions can be achieved by strengthening existing restrictions or adding new restrictions and the search space can be enlarged by relaxing or skipping so far valid restrictions. Pruning of solutions aims at prohibiting the selection of certain solutions. This technique is often used if the evaluation scheme cannot be used effectively to prevent the selection of low quality solutions. The promotion of additional solution

alternatives is a response if no adequate solution can be identified in the so far maintained search space.

1.3.3 Basic Algorithm

The adaptation of the decision model of the current problem instance should be based on the current system performance measured in term of the instantiation of one or more key indicators. Therefore, the current system performance is determined and the observed values are then compared with some major guidelines predicted by a superior authority (SA) that has the right to instruct the planning authority (PA). This concept is shown in a formalized fashion in Fig. 2. Initially, SA receives feedback information, e.g. performance information, from the process execution system (1). It compares the observed performance values with the values predicted by SA. In case that a discrepancy is detected, it instructs PA to adapt its decision rules in a fashion that supports the achievement of the SA guidelines (2). A confirmation of the adaptation is submitted from PA to SA (3). As soon as a replanning becomes necessary, PA pulls the required planning data from the process execution system (4), derives a new process using the currently valid planning rules and delivers the process information to the execution system (5). The interactions (2) and (3) form the planner adaptation cycle and the interactions (4) and (5) are the process control cycle. Both cycles are concatenated by the feedback interaction (1). The planner adaptation cycle enables the adaptation of the process control as soon as the system performance requires an adaptation. The overall planning system (consisting of the two mentioned interacting cycles) is therefore able to update the processes in the process execution system autonomously without any intervention of human assistance even if the underlying problem changes significantly.

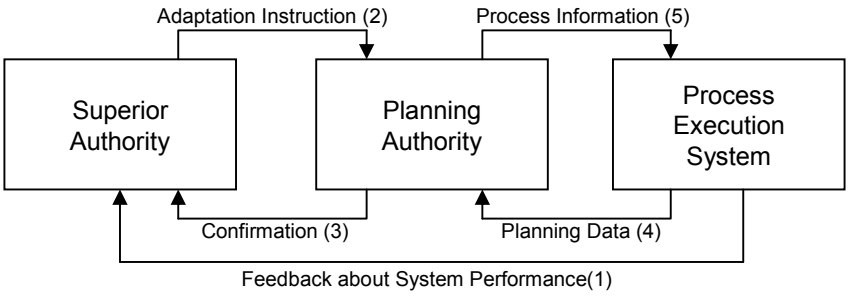

Fig. 2: Interaction of superior authority and planning authority

A generic process management algorithm is proposed in Fig. 3. It exploits an externally specified adaptation rule to adjust a generic decision model to the current performance. Such a generic decision model represents the underlying decision problem but does not exploit any feedback information about the current system performance or its environment.

Initially, the so far followed process *InitialSolution* is submitted together with the *AdaptationRule* to be followed (1). Then, the valid *CurrentSolution* is set (2). Now the procedure waits until the *CurrentSolution* is completed or additional requests collected in *R* become known (3). In the first case, the procedure terminates because nothing is to do anymore (4). In the latter case, it is check, whether R corrupts *CurrentSolution* (5). If this is not true then the procedure waits again otherwise it starts updating *CurrentSolution* (5). Therefore, the current time is fetched (6) and a generic decision model *dm* is derived that includes the additional requests (7). Next, the current *performance* of the logistic system is calculated (8). Now, the generic decision model is adapted with respect to the currently observed *performance* following the predicted *AdaptationRule* (9). The adapted decision model is solved and a new *CurrentSolution* is generated from the adapted decision model *dm* (10). This new solution is broadcasted to the process execution system (11). The update iteration is terminated (12) and the procedure waits again.

```
(1)  PROCEDURE process_management(InitialSolution,rule);
(2)  CurrentSolution := InitialSolution;
(3)  wait until (CurrentSolution is completed)
         or (ExternalEvents R are released);
(4)  if (CurrentSolution is completed) then
         goto (13);
(5)  if not (SolutionCorrupted(CurrentSolution,R)) then
         goto (12);
(6)  time:= GET_CURRENT_TIME();
(7)  dm:= GENERIC_MODEL(time,CurrentSolution,R);
(8)  performance:=SYSTEM_PERFORMANCE(time,CurrentSolution);
(9)  dm:=ADAPT_MODEL(dm,performance,rule);
(10) CurrentSolution := SOLVE_MODEL(dm);
(11) BROADCAST(CurrentSolution);
(12) goto (3)
(13) END PROCEDURE;
```

Fig. 3: Process management algorithm

This algorithmic describes the framework for the reactive management of a logistic process in response to external events that are detected over time.

In the VRPTWUD context, a generic decision model consists of a search space in which all feasible transportation plans are coded and the standard objective function C, that represents the costs associated with each single transportation plan (cf. 1.2.4).

To solve the adapted model, we apply the Memetic Algorithm framework introduced in Schönberger (2005). We desist of a detailed description of the algorithm components and configuration but refer to the previously given literature.

1.3.4 Adaptation Rules

An adaptation rule maps a constellation of key indicator values to an instruction that describes the modification of the current generic model in order to implement additional knowledge about the current process performance into the model.

In this contribution, we apply three different adaptation rules in the process management algorithm for solving the VRPTWUD. All four rules read the currently observed punctuality p_t and derive some model modification instructions from this value. The proposed modifications are applied immediately.

For purposes of comparison, we define the rule **NONE**, that do not apply any adaptation. Consequently, the generic decision model is solved in each iteration. No performance feedback is exploited.

The adaptation of the search direction is targeted in the experiment with the rule **SDAD** (**S**earch **D**irection **AD**aptation). The generic approach consists in the modification of the objective function of the generic decision model. In particular, it is aimed at adjusting the costs caused by a too late arrival at a customer location. The idea is to give less weight to a time window violation if the system load is very high so that time window violations cannot be prevented at all. Instead, decisions about subcontraction or self-fulfilment should be favoured. Furthermore, in case that the system load is low and time window violations can be prevented by subcontraction, such a time window violation should be penalized very hard. In order to realize the adaptation of the objective function, we replace the proposed cost function C in the memetic algorithm solver by an extended cost function that adjusts the weighting of the cost drivers to the current search state. The cost function C is replaced by

$$\widetilde{C}(s) := \left(\frac{e^{1+\chi(s,k)}}{e^{1}} \right)^{k} \cdot C(s)$$

where s denotes the current solution alternative to be evaluated, k refers to the search iteration of the applied Memetic Algorithm solver. The parameter $\chi(s,k)$ represents the fraction of the time-window-constraint-violations within s compared to all time window constraint violations observed in the k-th population generated by the Memetic Algorithm. $\widetilde{C}$ enlarges the costs of s compared to the other maintained solution proposals if s contains an above-average number of too-late-arrivals. On the other hand, it awards s if it comes along with a below-average number of too-late-arrivals at customer sites.

The second proposed rule aims at adjusting the constraint set of the given model with respect to the currently detected performance. **CSAD** (**C**onstraint **S**et **AD**aptation) shrinks decision variable domains of selected indicator variables determining the subcontraction of a request. More concretely, if the current punctuality falls below p^{target}, then the subcontraction

of a certain subset of requests is enforced by shrinking the set of possible values for the corresponding binary decision variables y_r from{0,1} down to {1}. If the least expected punctuality p^{target} is re-achieved, then 0 is added again to the domain of the affected y_r. A detailed description of the control of CSAD can be found in Schönberger and Kopfer (2007).

The third proposed adaptation rule **SDCS** (**S**earch **D**irection and **C**onstraint **S**et Adaptation) combines the features of the SDAD and CSAD rule. Both the search for appropriate solutions as well as the predetermination of subcontraction decisions is addressed for adapting the decision model.

1.4 Numerical Experiments

In this subsection, we report about the executed numerical experiments in which the proposed framework is assessed in combination with the proposed adaptation rules. The setup of the experiments is described in 1.4.1 and the achieved results are presented and discussed in 1.4.2.

1.4.1 Experimental Setup

We have simulated the four scenarios within three independent runs for each combination of rule and scenario and for each of the four adaptation rules NONE, SDAD, CSAD and SDCS. Overall, 3×4×4 = 48 experiments have been performed.

In each of the experiments, we have set the target punctuality p^{target} to 0.8.

The configuration of the CSAD rule is as follows. A first constraint set modification (intervention) is applied as soon as the currently observed punctuality falls below 0.85 percent and the maximal intervention intensity takes place as soon as p_t falls below 0.75. If the punctuality p_t falls below 0.85 the intensification of the application of the specified rule increases proportionally until it reaches the maximal possible intensity 1 for $p_t \leq 0.75$.

At the transportation plan update time t only requests contained in the interval [t-500;t+500] are considered for the calculation of p_t

A delay of less than 10 time units at a customer site does not cause any penalization costs. If a vehicle arrives more than 100 time units after the associated time window has been closed, a penalty amount of PENMAX=100 money units has to be paid for this out of time window arrival to the affected customer. With increasing delays larger than 10 time

units, the corresponding penalties increase proportionally up to the maximal penalty amount of 100 money units.

Since we want to demonstrate the ability of the model adaptation to overrule the short-term cost minimization objective, we enlarge the subcontraction costs by the prohibitive factor 20.

1.4.2 Results

We have recorded the observed punctualities within the moving time window specified above. The value p_t^{RULE} denotes the averagely observed punctuality at time t within the experiments where the adaptation is carried out according to RULE.

In a reference experiment with the application of rule NONE, p_t^{NONE} reduces by 60.4% after the demand peak has occurred (compared to p_{500}^{NONE}). Furthermore, it can be seen from Fig. 4 that in 89% of the observation interval, the observed punctuality lies below p^{target}. Immediately after the demand peak occurs at time t=1500, p_t^{NONE} falls below p^{target} and does not recover throughout the ongoing simulation experiment.

The logistic system performs better if the rule SDAD is applied in the process management procedure. In this case, a decrease by 21.9% of p_t^{SDAD} is observed after t=1500 and in only 21% of the length of the observation interval, the target punctuality is not reached. The duration for the recovery of p_t^{SDAD} is 1000 time units (from t=2100 where the target punctuality is not achieved for the first time until t=3100 when it is re-achieved). From the presented results, we state that the adaptation of the search direction boosts the system performance with respect to the current punctuality of the system.

A further increase in the system performance is observed for the application of the CSAD rule. Here, the maximal loss of p_t^{CSAD} after the demand peak is limited to 17% and in 20% of the observation interval, the least desired punctuality p^{target} is not achieved. Furthermore, the decrease of p_t^{CSAD} starts immediately at t=1700 units but the target punctuality has been reachieved after 900 time units at time t=2600. From this time on, the punctuality does not fall again below 80%.

The simultaneous adaptation of the search direction as well as of the constraint set as realized in the SDCS rule outperforms the other two adaptation rules and produces very convincing results. After the demand peak has been started, the punctuality p_t^{SDCS} does not reduce by more than 8% with

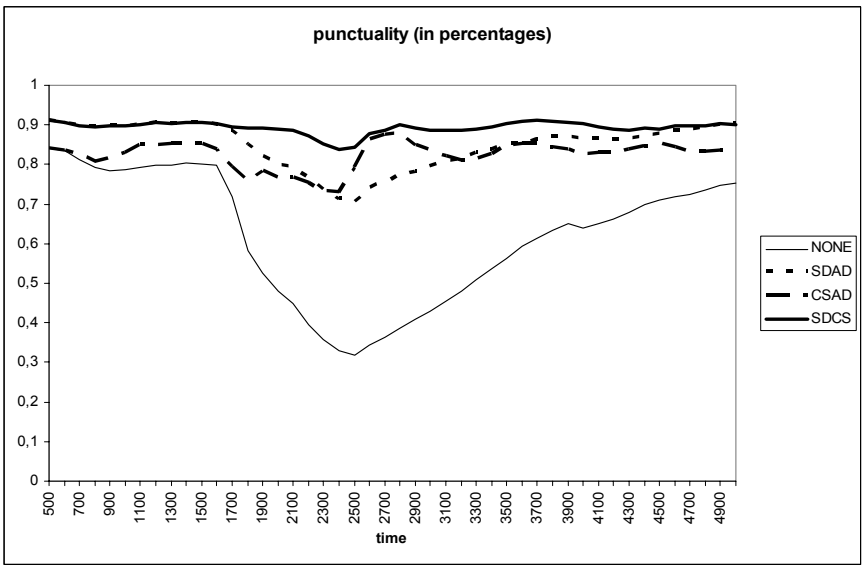

Fig. 4: Observed punctualities p_t^{RULE}

respect to p_{500}^{SDCS}. Throughout the overall observation period, the punctuality does not fall below 0.80.

To conclude the presentation of the observed results we state that the adaptation of the decision model to be solved after additional problem knowledge was known is necessary in a scenario with strongly varying system load. Instead of an overall quality reduction by 62,6% (comparing the maximal and the minimal observed punctuality values) in the reference experiment without any adaptation, the adjustment of the search direction results in a significant performance increase. The maximal reduction of p_t^{SDAD} is 22,4%. However, this value is further reduced down to 13,4% if the constraint set is adapted but the most convincing results are observed for the simultaneous adaptation of both the search direction and the constraint set (8% variation). Therefore, the main result we learn from this experiment is that the adaptation of the decision model is able to reduce the impacts of system load peaks and helps to keep the performance on a nearly unchanged level.

It is obvious that the stabilization of the punctuality is not achieved "free of charge" because the application of SDAD, CSAD or SDCS overrules the repeated cost minimization. For a couple of requests not the cheaper self service but the much more expensive subcontraction fulfilment mode has been selected to keep the punctuality on a sufficiently high level. In

order to compare the impacts of the application of the different adaptation rules, we have computed the relative increase cc_t^{RULE} in the cumulated costs observed up to a time t with respect to the cumulated costs CC_t^{NONE}. It is

$$cc_t^{RULE} := \frac{CC_t^{RULE}}{CC_t^{NONE}}, \text{ for RULE} = \text{SDAD}, \text{CSAD}, \text{SDCS},$$

where CC_t^{RULE} denotes the averagely observed cumulated costs up to time t observed in the experiment with RULE. The observed values for cc_t^{RULE} are summarized and presented in Fig. 5.

Closely after the demand peak has been occurred, the additional costs explode. However, for later observation times, the relative costs reduce. For SDAD and SDCS it seems to converge asymptotically towards the value 2. However, CSAD seems to produce costs that will be four time larger than in the NONE experiment but it should be stated that CSAD seems to be worried in early times (t<1500) producing an overreaction.

1.5 Conclusions

Within this contribution, we have introduced a generic framework for the reactive adaptation of logistic processes to unpredictable change in its environment. This framework has been tested successfully for artificial benchmark instances representing the VRPTWUD. Different adaptation rules have been assessed.

From the observed results we deduce the general applicability of the proposed framework as well as of the proposed adaptation rules for adjusting a generic optimisation problem to the currently observed system performance. However, the additional costs produced by a deviation from the pure cost minimization objective are significant larger.

The adaptation rule concept allows an autonomous self-adjustment of the two-cycle planning system to varying planning assumptions. Feedback from the process execution system (the real world) is exploited explicitly.

Future research will include the investigation of more complex adaptation rules. Furthermore, it should be investigated how the gap between the costs for the different available fulfilment modes (self-fulfilment and subcontraction) influences the applicability of the model adaptation.

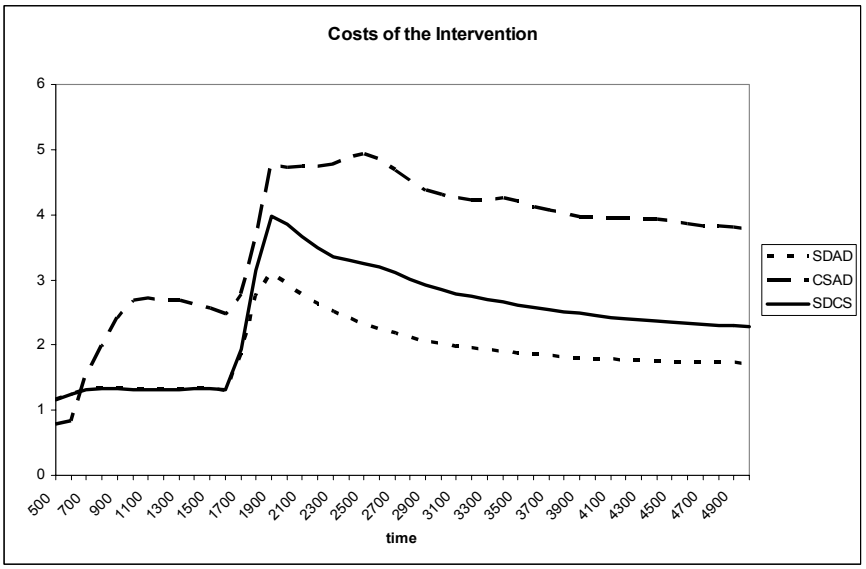

Fig. 5: Additional costs cc_t^{SDAD}, cc_t^{CSAD} and cc_t^{SDCS}

1.6 References

Ausiello G, Feuerstein E, Leonardi S, Stougie L, Talamo M (2001) Algorithms for the On-Line Travelling Salesman. Algorithmica 29: 560-581

Bertsimas DJ, van Ryzin G (1989) The Dynamic Traveling Repairman Problem. MIT Sloan School Working Paper No. 3036-89-MS

Bianchi L, Birattari M, Chiarandini M, Manfrin M, Mastrolilli M, Paquete L, Rossi-Doria O, Schiavinotto T (2005) Hybrid Metaheuristics for the Vehicle Routing Problem with Stochastic Demands. Technical Report No. IDSIA-0605.

Bramel J, Simchi-Levi D (1997) The Logic of Logistics. Springer, New York

Brotcorne L, Laporte G, Semet F (2003) Ambulance location and relocation models. European Journal of Operational Research 147: 451-463.

Chen Z-L, Xu H (2006) Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. Transportation Science 40(1): 74-88

Clifton C, Iyer A, Uzsoy R (2002) A Prototype Integrated Transaction Data Analysis and Visualization Environment for the Transportation, Distribution and Logistics Sector. Purdue University, Indiana, USA

Crainic T, Laporte G (1997) Planning models for freight transportation. European Journal of Operational Research 97: 409-438

Crainic T, Laporte G (1998) Fleet Management and Logistics. Kluwer

Fiat A, Woeginger G J (1998) Online Algorithms – The State of the Art. Springer, Berlin/Heidelberg

Fleischmann B, Gnutzmann S, Sandvoß E (2004) Dynamic Vehicle Routing Based on Online Traffic Information. Transportation Science 38(4): 420-433

Gayialis S, Tatsiopoulos I (2004) Design of an IT-driven decision support system for vehicle routing and scheduling. European Journal of Operational Research 152: 382-298.

Gendreau M, Guertin F, Potvin J-Y, Taillard E (1999) Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science 33 (4): 381-390.

Gendreau M; Potvin J-Y (1998) Dynamic Vehicle Routing and Dispatching. In: Crainic T G; Laporte G.: Fleet Management and Logistics: 115-126

Ghiani G, Guerriero F, Laporte G, Musmanno R (2003) Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. European Journal of Operational Research 151(1):1-11

Gutenschwager K, Niklaus C, Voß S (2004) Dispatching of an Electric Monorail System: Applying Metaheuristics to an Online Pickup and Delivery Problem. Transportation Science 38(4): 434-446

Hiller B, Krumke S O, Rambau J (2005) Reoptimisation Gaps versus Model Erros in Online-Dispatching of Service Units for ADAC. Submitted for publication in Discrete Applied Mathematics

Holzer M (2003) Hierarchical Speed-up Techniques for Shortest-Path Algorithms. Diploma Thesis, University of Konstanz, Germany

Ibaraki T, Nonobe K, Yagiura M (2005) Metaheuristics: Progress as Real Problem Solvers. Springer, Heidelberg

Irani, S, Lu, X, Regan, A (2004) On-line Algorithms for the dynamic traveling repairman problem. Journal of Scheduling 7: 243-258

Jaillet P (1988) A Priori Solution of a Traveling Salesman Problem in Which a Random Subset of the Customers Are Visited, Operations Research, 36(6): 929-936

Jaillet P, Odoni A R (1988) The Probabilistic Vehicle Routing Problem. In: Golden B L, Assad A A (Eds.) Vehicle Routing: Methods and Studies, North-Holland: 293-318

Jensen M T (2001) Robust and Flexible Scheduling with Evolutionary Computation. PhD Dissertation, University of Aarhus, Denmark

Kopfer H, Schönberger J (2006) Adaptive Optimierung: Selbststeuernde Anpassung einer operativen Transportkostenoptimierung an strategische Qualitätsziele. In: Jacquemin M, Pipernik R, Sucky E (Eds.) Quantitative Methoden der Logistik und des Supply Chain Management, Verlag Dr. Kovac: 321-337

Krumke, S (2001) Online-Optimisation – Competitive Analysis and Beyond. Habilitation Thesis, Technical University Berlin, Germany.

Lackner, A (2004) Dynamische Tourenplanung mit ausgewählten Metaheuristiken. Cuvilier Verlag Göttingen.

Makowski, M. (1994) Design and Implementation of Model-based Decision Support Systems. Working Paper WP-94-86, IIASA

Michalewicz Z, Fogel D B (2004) How to Solve It: Modern Heuristics. 2nd Edition. Springer, Heidelberg

Mitrović-Minić S, Krishnamurti R, Laporte G (2004) Double-horizon based heuristic for the dynamic pickup and delivery problem with time windows. Transportation Research B 38: 635-655

Pankratz G (2002) Speditionalle Transportdisposition. DUV.

Psaraftis H (1988) Dynamic vehicle routing problems. In: Golden B L, Assad A A (Eds.) Vehicle Routing: Methods and Studies, North-Holland: 223-248.

Psaraftis H (1995) Dynamic vehicle routing: status and prospects. Annals of Operations Research 61: 143-164.

Sandvoss E (2004) Dynamische Tourenplanung auf Basis von Online-Verkehrs-Informationen. ProBusiness.

Savelsbergh M, Sol M (1998) DRIVE: Dynamic Routing of Independent Vehicles. Operations Research 46:474-490

Schönberger J, Kopfer H (2007) On Decision Model Adaptation in Online Optimisation of a Transport System. Accepted for publication in the proceedings of the conference Supply Networks and Logistics Management – Decision Support, Information Systems and OR Tools

Schönberger J (2005) Operational Freight Carrier Planning. Springer.

Séguin R, Potvin J-Y, Gendreau M, Crainic T G, Marcotte P (1997) Real-time decision problems: an operational research perspective. Journal of the Operational Research Society 48(2):162-174

Slater A (2002) Specification for a dynamic vehicle routing and scheduling system. International Journal of Transport Management 1: 29-40.

Solomon M (1987) The vehicle routing and scheduling problems with time window constraints. Operations Research 35 (2): 254-265.

Williams H P (1999) Model Building in Mathematical Programming. 4th Edition. Wiley, Chichester