7 Real-time control (RC)

- In what follows, we consider main attributes of real-time control of complex processes
- Complex processes
  - Managing a freight forwarder's transportation network
  - Material handling in a production system
  - Variant sequencing within a production process that is executed on a mixed model assembly line
  - Real-time scheduling of a Job Shop system
  - Cargo handling in a transshipment hub

Observations

- Complex processes cannot be handled efficiently in an intuitive way
- They have to be planned and controlled in order to attain competitive results
- Note that an efficient implementation of complex processes in industry may frequently be crucial to the firms' survival

7.1 Real-time control (RC) – Introduction

- *Result of planning is the plan*
- It defines on an application-dependent accuracy level
  - the measures to be taken
  - the chronology of these measures
  - the impact on total system's effectiveness
  - the made assumptions
  - the requirements presupposed for its execution

Combinatorial Optimization

- Planning is done by using combinatorial optimization techniques and problem descriptions
- A real-world problem is transformed into an optimization problem with
  - Parameters
  - Variables
  - Objective function
  - Restrictions

The problem can be defined by:

Minimize $c^t \cdot x$, s.t., $A \cdot x = b$

Start of process

- What we are going to do at the point of time when the process starts, i.e., at the point in time when the planned process commence its execution?
- We simply launch the execution of our calculated plan
- Now,
  - anticipation is over
  - and the “reality rules”

Disturbances…

- can be characterized as each significant change from the presupposed data
- can occur throughout the plan execution
- can endanger an efficient or even a feasible execution of an existing plan
- enforce necessary plan adaptations executed simultaneously to the plan execution

Disturbances in industrial processes

- Delay of material supplies
- Defective preliminary products
- Machine breakdowns
- Occurring traffic jams
- Worker drop out
- Driver drop out
- Decelerations of vehicles for technical reasons

➢ All apart from dynamically incoming requests (no disturbances)

Disturbances in complex everyday processes

- Car drop out
- Delay of trains
- Missing the connection train
- Hotel is overbooked
- Traffic jam
- Accident
- Blockage of a street

Real-time control

- **Control level**

  =comprises all activities necessary for an efficient feasible execution of the respective process, i.e., all tasks of the planning level are passed to the control level to ensure a feasible and, if possible, efficient execution of the process

  - receives the output of the planning level as an initial process plan
  - has to handle occurring disturbances
  - has to handle dynamic problem constellations
  - has to implement necessary plan adaptations simultaneously to the plan execution

Disturbance scenarios

- A real-time control has to estimate how an occurred disturbance affects the plan currently in execution
- Therefore, possible consequences have to be revealed and rated
- Countermeasures have to be designed so that the control can use it appropriately under the hard time restriction of a real-time control

7.1.1 Basic definitions

- In this section, we introduce several notations
- All of them deal with the definition of a specific design structure of real-time oriented control approaches
- Consequently, this section addresses **meta concepts**, i.e., basic concepts used for implementing an application-dependent real-time approach

Basic model

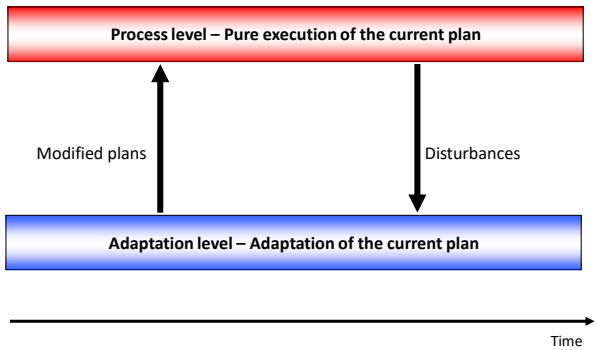
When we think about real-time control in this lecture, we have a specific constellation in mind

- A plan is present
- This plan is feasible and already in execution
- This plan guidelines the execution of the process in question
- However, this execution is affected by disturbances coming from reality
- Therefore, the control has to adapt the plan

Control

- Execution of the plan
- Disturbance handling
  - How to handle?
    - Storing or instant processing?
    - Predefined rules or resolving the model?
- Implementation of adaptations regarded to be necessary
  - Correction of plan decisions already taken
  - i.e., resolving the combinatorial model

Process level – Adaptation level



Disturbance handling

- Disturbances may occur throughout the planning horizon
- Obviously, they are unpredictable
- When should we react on them?
  - Instantly?
  - After a predefined period?
  - After a specific event triggers a reaction?
  - …?

Instant Reaction

- Each incoming disturbance is handled immediately
- Such an event triggers the execution of a specific procedure adapting the plan in execution according to the requirements of the disturbance
- Effects caused by the disturbance are directly transformed into plan corrections without any delay

Instant Reaction

+ No loss of decision range
+ Best policy for an appropriate reaction
+ Keeps all possibilities
+ No roll back is necessary
+ Adequate for all scenarios

However:

- Control can be reduced to disturbance handling
- Error prone processes cannot be controlled in this way

Triggered by events

- Reaction on occurring disturbances is triggered by predefined events apart from the disturbance occurrence or the elapse of a time limit
- For instance,
  - buffers are full
  - occurred disturbances are too urgent to be delayed until the checking

Triggered by events

+ Flexible approach
- Extremely expensive roll backs can become necessary
- Reactive decision range is reduced almost unpredictable
- Combinations with other concepts, which guarantee a reaction in reasonable time (e.g., by time limits), become necessary in order to implement a “real” real-time control

Adaptation handling

- Due to the simultaneity of the plan adaptation and execution, some **specific coordination problems** arise
- Therefore, the adaptation-handling of an applied control-approach defines the basic structure of the entire concept
- Thus, besides the **applied solution approaches** and **problem model,** the efficiency of a real-time control approach depends on its implemented adaptation handling

Adaptation handling defines…

- …when an adaptation of the plan is executed (**adaptation frequency**)
- …how the process is continued during its modification (**adaptation synchrony**)
- …how the adaptation is executed (**adaptation technique**)
- …the size of the decision range of the adaptation procedure (**adaptation range**)
- …the duration of the adaptation processes (**adaptation duration**)

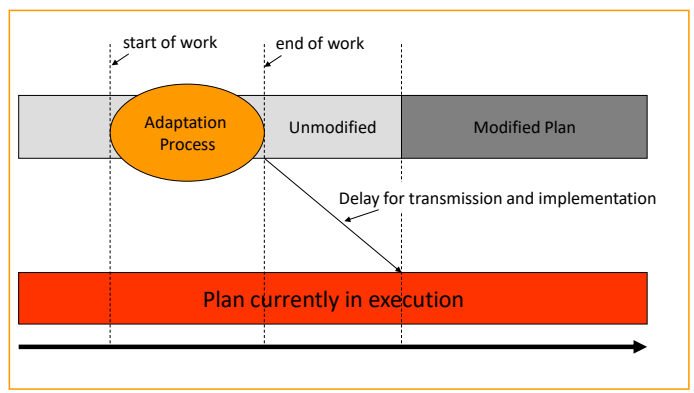
Adaptation frequency

- Defines the scenarios **when an adaptation of the current plan is initiated**, e.g.,
  - *Disturbance-oriented concepts*: Those approaches start a modification only after the occurrence of a disturbance
  - *Event-oriented concepts*: Here, the occurring of specific predefined events results in the execution of an adaptation process
  - *Periodic concepts*: Those concepts adapt the current plan in predefined time intervals
  - *Continuous concepts*: Here, the current plan is always adapted according to all decisions which have not been taken

Adaptation synchrony

- Decides about the priority handling between plan modifications and plan realization
  - *Planning priority*: Execution of the controlled process is stopped during the computational re-planning of the current plan. **Only applicable for correction processes working in zero time**. No simultaneity of plan adaptation and execution
  - *Process priority*: This policy fixes major subsequent parts of the plan currently in execution independent from the fact if it is still possible to modify them. Frequently proposed for controlling approaches working with a rolling planning horizon and a periodic adaptation strategy
  - *Simultaneity*: In this **ideal constellation**, the only decisions not changeable by the adaptation process are solely those which are currently transferred to the controlled process

Implementation delay



Adaptation technique

- By choosing the applied adaptation technique, the proceeding for modifying the currently executed plan is defined
  - Priority rules: These are so-called greedy approaches that are frequently assumed to be able to work in zero time
  - Improvement procedures: By the execution of specific moves, the currently stored plan is adapted
    - Pure improvement approaches (hill climber)
    - Metaheuristics

Adaptation range

- Starting from the **used problem model**, the adaptation range defines **which part of the existing variables can be modified** during the adaptation process, e.g.,
  - *Complete range*: No reduction
  - *Usable range*: Reduction by the time period elapsing for information transfer and execution
  - *Time-limit depending range*: Reduction by the time limit
  - *Selected range*: Predefined subsets of variables are excluded, e.g., next hour, specific started processes

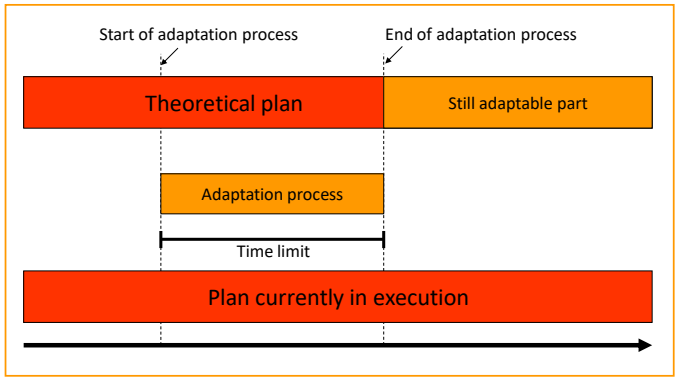
Simple observation

- The more decisions can be changed, the more appropriate the control can react
- The **variable set of the used problem model** determines an upper bound for the decision range
- Additional reductions result from the implemented adaptation handling, i.e., which remaining free variables are additionally fixed for the adaptation of the plan currently in execution?

Adaptation duration

- Time available for each executed adaptation process
  - Time limit based: Duration is predefined
  - Zero time: Maximal time duration that still can be interpreted as zero time
  - Infinite time: Continuously applied adaptation processes

Illustration – Time limits



Time limits

- The duration is predefined by a fixed duration of t time units
- The problem to be examined is the constellation occurring in future after the elapse of t time units, i.e., here the modified plan commences its execution
- An interesting tradeoff occurs:
  - The longer the time limit is, the more adequate are the calculated solutions of each considered static problem
  - The shorter the time limit is, the less the available decision is reduced by the implemented adaptation handling
  - What is an adequate compromise?

Finding efficient time limits

- Self-adapting approaches
  - Depending on the elapsed computations, the duration of the time limits are adjusted iteratively
  - For instance, the more disturbances occur, the shorter the time limit is defined
  - Why? How does the disturbance rate effect an efficient choice of the time limit?

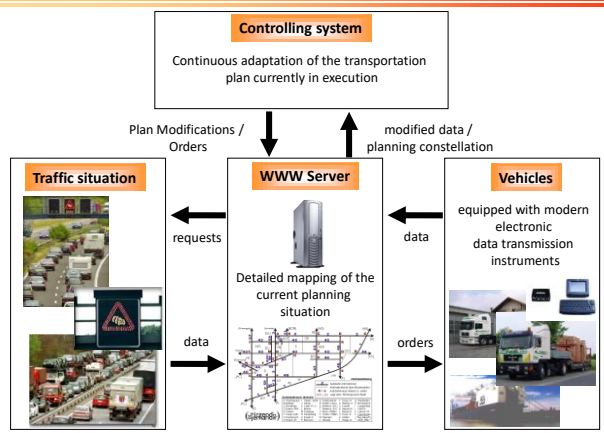
Zero Time

- What is zero time? Really zero?
- No, of course not
- However, an application dependent lower bound can be defined up to which a consumed computational time is negligible
- This application-dependent duration is denoted as zero time
- Frequently, simple heuristics basing on priority rules are used to ensure this short response time
- Note that “zero time handling” is simple since there is no handling necessary at all

7.1.2 Real-time control in transportation

- In what follows, we consider real-time control of transportation processes
- I.e., approaches are generated that are able to adapt a currently executed transportation process (i.e., the currently executed transportation plan) according to the requirements of the current situation in the transportation network
- First of all, however, we must state that transportation processes are widely distributed
  - Thus, information necessary to know for plan adaptations have to be transferred over large distances in zero time
  - Other way round, plan adaptations caused by the real-time control have to be transferred instantly to the drivers
- This becomes manageable by making use of the following concept

Information flow



Real-time control of VRPs

- In what follows, we consider real-time control of transportation processes based on the Vehicle Routing Problem (VRP)
- In literature, there is fast growing number of recent approaches dealing with real-time control issues
- In this section we exemplarily introduce three different real-time oriented VRP approaches. These are
  - Parallel Tabu Search for real-time vehicle routing and dispatching (Subsection 7.2)
  - Diversion issues in real-time vehicle dispatching (Subsection 7.3)
  - Exploiting past request data in urgent real-time vehicle routing processes (Subsection 7.4)

Basic problem

- As already mentioned, the **Vehicle Routing Problem with Soft Time windows** is considered in both approaches
- In addition, both approaches comprise the **application of the parallel Tabu Search procedure** described in Section 6.4.2.3. All **adaptations of the Transportation plan** currently in execution are **derived from the variable set of the VRP**
- As one important consequence, restricting the potential measurements of the controlling process, a **request already picked up by a vehicle cannot be transshipped** to be delivered by a different vehicle
- Both approaches can be mainly distinguished from each other in terms of **handling diversions of currently executed transports**. While the first approach excludes this possibility completely, the second one integrates these additional operations to improve the adaptability of the control process

7.2 RC of VRP's without diversion issues

- This approach is a straightforward adaptation of the approach considered in Section 6.4.2.2 and 6.4.2.3 for real-time constellations
- In contrast to the constellation considered there, we concentrate on a dynamic problem while in detail the following assumptions are made (see the following slide)

Assumptions of the approach

- Requests must be received **before a fixed time deadline** to be serviced the same day
- Those that are received after the deadline, however, may be kept for the next day
- Thus, the considered planning day starts with a number of pending requests for which an initial plan is already generated
- There is only **one single source of uncertainty, namely the occurrence of new requests**
- In particular, there is no uncertainty associated with the service locations, like cancellations or vehicle or traffic network breakdowns

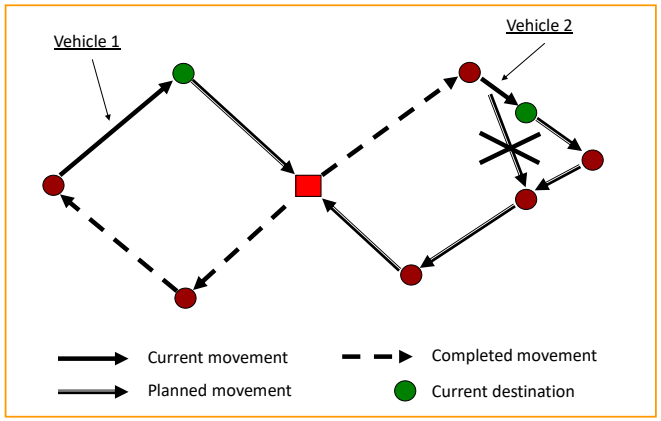
Assumptions of the approach

- Communication between the central dispatch office and the drivers is only performed **at each service location** and always determines the **next destination** for the subsequent transportation activity
- Note that this assumption significantly reduces the adaptation possibilities of the dispatch office
- If waiting time is expected at the next location, drivers are asked to wait at their current position
- Note that this **commitment enlarges adaptation possibilities** but **increases also the damage caused by failures** during the execution of the transportation processes

Diversion issues

- In contrast to the following approach that will be presented in the subsequent Section 7.3, diversion is not allowed
- As a consequence, **each transport already in execution cannot be diverted** at all
- Therefore, a modification of the tours is only possible at the end of the current transport activities, i.e., after the subsequent service activity of the respective vehicle

Illustration



Current solution

- The controlling process becomes an **iterative consideration of different static problems** which are stepwise derived from each other
- These problems are defined as **open constellations**, i.e., starts and ends of the tours are no longer forced to be identical
- Specifically, tours **start always from the current position** of the respective vehicle and comprise as its **final step the return to the depot**
- The **objective is – as already known from the static case –** the minimization of the weighted sum of total distance traveled and lateness at customer locations

Adaptations of the static procedure

1. While no event occurs, continuously optimize the planned routes using the Tabu Search procedure in background
2. If an event occurs, then
   1. stop each Tabu Search thread and add the routes of their best solution to the adaptive memory
   2. If the event is the occurrence of a new request, then
      1. update the adaptive memory through the insertion of the new request in each solution.
      2. If no feasible insertion position is found, reject the request;
   3. otherwise (end of service at a customer location)
      1. Identify the driver's next destination by using the best solution currently stored in the adaptive memory
      2. Update the other solutions accordingly
   4. Restart the Tabu Search processes with new solutions that are obtained from the adaptive memory

Continuous optimization

- Between two events the Tabu Search threads work in the background in order to further improve the current transportation plan
- The search process is always interrupted whenever an input update occurs
- No input buffer is applied
- Problem: Frequently occurring input modifications can prevent the execution of the necessary optimization phases

Event handling

- End of service
  - If a tour has ended its current service
  - The respective driver has to know his next destination
  - The best solution of the current adaptive memory is taken
  - Note that this constellation already starts from the position the vehicle is currently located
  - The remaining solutions in memory are updated by removing this customer from all tours and updating the new starting positions
- New request
  - Is inserted in all solutions stored in memory
  - The insertion position is chosen that minimizes the additional cost of the currently considered solution
  - If there is no valid insertion position in all tours, the request is rejected
  - Thus, the customer is told almost immediately that his request cannot be handled at all

Computational results

- Again, the algorithm is applied to different problem instances
- Used computer system is a network of 17 SUN UltraSparc workstations (143 MHz)
- To test the procedure in a dynamical environment for which it was developed, a discrete-time simulator was determined
- It uses again data taken from Solomon's 100-customer Euclidean problems to produce new occurring requests
- The time horizon for the simulation is adjusted to create two different types of scenarios
  - First scenario:

    Approximately **three requests per minute** on average
  - Second scenario:

    Approximately **one request per minute** on average

Request handling

- The set of requests is divided into two subsets (each one with half of the requests)
- **First subset** contains requests that are **known at the start of the day**, like the set received after the predefined threshold the day before
- In order to include requests in the first set, jobs with early time windows are preferred
- The initial routes are determined by the Tabu Search procedure applied to the static initial problems

Request handling

- **The second group** contains requests that are **received in real-time**
- To do so, the occurrence time of each request is determined in advance
- For request *i*, this value is randomly generated in the interval $[0,\hat{e}_i]$ with

$$\hat{e}_i = \frac{T}{l_0 - e_0} \cdot \min\{e_i, t_{i-1}\}$$

t_{i-1}: Departure time of *i*'s predecessor in the best known static solution

T: Time horizon of the simulation

Parameter setting

- **Initialization**
  - Number of initial solutions I=max{4,P}, where P is the number of Tabu Search processes
  - T=15 minutes for scenarios of type 1 and T=60 minutes for scenarios of type 2
- **Adaptive memory**
  Size: M=30 solutions
- **Decomposition/Reconstruction**
  Minimum number of cycles C=6
- **Tabu Search**
  Number of iterations: A·[1+(DR-1)/B], with:
  DR: Current decomposition/reconstruction 1..C
  Scenario 1: A=30, B=3
  Scenario 2: A=60, B=6
- **Neighborhood**
  Maximum length of route segment L=6
- **Objective function**
  α_i=1

Tested approaches

- Insertion (Ins)
  Inserts a new customer at the location that minimizes the additional costs over the current set of planned routes
- Insertion+ (Ins+)
  Applies additionally a local search procedure based on CROSS exchanges to improve the planned routes
- Rebuild (Reb)
  Reconstructs the planned routes each time a new service request occurs, using an adaptation of Solomons I1 insertion heuristic
- Rebuild+ (Reb+)
  Applies additionally the heuristic local search procedure which bases on the CROSS operation

Note that all these approaches only work on a single solution

Tested approaches

- Adaptive descent method (Ad Des)
  - Slight modification of the proposed Parallel Tabu Search approach
  - More precisely, the Tabu Search processes are stopped instantly if a local optimum is yielded which is recognized by the fact that no improvement is possible
  - Consequently, the process can be characterized as a multi-start local search process working on different starting solutions taken from the adaptive memory

Tested approaches

- Adaptive Tabu Search (Ad Tabu)

  This is the Parallel Tabu Search procedure described above

Scenario 1

- Three requests per minute on average
- I.e., more dynamism

	Ins	Ins+	Reb	Reb+	Ad Des	Ad Tabu
Average distance	1349,9	1080,8	1157,3	1124,5	1050,1	1039,2
Average lateness	485,2	55,2	171,6	124,8	45,1	30,7
Number of non serviced customers	2,05	0,57	0,57	0,46	0,25	0,09

Scenario 2

- One request per minute on average
- I.e., less dynamism

	Ins	Ins+	Reb	Reb+	Ad Des	Ad Tabu
Average distance	1349,9	1080,8	1150,8	1117,4	1038,6	1031,6
Average lateness	485,2	55,2	111,4	95,4	23,8	21,6
Number of non serviced customers	2,05	0,57	0,54	0,46	0,21	0,05

Main results – Comparison of heuristics

- Adaptive Tabu procedure and adaptive descent method work most efficiently
- Especially for scenario two where longer improvement phases are possible, these procedures yielded best results
- These approaches can serve more customers by an appropriate adaptation of the current plan
- There is little or no difference between scenarios of type 1 and 2 for the more simple approaches, i.e., they do not use the additional time in scenario 2 efficiently

Static vs. Dynamic

	Dynamic		Static
Results	**Scenario 1**	**Scenario 2**	
Average distance	1039,2	1031,6	1027,2
Average lateness	30,7	21,4	0,0
Number of non serviced customers	0,09	0,05	0,00

Parallel impact

Results	Number of used processors				
	1	2	4	8	16
Average distance	1058,0	1058,0	1050,5	1037,4	1039,2
Average lateness	34,8	25,9	28,0	30,5	30,7
Number of non serviced customers	0,29	0,27	0,23	0,11	0,09

Placement in the classification

- Adaptation frequency:
  Continuous
- Adaptation synchrony
  Process priority / Simultaneity
- Adaptation technique
  Tabu Search
- Adaptation range
  VRP variables reduced by decisions for the current tour
- Adaptation duration
  Continuous adaptation processes

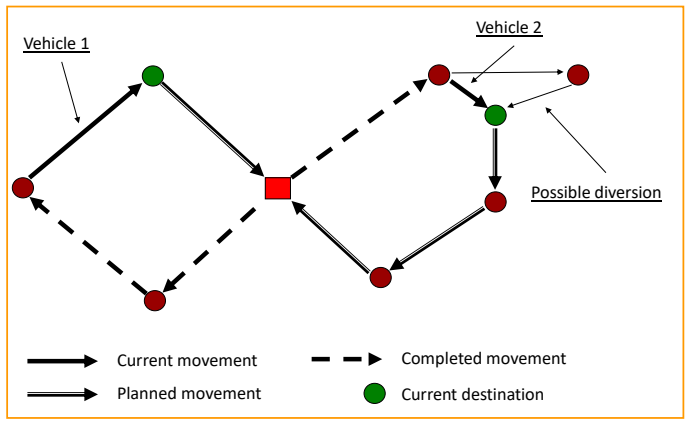
7.3 RC of VRP's with diversions

- Now, the set of potential adaptations is enlarged by the possibility of diverting a transportation process that is currently in execution
- This possibility is additionally integrated into the approach that was described before
- By doing so, however, we are facing the additional problem of simultaneous optimization and process execution during the currently executed transport activities

Time limit based extension

- The whole time horizon starting from the start of the transport processes is separated in intervals of length tδ
- In order to feasibly start optimization processes, we have to simulate this predefined interval in advance to freeze all decisions in this interval
- Therefore, the applied improvement procedure can work on a static problem and may change the subsequent decisions only
- As a consequence, we face a tradeoff between single solution quality and size of adaptability

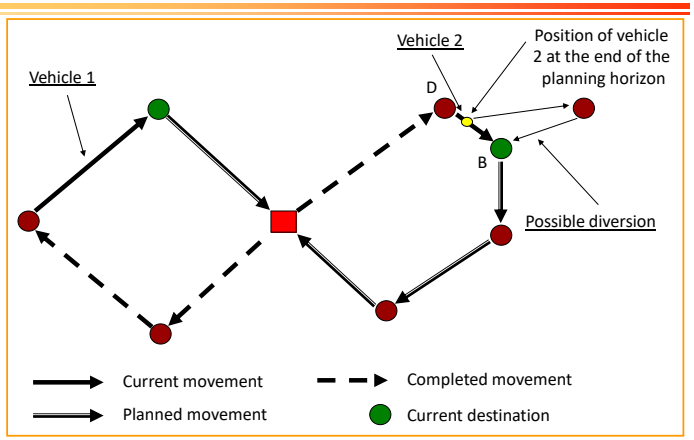
Illustration



Time limit integration

- Let us assume a new request is received at point of time t while a vehicle is on its way from D just serviced to destination B
- Then, we have to determine the next position that is reached as a starting point of some possible diversion activities to adapt the current plan
- That means we have to generate a plan corresponding to the situation occurring in next future after ending the optimization procedure, i.e., we cannot neither use D, nor B as the position of the vehicle
- Therefore, we use a predefined time interval and generate all positions at the end of this short intermediate planning horizon
- Consequently, the objective is to provide **decisions which will fit to the context found at the end of the time allocated to the optimization procedure**

Illustration



Time limit – The resulting tradeoff

- The larger the size of the time limit is, the more computation time is available in each improvement iteration
- I.e., the better becomes each solution to the respective static problem
- The smaller the size of the time limit is, the more the size of the set of adaptability is enlarged wherefore the controlling process can change the given solution more comprehensively, i.e., particularly diversion possibilities are won

Time projection

- When a new request is received at time t, all constellations in the memory are updated **according to the state of the system at time t+tδ**, while the subsequently applied optimization procedure gets the computational time horizon tδ to improve the current plan
- To do so, all routes get the new starting position which is reached by the respective vehicle at t+tδ
- This is done for all existing solutions in memory

Illustration – Before updating

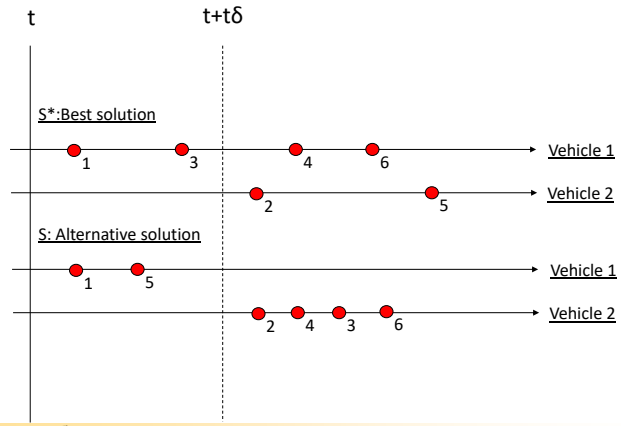
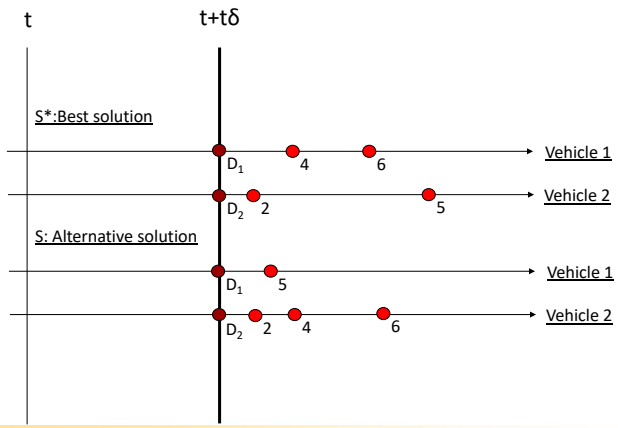


Illustration – After updating



Event handling

- Event occurs at time t
  - All events in interval [t, t+tδ] will not effect the search threads, but handled as follows:
    - When a vehicle has finished servicing a customer, its next destination is determined by the best constellation found in adaptive memory, i.e., we follow the best solution currently known during this frozen interval
    - When a new request occurs, it is inserted in all solutions, which are stored as a copy of the adaptive memory. If this is possible for at least one solution, the request is accepted, otherwise rejected (it is checked (after insertion) whether the end time of the respective route exceeds the deadline at the depot). Solutions with no feasible insertion positions are discarded
    - Note that the copy is always updated in this period if a new constellation was found by the Tabu Search process and inserted in the adaptive memory

Event handling

- At time t+tδ, all search threads are stopped and the adaptive memory is updated with the best solution found. Finally, all pending requests are handled one by one as known from the original algorithm that was considered in Subsection 7.2
- Subsequently, the continuous optimization process described in Subsection 7.2 is restarted until a new event occurs

Setting the time limit – Rule 1

- Here, the planning horizon $t\delta$ is set to the smallest time distance to a next event in some transportation tour
- Unfortunately, this rule frequently leads to very small sized intervals since the approach is applied for local traffic constellations
- Consequently, this rule was **not applied!**

Setting the time limit – Rule 2

- $t\delta$ is chosen according to the parameter D that measures the average distance between two neighboring requests during the last l requests
- For this purpose, $t\delta$ is set to $\alpha_1 \cdot D$, while α_1 is an empirically chosen parameter
- l is set by defining $l=\beta_1 \cdot K$, with K as the number of jobs known in advance and, again, β_1 is a predefined parameter
- Effect:

  If the distance D is small, a large horizon is counterproductive, and, therefore, $t\delta$ is decreased

Setting the time limit – Rule 3

- In rule 2, all requests are equally weighted, i.e., also requests that still have a long waiting time between requesting and processing have an identical influence on the result as well as very urgent requests have
- In order to modify this, only jobs are considered whose processing is part of a **predefined time interval *X***, while l_X gives their total number
- Then, we define
  - $t\delta=\alpha_2 X/l_X$ and
  - $X=\beta_2 L$, with L as the length of the time horizon for the simulation

Computational results

- Again, both scenarios described in section 7.1 are tested on a network of 9 SUN UltraSparc-II Workstations (300 MHz)
- Each process was programmed in C++ and communication between the processes was handled by the Parallel Virtual Machine Software (PVM library)
- Fleet size was determined by the number of routes in the best solution reported in literature so far
- First of all, tests were executed to find out efficient parameter constellations for the rules 2 and 3
- Subsequently, a comparison between these optimized procedures and the original procedure is made in order to analyze the impact of diversion operations
- Two scenarios are simulated
  - Scenario 1: Lower dynamism (50 % of requests are static)
  - Scenario 2: Higher dynamism (25 % of requests are static)

Computational results – Rule 2

TABLE II
Searching for the Best Parameter Values under Rule 2

	Scenario 1					Scenario 2				
Problem Set	$\alpha_1 = 0.90$ $\beta_1 = 0.15$	$\alpha_1 = 0.90$ $\beta_1 = 0.25$	$\alpha_1 = 1.20$ $\beta_1 = 0.25$	$\alpha_1 = 0.50$ $\beta_1 = 0.10$	$\alpha_1 = 0.50$ $\beta_1 = 0.15$	$\alpha_1 = 0.50$ $\beta_1 = 0.10$	$\alpha_1 = 0.50$ $\beta_1 = 0.15$	$\alpha_1 = 1.20$ $\beta_1 = 0.15$	$\alpha_1 = 0.90$ $\beta_1 = 0.25$	$\alpha_1 = 0.25$ $\beta_1 = 0.25$
C_1 4 problems	0[a]	0	0	0	0	0	0	0	0	0
	829.89[b]	832.35	828.535	852.31	828.94	882.42	871.055	908.478	925.06	881.658
	0[c]	0	0	0.65	0	0.105	0	19.79	15.027	2.127
	829.89[d]	832.35	828.535	852.96	828.94	882.525	871.055	928.268	940.088	883.785
R_1 4 problems	0.25	0	0.25	0.25	0	0.25	0	0.25	0	0.50
	1253.02	1284.25	1262.32	1266.03	1281.79	1296.70	1284.24	1303.97	1303.56	1296.28
	63.435	38.307	43.90	47.12	45.385	76.00	63.982	73.315	67.71	53.067
	1316.46	1322.56	1306.22	1313.15	1327.18	1372.70	1348.22	1377.28	1371.28	1349.35
RC_1 4 problems	0.5	0.75	0.5	0.75	0.25	0.25	0.75	0.5	1.00	1.00
	1334.86	1324.50	1310.76	1318.16	1330.19	1366.52	1349.24	1405.42	1402.88	1354.10
	74.472	62.985	55.962	67.762	83.022	54.99	80.737	77.917	61.32	64.82
	1409.33	1387.49	1366.72	1385.92	1413.22	1421.50	1429.98	1483.34	1464.20	1418.92
C_2 4 problems	0	0	0	0	0	0	0.25	0	0	0
	642.978	625.628	612.602	616.238	627.51	607.572	619.515	613.62	621.495	612.33
	0	0	0	0	0	0	0	0.002	0	0
	642.978	625.628	612.602	616.238	627.51	607.572	619.515	613.62	621.495	612.33
R_2 4 problems	0.5	0.25	0	0.75	0	0	0.75	0	0.25	0
	1096.48	1106.52	1106.13	1126.82	1107.19	1113.66	1093.96	1119.40	1140.72	1106.98
	24.182	30.155	19.337	53.782	14.645	17.63	93.605	23.73	14.032	22.36
	1120.66	1136.67	1125.47	1180.60	1121.84	1131.29	1187.56	1143.13	1154.76	1129.34
RC_2 4 problems	0	0	0	0	0	0	0	0	0	0
	1226.10	1184.14	1188.22	1198.09	1180.53	1207.95	1235.45	1213.50	1144.22	1218.88
	24.045	38.74	10.745	45.41	17.482	37.50	19.827	35.867	60.822	37.622
	1250.14	1222.88	1198.96	1243.50	1198.01	1245.45	1255.28	1249.37	1205.04	1256.50

[a]Number of unserved customers.
[b]Total distance traveled.
[c]Total lateness.
[d]Objective value.

Computational results – Rule 3

TABLE III
Searching for the Best Parameters Values Under Rule 3

	Scenario 1					Scenario 2				
Problem Set	$\alpha_2 = 0.50$ $\beta_2 = 0.15$	$\alpha_2 = 0.90$ $\beta_2 = 0.15$	$\alpha_2 = 1.20$ $\beta_2 = 0.15$	$\alpha_2 = 0.50$ $\beta_2 = 0.25$	$\beta_2 = 0.25$ $\beta_2 = 0.25$	$\alpha_2 = 0.25$ $\beta_2 = 0.25$	$\alpha_2 = 0.50$ $\beta_2 = 0.25$	$\alpha_2 = 0.90$ $\beta_2 = 0.15$	$\alpha_2 = 1.20$ $\beta_2 = 0.15$	$\alpha_2 = 0.50$ $\beta_2 = 0.15$
C_1 4 problems	0[a]	0	0	0	0	0	0	0	0	0
	829.26[b]	834.125	857.005	829.112	839.488	880.675	889.275	880.555	955.958	882.895
	0[c]	0	0.375	0	0	1.977	0	0.90	1.695	0
	829.26[d]	834.125	857.38	829.112	839.488	882.652	889.275	881.455	957.652	882.895
R_1 4 problems	0.25	0.25	0.25	0.25	0.25	0	0.25	0.25	0.25	0.25
	1272.71	1274.41	1267.60	1275.08	1265.19	1282.77	1302.15	1293.42	1282.50	1285.20
	51.997	52.532	43.627	39.537	45.217	46.415	51.71	60.865	58.432	56.23
	1324.71	1326.94	1311.23	1314.62	1310.41	1329.18	1353.86	1354.29	1340.93	1341.42
RC_1 4 problems	0.75	0.75	1.00	0.75	0.75	0	1.00	1.00	1.00	0.25
	1339.43	1344.64	1321.06	1325.66	1352.71	1366.24	1358.48	1380.77	1363.36	1334.95
	62.587	74.707	71.185	60.04	64.92	71.422	89.21	60.082	71.322	68.03
	1402.02	1419.34	1392.25	1385.70	1417.63	1437.66	1447.69	1440.86	1434.68	1402.98
C_2 4 problems	0	0	0.5	0	0	0	0.25	0	0	0
	615.518	614.925	616.928	616.878	624.21	601.972	650.912	612.018	630.335	603.122
	0	0	0	0	0	0	0	0	0	0
	615.518	614.925	616.928	616.878	624.21	601.972	650.912	612.018	630.335	603.122
R_2 4 problems	0	0	0.5	0	0	0	0.50	0	0.50	0.50
	1110.10	1095.61	1113.52	1091.81	1105.60	1089.10	1090.41	1083.09	1104.65	1101.28
	60.165	61.09	46.065	56.442	64.455	14.995	36.237	22.917	14.192	40.322
	1170.27	1156.70	1159.58	1148.25	1170.06	1104.09	1126.65	1106.00	1118.84	1141.60
RC_2 4 problems	0	0	0	0	0	0	0	0	0	0
	1186.05	1198.42	1172.54	1177.27	1198.94	1221.02	1207.80	1191.57	1207.86	1218.99
	23.472	29.502	23.862	25.457	32.652	35.692	28.942	40.562	46.522	27.662
	1209.52	1227.92	1196.40	1202.73	1231.60	1256.72	1236.75	1232.13	1254.38	1246.66

[a]Number of unserved customers.
[b]Total distance traveled.
[c]Total lateness.
[d]Objective value.

Best parameter constellations

- Rule 2:
  - Scenario 1:
    - α_1=0,5
    - β_1=0,15
  - Scenario 2:
    - α_1=0,5
    - β_1=0,10
- Rule 3:
  - Scenario 1:
    - α_2=0,5
    - β_2=0,25
  - Scenario 2:
    - α_2=0,25
    - β_2=0,25

TABLE IV

Comparison with the Original Algorithm for Both Scenarios

	Scenario 1			Scenario 2		
		New Algorithm			New Algorithm	
Problem Set	Original Algorithm	Rule 2 $\alpha_1 = 0.50$ $\beta_1 = 0.15$	Rule 3 $\alpha_2 = 0.50$ $\beta_2 = 0.25$	Original Algorithm	Rule 2 $\alpha_1 = 0.50$ $\beta_1 = 0.10$	Rule 3 $\alpha_2 = 0.25$ $\beta_2 = 0.25$
C_1 9 problems	0[a]	0	0	0	0	0
	835.098[b]	830.024	829.628	877.332	843.924	833.061
	5.535[c]	3.341	0	10.455	11.227	8.027
	840.633[d]	833.366	829.628	878.378	855.152	841.089
R_2 12 problems	0.5	0.166	0.166	0.4	0.333	0.333
	1219.42	1194.75	1178.54	1226.21	1203.99	1197.30
	43.267	48.625	51.460	55.431	67.167	49.121
	1262.69	1243.38	1230.00	1281.64	1271.16	1246.42
RC_1 8 problems	0.375	0.25	0.125	0.375	0.125	0
	1349.65	1304.58	1333.41	1400.50	1358.78	1346.98
	44.22	64.955	60.965	56.391	73.28	69.002
	1393.87	1369.53	1394.38	1456.89	1432.06	1415.98
C_2 8 problems	0.125	0	0	0	0	0
	612.798	609.232	597.011	610.775	600.796	595.635
	0.326	0	0	0	0	0
	613.124	609.232	597.011	610.775	600.796	595.635
R_2 11 problems	0.454	0.181	0.09	0.818	0.727	0.636
	1028.34	1035.09	1020.55	1043.96	1043.43	1027.82
	31.96	29.053	46.334	216.696	193.885	207.639
	1060.30	1064.15	1066.88	1260.65	1237.32	1235.46
RC_2 8 problems	0	0	0	0	0	0
	1203.75	1175.87	1160.83	1222.22	1189.48	1184.77
	30.811	22.108	20.672	27.441	41.023	19.568
	1234.56	1197.98	1181.50	1249.66	1230.51	1204.34
Overall 56 problems	0.27	0.10	0.07	0.30	0.23	0.19
	1049.83	1034.11	1027.95	1070.75	1048.45	1039.11
	27.20	29.10	31.79	68.10	70.61	65.25
	1077.03	1063.22	1059.74	1138.85	1119.06	1104.36

[a]Number of unserved customers.
[b]Total distance traveled.
[c]Total lateness.
[d]Objective value.

Consequences

- It becomes obvious that the additional control variables achieved by integrating the diversion possibilities can lead to significantly improved results
- Applying Rule 2
  - Scenario 1:

    Both, the number of non serviced customers (33,3-100%) and the objective value (0,6-3%) were improved with regard to the original algorithm
  - Scenario 2:

    Reduces the number of non serviced customers (11,2-67%) and the objective function value (1,0-2,7%) in all categories tested

Consequences

- Applying Rule 3:
  - Scenario 1:

    Both, the number of non serviced customers (66,8-100%) and the objective value (2,6-2,7%) were improved for most scenarios with regard to the original algorithm. Improves the objective value of rule 2 (0,5-2%)
  - Scenario 2:

    Reduces the number of non serviced customers (16,8-100%) and the objective function value (2,0-4,3%) in all categories tested

➢ The version that applies rule 3 outperforms the other procedures

Placement in the classification

- Adaptation frequency

  Continuous + disturbance triggered
- Adaptation synchrony

  Process priority / Simultaneity
- Adaptation technique

  Tabu Search
- Adaptation range

  VRP variables reduced by decisions for the current tour

  BUT:

  In case of a disturbance only time-limit depending fixing
- Adaptation duration

  Continuous adaptation processes + time limit dependent (flexible time limit)

Pros and Cons

- Pros
  - Real-time oriented approach with customizable adaptation range
  - Self-adapting determination of reasonable time-limits
  - Integration of the current degree of dynamism
- Cons
  - Excessive use of diversion activities may stress and distract the drivers (see Ferrucci and Bock (2015)) → Additional costs due to accidents or fluctuation may occur
  - Parameter setting of the applied rules is not robust

7.4 Exploiting past request data

- In what follows, we consider a somewhat modified problem constellation
- Specifically, real-time approaches introduced before are extended in order to exploit available knowledge about future transportation requests
- This knowledge is used in order to use the resources more efficiently
- The basic approach relies on the assumption that future events often arise in a somewhat predictable way (e.g., probability distributions can be derived from historical request data)

Motivation

- Project: Subsequent delivery of newspapers
- Typical problem: Sometimes subscribers do not receive their newspaper
- Typical reasons are for instance
  - Technical or organizational problems
  - New delivery man
  - Thievery

- However: Newspapers are a **highly perishable** good and therefore extremely time-critical
- Installment of a subsequent delivery service
  - Subsequent delivery has to be executed very quickly, otherwise the service has no value for the subscribers
  - Publishers pursue maintaining a high number of subscribers

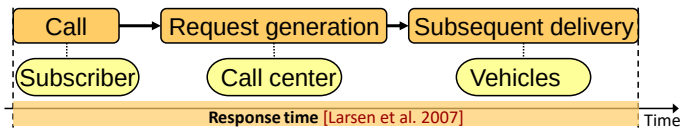
Typical example of urgent deliveries

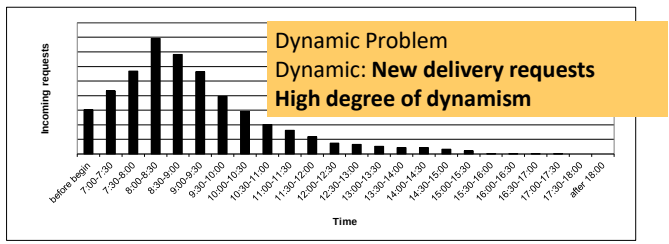
Process: Urgent deliveries

- Repairmen: Production breakdown → Service level agreements
- Subsequent delivery of **newspapers** (due to thievery or delivery failure)

Important: Quick delivery of goods

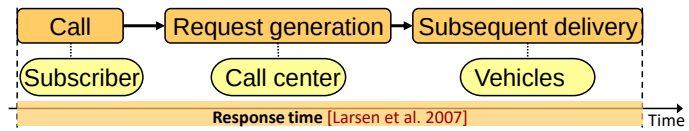
Process description and aims





Dynamic Problem
Dynamic: **New delivery requests**
High degree of dynamism

Process description and aims



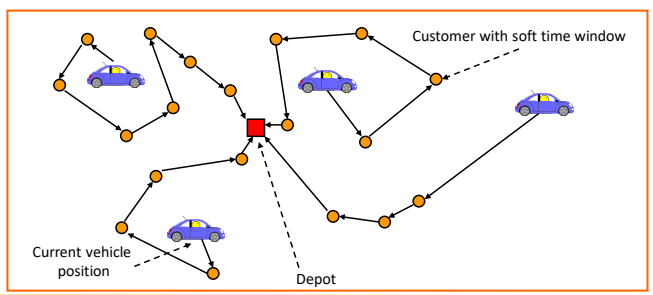
Practical aim:
Reduction of the response time in order to
- minimize customer inconvenience and to
- communicate ambitious service levels that may attract new customers

Considered optimization problem

- Dynamic problem
  - A certain proportion of requests is known, further requests arrive dynamically (only source of dynamism (see 7.2 and 7.3))
  - Problem is mapped by a sequence of static problems that result from each other and are iteratively solved
  - Diversion is integrated, i.e., the first request currently assigned to a tour of a vehicle is allowed to be changed
- Static problems
  - Given
    - Homogenous fleet of vehicles (Velocity, Costs, no capacity constraints) with current locations in the mapped road network
    - Urgent delivery requests with individual time assignments
    - Detailed road network of a medium sized city is used
  - Sought (in the static problems)
    - Assignment of all known requests to vehicles
    - Generation of detailed tour plans for each vehicle (sequence of visited locations, detailed time table)

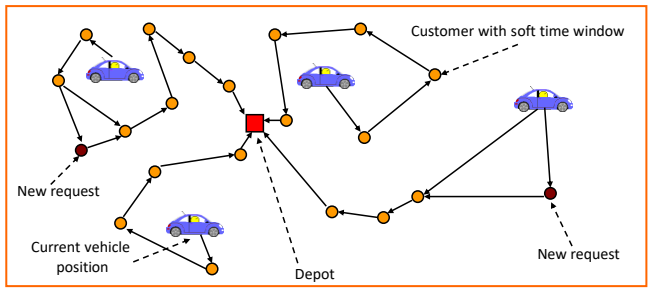
Characterizing the optimization problem

- The considered problem is a „Dynamic Vehicle Routing Problem with soft time window restrictions“ (DVRPSTW)
- Time windows open directly with incoming requests
- End of time window is determined by the maximum service time



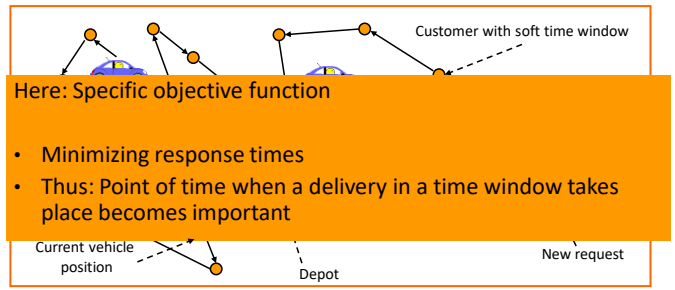
Characterizing the optimization problem

- The considered problem is a „Dynamic Vehicle Routing Problem with soft time window restrictions“ (DVRPSTW)
- Time windows open directly with incoming requests
- End of time window is determined by the maximum service time



Characterizing the optimization problem

- The considered problem is a „Dynamic Vehicle Routing Problem with soft time window restrictions“ (DVRPSTW)
- Time windows open directly with incoming requests
- End of time window is determined by the maximum service time



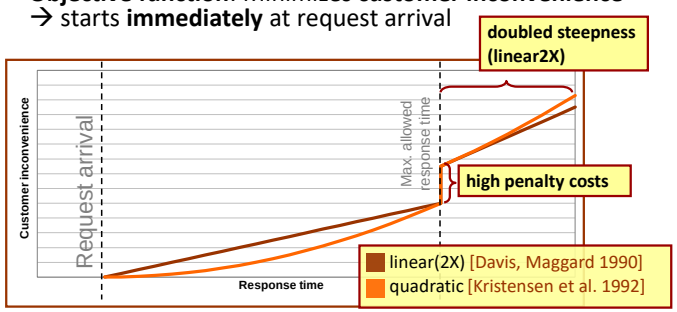
Here: Specific objective function

- Minimizing response times
- Thus: Point of time when a delivery in a time window takes place becomes important

Specific objective function

Real-time routing problem, modeled as a **DVRPSTW**:

- Specific time windows and objective function
- **Objective function**: Minimizes **customer inconvenience** → starts **immediately** at request arrival



Problem definition

- τ Current time assignment of the considered static problem
- $K = \{1, \dots, m\}$: Set of homogenous vehicles starting their respective delivery tour at the single depot
- $R = \{1, \dots, n\}$: Set of customer requests that occur over the day
- $R_\tau = \{i \in R | a_i \leq \tau\}$: Set of customers known at time τ
- Each request $i \in R$ has an arrival date a_i and can be directly serviced upon arrival so that its time window e_i opens at time a_i
- R^{mrt} maximum allowed response time that is identical for all requests
- s_i service time of request $i \in R$ (set to 60 seconds for real requests)
- y_i planned time when service begins for request $i \in R$
- $R_\tau(U) = \{i \in R_\tau | y_i > \tau\}$: Set of customers that are non-serviced at time τ

Objective function

$$\min Z = \sum_{i \in R_+(U)} w_i \cdot \left(F(t_i) + R^{pen} \cdot \Theta(t_i - R^{mrt}) \right)$$

w_i: weight of request (=1 for real requests; dummy request obtain their ocurrence propability)

$F(t_i)$: Variable inconvenience

$R^{pen} \cdot \Theta(t_i - R^{mrt})$: Lateness inconvenience

with $t_i = y_i - e_i$ and $\Theta(x) = 1$ if $x > 0$ and 0 otherwise.

F is defined either as a linear or a quadratic customer inconvenience function

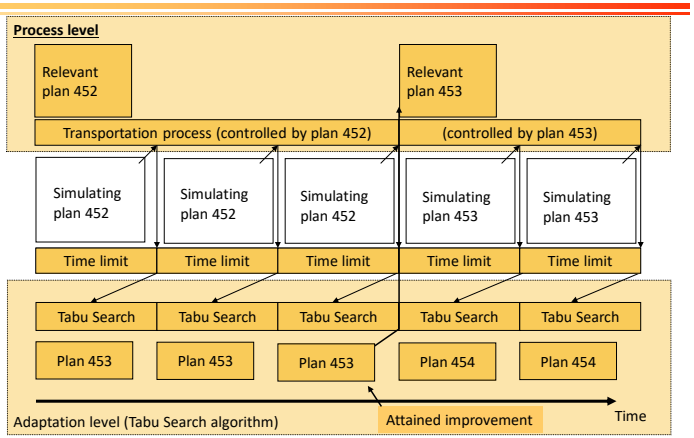
Significant differences in both inconvenience functions:

- Independent of linear or quadratic inconvenience function, a response time equal to R^{mrt} results in a customer inconvenience contribution of 1.0
- whereas exceeding R^{mrt} is penalized by R^{pen}=100

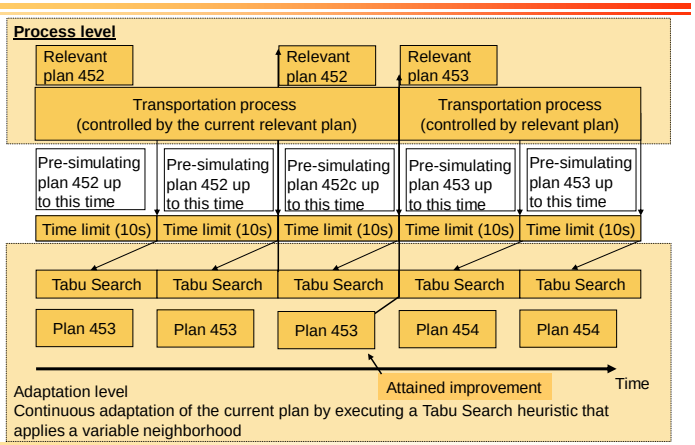
Reactive deterministic real-time approach

- Contradictive demands on a real-time control
  - Fast response to unexpected disturbances
  - Comprehensive plan adaptations resulting in optimal or near optimal transportation plans
  - Possible workaround
    - Instant plan correction in order to guarantee feasibility
    - Continuous plan adaptation by applying a sophisticated metaheuristic
- Deterministic real-time approach [Bock 2004, 2010], [Gendreau et al. 1999], [Gendreau and Potvin 1998]
  - Concurrency of tour plan execution and continuous tour plan adaptation and allowing diversion
  - Specific adaptation handling is necessary
  - Time elapse is separated into small anticipation horizons

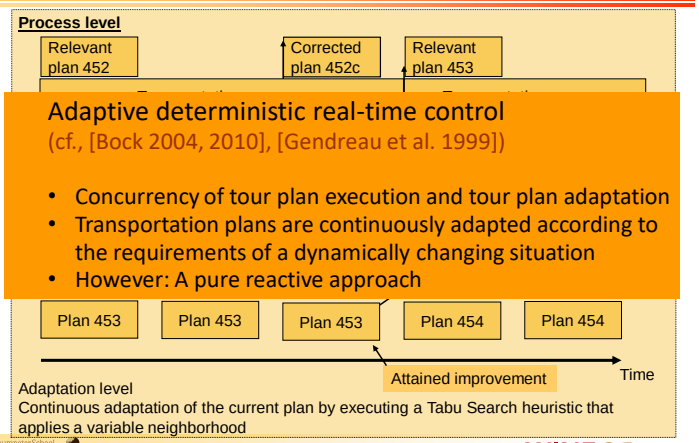
Real-time control process



Deterministic real-time approach



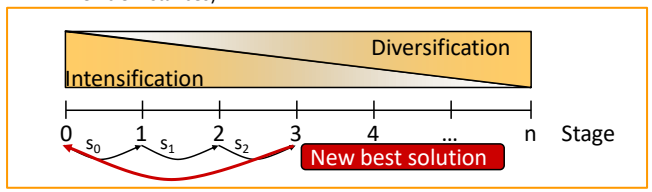
Deterministic real-time approach



Applied solution procedure

- Solution method: Tabu Search based metaheuristic
  - Variable neighborhood with different stages
  - Stage determines the applied neighborhood operations
  - Higher stages allow more substantial modifications
  - Test series with static instances prove a high efficiency

    Compared to tight lower bounds provided by the exact approach of Westphal and Krumke (2008), a gap of only 0.017 percent occurs (10 vehicle instances)



Generation of an initial tour plan

- An initial tour plan is generated at the beginning of the day by an iterative least cost insertion of all real requests known at this point in time followed by the dummy customers.
- Dummy customer requests are integrated in sequence of non-decreasing time window start times.

Neighborhood operators of the Tabu Search

Depending on the current state of the searching process, the used Tabu search approach applies the following *various neighborhood operators*:

- **Within Tour Insertion (WTI)**
  - WTI removes a single request and reinserts it at its best position on the same tour if this leads to a solution improvement.
  - For each tour the best relocation that is not tabu is implemented
- **Relocate (REL)**
  - REL and WTI work similarly,
  - but REL evaluates all positions that are on tours *other than the one* a request is currently assigned to

Neighborhood operators of the Tabu Search

- **MultiRelocate (MREL)**
  - MREL removes *n* requests where n is randomly drawn out of the interval [MREL^{reqMin},MREL^{reqMax}]
  - They are successively reinserted at their least cost positions by considering all possible reinsertion permutations.
  - The *n* requests are randomly chosen out of the *n* MREL^{reqSelRatio} requests with the highest objective function value contribution.
  - In MREL, the entire procedure is repeated *c* times, (the parameter *c* is randomly chosen out of the interval [MREL^{execMin},MREL^{execMax}])
  - The best non-tabu move is executed

Neighborhood operators of the Tabu Search

- **Large Neighborhood Search (LNS)**
  - LNS reinserts *n* requests in the sequence of non-increasing cost contributions where *n* is set to $\max\left\{2, m \cdot \left|R_{\tau^+}^U\right|\right\}$
  - and *m* is randomly drawn out of the interval [0,LNS^{reqRemFactor}]
  - The *n* requests are randomly chosen out of the *d* requests with the highest objective function value contribution, with $d = \min\left\{\left|R_{\tau^+}^U\right|, n \cdot LNS^{reqSelRatio}\right\}$
  - This is repeated *c* times where *c* is randomly chosen out of the interval [LNS^{execMin},LNS^{execMax}]

Neighborhood operators of the Tabu Search

- **Exchange Between Tours (XBT)**
  - XBT exchanges two requests of two tours
  - After evaluating all combinations, the best non-tabu move is performed

Stage-based selection scheme

- Depending on the results of the ongoing search process, the Tabu Search approach is in a different stage
- It starts with the initial state 1
- The current stage deterministically determines the applied operator and the condition under which a stage switch to the next higher indexed stage is performed
- If a defined number of iterations has been performed without finding a new best solution, the Tabu Search algorithm switches into the next higher stage and continues its search with a different neighborhood operator.

Stage	Operator	Switch after number of iterations
1	WTI	0 (only improvements are accepted)
2	REL	10
3	MREL	10
4	LNS	1,000
5	XBT	5

Basic idea behind the different stages

- Depending on the current stage a specific distance of promising, still-unexplored solutions to the currently considered plan is assumed
- Due to a large number of recent consecutively failed attempts at improving the currently best known solution, this distance is assumed to be significantly larger in higher stages than in lower stages
- Therefore, by switching to higher stages, the search process increasingly applies more globally-acting neighborhood operators that provide a higher diversification
- Moreover, whenever a move leads to a new overall best known solution, the search process is intensified in its direct vicinity by switching back into the first stage

Applied parameter values for MREL and LNS

- The following parameter values are applied for the operators MREL and LNS
- This values reflect basic cognitions derived from first evaluations
- However, the values were not optimized by empirical studies

Parameter	Value	Parameter	Value
$MREL^{reqMin}$	2	$LNS^{reqRemFactor}$	0.75
$MREL^{reqMax}$	3	$LNS^{reqSelRatio}$	1.5
$MREL^{reqSelRatio}$	3	$LNS^{execMin}$	1
$MREL^{execMin}$	10	$LNS^{execMax}$	2
$MREL^{execMax}$	20		

Tabu status

- A fingerprint of a considered tour plan is applied as the **tabu-active attribute**
- It is implemented by combining the following two quickly computable checksums
  - CRC-32 (Moon (2005), pp. 147)
  - Adler-32 (Deutsch (1996))
- Two tour plans are interpreted as identical if both checksums coincide with each other
- Since empirical tests indicate that this combination allows a quite reliable identification of tour plans, an assigned tabu status is never revoked during the searching process

Computational results

- The application of the deterministic approach yields significant improvements
- Average reduction of response times by 40 percent (in best cases by up to 70 percent)
- Efficient handling of incoming requests
- No late deliveries after 60 minutes
- However, detailed analyses reveal that
  - there are frequently situations where vehicles have to perform considerable detours in order to provide service in remote areas
  - request forecasting may be very beneficial
- **Upcoming research question:**

  Can we further improve these promising results by the integration of stochastic knowledge?

Pro-active: Using stochastic knowledge

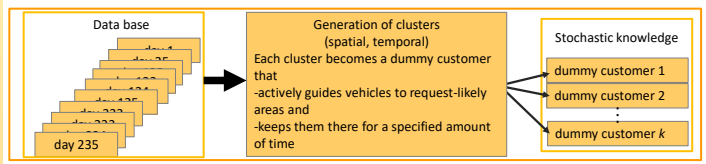
- Forecast future requests (of upcoming day)
- Intensively discussed topic in recent literature
  c.f. [Ichoua et al. 2006], [Hvattum et al. 2006], [Hentenryck and Bent 2009], [Hvattum et al. 2007]

Methods

1. Vehicle waiting strategy:
   Keeping vehicles in **request-likely areas**
2. Use of stochastic (dummy) customers:
   Routing vehicles a priori to **request-likely areas**

Stochastic approach – Basic ideas

- Analyzing **past request information of the last n^f days** that are assumed to be representative for the next day in order to derive stochastic knowledge
- The derived knowledge is integrated into the deterministic real-time control approach by the generation of so-called **dummy (customer) requests**
- These dummy requests are generated in an offline step (before the application of the real-time control)

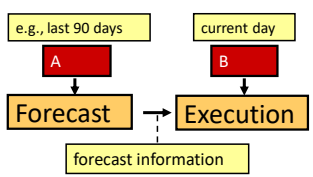


The forecast approach

1. Waiting: Vehicle has no pending requests
→ waiting at current position

2. Dummy customers: Routing vehicles to request-likely areas (using forecast information)

- Generate the forecast information out of **real past request information**
  - Two distinct processes:
    A: past requests (known)
    B: current day requests (unknown)

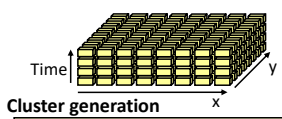


- Difference to many other approaches: No closed formulation for request arrivals

→ **Coming closer to reality**

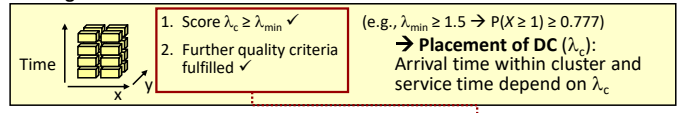
Cluster&DC generation

- Dividing the day into **segments** (time and space)
  e.g., 1min x 2.5 x 2.5 km²



Assumption: Each segment represents a **spatial Poisson process**
→ segment values λ_s: SMA(90)

Cluster generation



- Aim: Find the maximum number of valid clusters
  1. MIP formulation: Definition of all theoretical valid clusters (**allowing overlapping**)
  2. CPLEX: Selection of maximum number of clusters **without overlapping**, earliest possible start

Cluster generation – Quality criteria

Segments are combined into clusters so that the following quality criteria are fulfilled

1. A cluster does not exceed a predefined maximum spatial and temporal extension DC^{mse} and DC^{mte} respectively
2. The sum of the rate values of assigned segments is at least DC^{mink}
3. Further quality aspects are met
   1. If the average travel in the cluster is above a certain threshold the cluster is discarded
   2. Poisson quality is checked by performing Pearson's Chi-Square Goodness-of-Fit test (cf. Kvam and Vidakovic, (2007), pp. 155)

Cluster generation

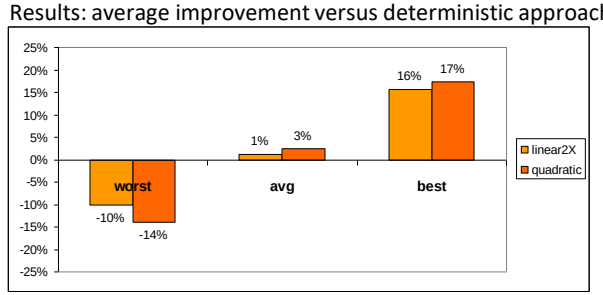
- Preliminary tests revealed that clusters of desired quality cannot be generated by several well-known approaches such as the k-Means algorithm (cf. MacQueen (1967), Arthur and Vassilvitskii (2007))
- First, k-Means requires the number of clusters to be specified. In our case, this is inappropriate since we aim at maximizing the number of generated clusters which depends on the quality of the past request information.
- Moreover, considering criteria 1, cluster extensions cannot be controlled and clusters generated by k-Means are rarely compact and often irregular

Cluster generation

- First, all valid clusters are generated, i.e., this includes overlapping between all these alternative clusters
- Second, the maximum number of non-overlapping clusters is selected by solving a specific MIP
  - The primary goal of the objective function of the MIP is the maximization of the number of selected clusters.
  - The secondary goal is to start clusters at the earliest point in time in order to support an earlier service of expected requests
- The MIP can be optimally solved in reasonable time by a standardized solver, for instance by CPLEX
- Each generated cluster defines a dummy customer possessing
  - a weight that corresponds to the occurrence probability
  - a service time that represents the sum of average travel and service time for fulfilling the expected requests in the cluster

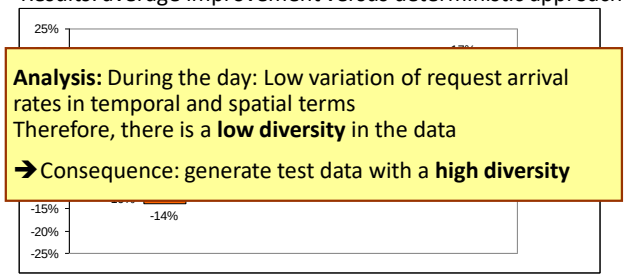
Computational results: Real request data

- **Real road network** for simulation, distance, and time calculations
- 10 vehicles, service time of each real request: 60s
- 30 randomly chosen days, #requests: min 126, max 182, Ø 151.7
- Cluster acceptance $\lambda_{min} \geq 1.8 \rightarrow P(X \geq 1) \geq 0.835$
- Results: average improvement versus deterministic approach



Computational results: Real request data

- **Real road network** for simulation, distance, and time calculations
- 10 vehicles, service time of each real request: 60s
- 30 randomly chosen days, #requests: min 126, max 182, Ø 151.7
- Cluster acceptance $\lambda_{min} \geq 1.8 \rightarrow P(X \geq 1) \geq 0.835$
- Results: average improvement versus deterministic approach

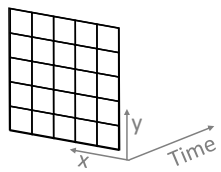


Analysis: During the day: Low variation of request arrival rates in temporal and spatial terms
Therefore, there is a **low diversity** in the data

➔ Consequence: generate test data with a **high diversity**

Diversity: Generating data with useful structure

Dividing the considered area into **regions**

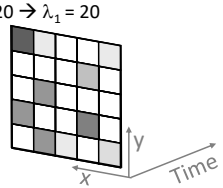


Diversity: Generating data with useful structure

Dividing the considered area into **regions**

- Different **request arrival probabilities** by region

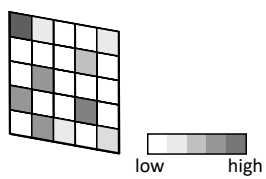
Øreq: 20 ➔ $\lambda_1 = 20$



Diversity: Generating data with useful structure

Dividing the considered area into **regions**

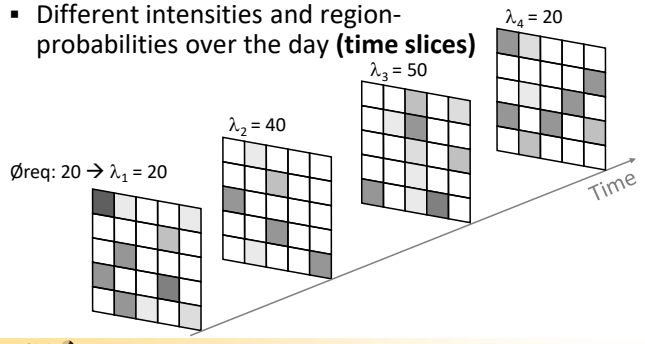
- Different **request arrival probabilities** by region
- Different intensities and region-probabilities over the day **(time slices)**



Diversity: Generating data with useful structure

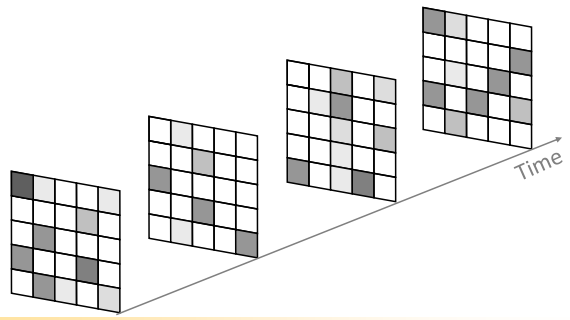
Dividing the considered area into **regions**

- Different **request arrival probabilities** by region
- Different intensities and region-probabilities over the day **(time slices)**



Request data: Structural diversity [1/2]

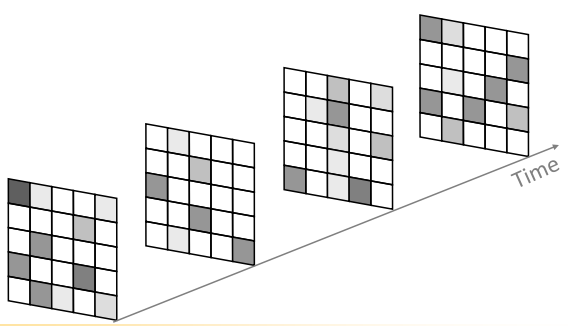
Reducing diversity: **Step-wise averaging** in two dimensions



Request data: Structural diversity [1/2]

Reducing diversity: **Step-wise averaging** in two dimensions

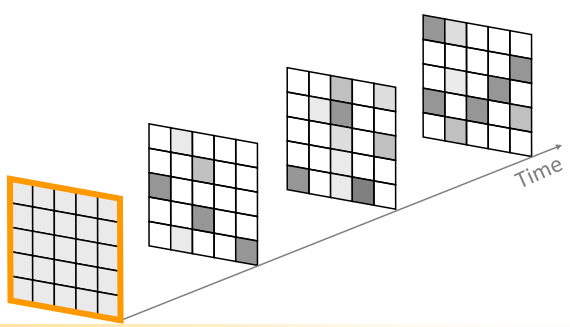
1. RegionDiversity (RD): Averaging regional differences within **one time slice**



Request data: Structural diversity [1/2]

Reducing diversity: **Step-wise averaging** in two dimensions

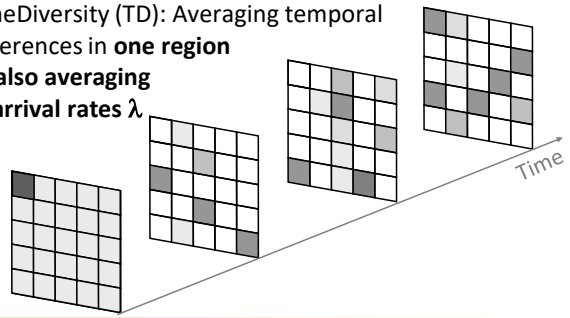
1. RegionDiversity (RD): Averaging regional differences within **one time slice**



Request data: Structural diversity [1/2]

Reducing diversity: **Step-wise averaging** in two dimensions

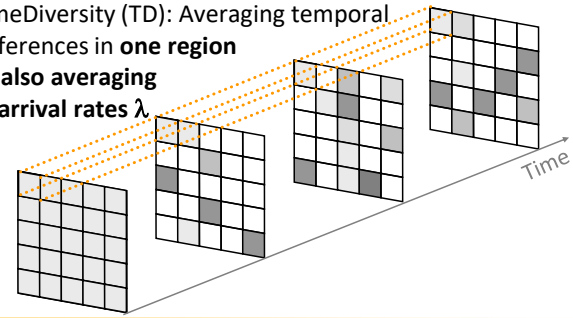
1. RegionDiversity (RD): Averaging regional differences within **one time slice**
2. TimeDiversity (TD): Averaging temporal differences in **one region**
   → **also averaging of arrival rates λ**



Request data: Structural diversity [1/2]

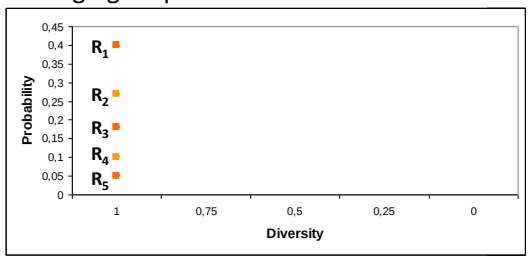
Reducing diversity: **Step-wise averaging** in two dimensions

1. RegionDiversity (RD): Averaging regional differences within **one time slice**
2. TimeDiversity (TD): Averaging temporal differences in **one region**
   → **also averaging of arrival rates λ**



Request data: Structural diversity [2/2]

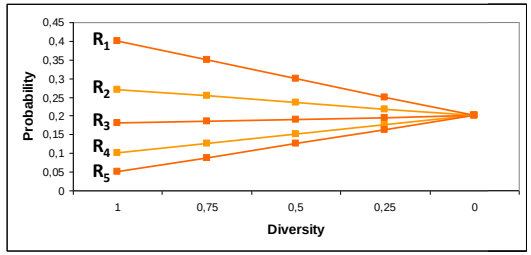
- Averaging: Implemented as **linear transformation**



- **Combination of both diversity dimensions**
  - RegionDiversity (RD) and TimeDiversity (TD): 5 steps each: [1.00, 0.75, 0.50, 0.25, 0.00]

Request data: Structural diversity [2/2]

- Averaging: Implemented as **linear transformation**



- **Combination of both diversity dimensions**
  - RegionDiversity (RD) and TimeDiversity (TD): 5 steps each: [1.00, 0.75, 0.50, 0.25, 0.00]

Computational results: Structural request data

- 10 vehicles, service time of each real request: 60s
- Each setting: 30 days, # of requests: min 92, max 172, Ø 130.1
- **Different λ_{min} values tested (1.0, 1.2, 1.5, 1.8, 2.0)**
- Results: average improvement versus deterministic approach (pure reactive approach)

TD \ RD lin2X	1.00	0.75	0.50	0.25	0.00
1.00	23.5%	16.7%	9.7%	5.3%	1.8%
0.75	14.3%	10.8%	7.3%	4.8%	3.7%
0.50	10.3%	8.2%	4.6%	3.7%	3.9%
0.25	7.2%	6.7%	5.1%	2.6%	1.9%

0/30 worse | 1-3/30 worse | 4-9/30 worse | > 9/30 worse

Computational results: Structural request data

- 10 vehicles, service time of each real request: 60s
- Each setting: 30 days, # of requests: min 92, max 172, Ø 130.1
- **Different λ_{min} values tested (1.0, 1.2, 1.5, 1.8, 2.0)**
- Results: average improvement versus deterministic approach (pure reactive approach)

TD \ RD quad	1.00	0.75	0.50	0.25	0.00
1.00	38.6%	27.3%	17.8%	11.6%	7.1%
0.75	25.6%	20.0%	14.0%	12.5%	10.8%
0.50	20.7%	17.7%	13.9%	11.0%	7.9%
0.25	13.2%	13.4%	10.6%	7.5%	5.8%

1-3/30 worse

Computational results: Structural request data

- 10 vehicles, service time of each real request: 60s
- Each setting: 30 days, # of requests: min 92, max 172, Ø 130.1
- **Different λ_{min} values tested (1.0, 1.2, 1.5, 1.8, 2.0)**
- vs. myopic approach (**no forecasting**) but improvements for all λ_{min} values

Less vehicles: 8 instead of 10

TD \ RD lin2X	1.00	0.75	0.50	0.25	0.00
1.00	38.6%	27.3%	17.8%	11.6%	7.1%
0.75	25.6%	20.0%	14.0%	12.5%	10.8%
0.50	20.7%	17.7%	13.9%	11.0%	7.9%
0.25	13.2%	13.4%	10.6%	7.5%	5.8%

best results with **higher** lambda values

best results with **lower** lambda values

1-3/30 worse

Computational results: Structural request data

- 10 vehicles, service time of each real request: 60s
- Each setting: 30 days, # of requests: min 92, max 172, Ø 130.1
- **Different λ_{min} values tested (1.0, 1.2, 1.5, 1.8, 2.0)**
- vs. myopic approach (**no forecasting**) but improvements for all λ_{min} values

More vehicles: 12 instead of 10

TD \ RD lin2X	1.00	0.75	0.50	0.25	0.00
1.00	38.6%	27.3%	17.8%	11.6%	7.1%
0.75	25.6%	20.0%	14.0%	12.5%	10.8%
0.50	20.7%	17.7%	13.9%	11.0%	7.9%
0.25	13.2%	13.4%	10.6%	7.5%	5.8%

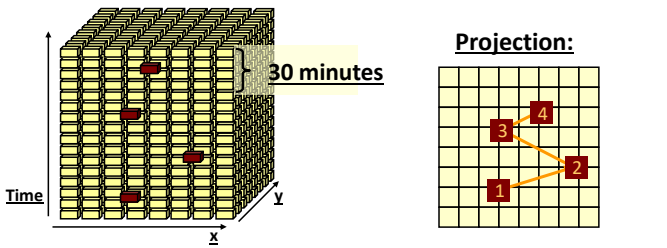
best results with **higher** lambda values

best results with **lower** lambda values

1-3/30 worse

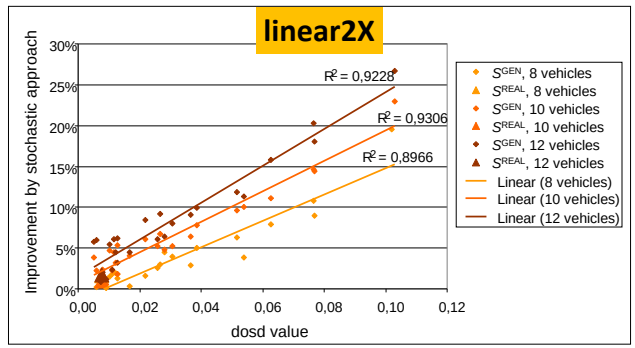
The degree of structural diversity (dosd)

1. Separation into time intervals (e.g. 30 minutes)
2. Generation of the average **barycenter of each interval**
3. Calculation of **distances between barycenters**



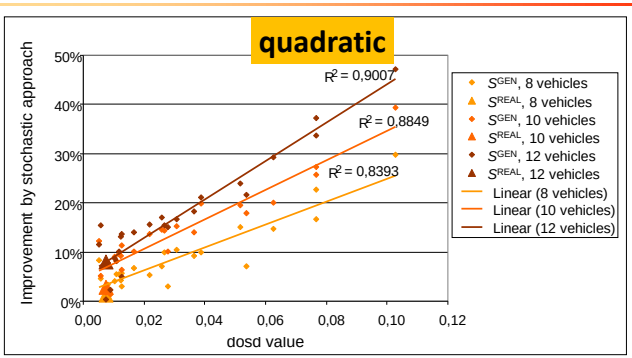
***dosd* =** ≈ 0,4

The *dosd* value: Measured results



- **Strong correlation between** *dosd* and the attained improvements ➔ *dosd* is a suitable offline criterion

The *dosd* value: Measured results



- **Strong correlation between** *dosd* and the attained improvements ➔ *dosd* is a suitable offline criterion

Number of generated dummy customers

- The number of generated dummy customers represent the quality of the derived stochastic knowledge
- Hence, the following table indicates this number in dependence of the required minimum stochastic quality and the diversity in the past request data

Level of structural diversity	$DC^{min\lambda}$				
	1.0	1.2	1.5	1.8	2.0
RD=0.00, TD=0.00	2	0	0	0	0
RD=0.25, TD=0.25	17	3	0	0	0
RD=0.50, TD=0.50	36	20	7	0	0
RD=0.75, TD=0.75	54	38	24	15	11

Conclusions and future research

Attained results

- DC approach: Significant improvement of customer service quality
- Necessary: Data has a useful structure (high diversity values)
- High diversity: More structure → choose high quality clusters, in particular if the fleet size is strongly limited
- The proposed *dosd* measure is a useful offline criterion for deciding whether the integration of stochastic knowledge is promising

Future research

- Improvement of dummy customer methods by more flexible cluster structures
- Extension to more complex applications

7.5 Job Shop Scheduling in real-time

- In what follows, we consider the real-time control approach for Job Shop Scheduling of Bierwirth and Mattfeld (1999)

The static model

- n Jobs $J_1, J_2, \ldots, J_n$ have to be processed
- Each job consists of at most m operations
- Therefore, m_i – denoting the number of operations of the i-th job – is lower or equal to m
- Operations are denoted as $o_{i,j}$
- Machines are denoted as $M_1,\ldots,M_m$
- Each job i has an individual process recipe μ_i that determines the machine sequence of the job
- Thus, $\mu_i(k)$ gives the index of the machine that processes the k-th operation of job i, i.e., it is machine $M_{\mu_i(k)}$

Further notations

- Processing time $o_{i,k}$ is denoted by $p_{i,k}$
- Starting times are given by $t_{i,k}$
- Hence, the completion time C_i is $t_{i,k}+p_{i,k}$
- Owing to predecessors on the machine (job h) and preceding operations on stage k-1, it can be stated:

$$t_{i,k} = \max\left\{t_{i,k-1} + p_{i,k-1}, t_{h,l} + p_{h,l}\right\}$$
$$\text{with } \mu_i(k) = \mu_h(l)$$

Dynamic aspects

- Extension of the static case
- Release times r_i of each job have to be respected
- Now, C_{max} is substantially affected by the release dates, particularly by the largest one
- Consequently, it is **no longer an appropriate objective function**
- Instead, we take the **mean flow time of jobs**. It is defined as follows

$$\bar{F} = \frac{1}{n} \cdot \left(\sum_{i=1}^{n} C_i - r_i \right)$$

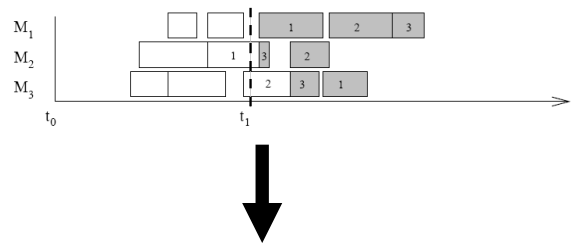
Dynamic handling

- As nearly all dynamic approaches, the concept carries out its dynamic processing like a sequence of static problem solving
- Therefore, a problem considered at t_0 is transformed to a new one at t_1
- The trigger event is the arrival of new requests in the control system
- At time t_1
  - all operations already processed between t_0 and t_1 are deleted from the problem definition (the machine is kept unavailable)
  - If no operation remains for a job currently in consideration, this request is entirely erased
  - Otherwise, the release dates of the affected requests are updated to respect the fixed operations

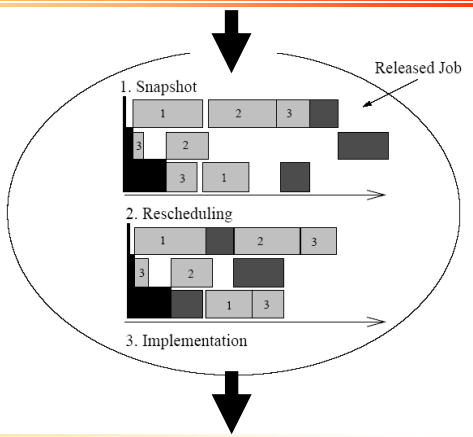
Integration of setup time

- Since **machines are blocked** by operations commenced before t_1, we need additional instruments to indicate an unavailability of a resource
- For this purpose, **artificial setup times** s_j are additionally introduced
- They are determined for each machine by the longest processing of an operation on this machine that has started before t_1
- An instance of this processing is provided by the following simple depiction

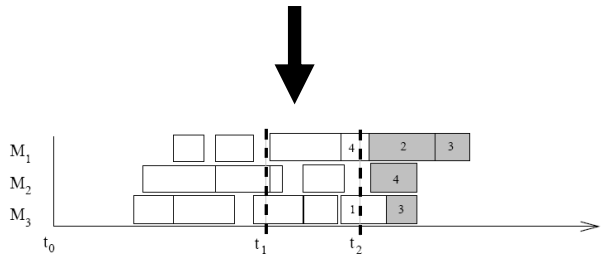
Production scheduling on a rolling time basis



Production scheduling on a rolling time basis



Production scheduling on a rolling time basis



Genetic encoding

- Permutation coding
- Each operation in an instance is clearly defined, e.g., A, B, C, D, …, Z,….
- Thus, a solution can be interpreted by a single permutation, i.e., the permutation determines the priority of the respective operations if ties have to be broken
- Note that this definition does not guarantee any feasibility!
- Therefore, **feasibility has to be ensured during the decoding process**

Genetic operators

We have to define

- **Crossover operations**
  Here, the PPC operation is applied
- **Mutation operations**
  Here, an existing permutation is slightly changed. More precisely, an operation is deleted and randomly reinserted in a different position

Precedence Preservative Crossover (PPC)

parent permutation 1	A	B	C	D	E	F
parent permutation 2	C	A	B	F	D	E
select parent no. (1/2)	1	2	1	1	2	2
offspring permutation	A	C	B	D	F	E

Permutation decoding

- The authors provide decoding procedures that result either in semi-active, active or non-delay schedules
- In what follows, brief depictions for all these cases are given
- Note again that there is no performance dominance between the different schedule groups

Semi-active schedules

- Here, schedules arise whenever an earlier processing of operations cannot be achieved without altering the job sequence on a machine
- All available operations (operations with no non serviced predecessor) are stored in a specific operation set A
- One by one they are all scheduled according to the characterizing permutation vector

The decoding procedure

1. Build the set of all beginning operations, $A := \{o_{i1} \mid 1 \leq i \leq n\}$.
2. Select operation o^*_{ik} from A which occurs leftmost in the permutation and delete it from A, $A := A \backslash \{o^*_{ik}\}$.
3. Append operation o^*_{ik} to the schedule and calculate its starting time.
4. If a job successor operation $o^*_{i,k+1}$ of the selected operation o^*_{ik} exists, insert it into A, $A := A \cup \{o^*_{i,k+1}\}$.
5. If $A \neq \emptyset$ goto Step 2, else terminate.

Why we fail to produce active schedules

- The procedure above is obviously not able to guarantee active schedules
- Why?
- Reasons
  - Jobs in A are processed only according to the permutation vector
  - Hence, jobs whose earlier processing would not effect completion times of the other operations are potentially delayed
  - Consequently, the resulting schedule does not have to be an active one

Active schedules

- Step 2 has to be modified
- We have to exclude all operations from A that do not start before the earliest completion time of all requests in A
- Therefore, a set B is introduced as a specific subset of A
- B consists of all available operations whose first starting time is smaller than the earliest completion time of all requests in A

Decoding procedure

2.A1 Determine an operation o' from A with the earliest possible completion time, $t' + p' \leq t_{ik} + p_{ik}$ for all $o_{ik} \in A$.

2.A2 Determine the machine M' of o' and build the set B from all operations in A which are processed on M', $B := \{o_{ik} \in A \mid \mu_{i(k)} = M'\}$.

2.A3 Delete operations in B which do not start before the completion of operation o', $B := \{o_{ik} \in B \mid t_{ik} < t' + p'\}$.

2.A4 Select operation o^*_{ik} from B which occurs leftmost in the permutation and delete it from A, $A := A \backslash \{o^*_{ik}\}$.

Non-delay schedules

- In these schedules, machine waiting time (i.e., idle time) is not allowed
- Consequently, for each machine, only the earliest available jobs compete to be processed
- Thus, B is reduced to all jobs with the smallest availability time

Decoding procedure

2.N1 Determine an operation o' from A with the earliest possible starting time, $t' = t_{ik}$ for all $o_{ik} \in A$.

2.N2 Determine the machine M' of o' and build the set B from all operations in A which are processed on M', $B := \{o_{ik} \in A \mid \mu_{i(k)} = M'\}$.

2.N3 Delete operations in B which start later than operation o', $B := \{o_{ik} \in B \mid t_{ik} = t'\}$.

2.N4 Select operation o^*_{ik} from B which occurs leftmost in the permutation and delete it from A, $A := A \backslash \{o^*_{ik}\}$.

Hybrid schedules

- Combination of active and non-delay schedules
- Note that the set of optimal schedules is a subset of the unification of both sets
- Introduction of an additional parameter δ out of the continuous interval [0,1]
- At the extremes, δ=0 produces non-delay schedules while δ=1 generates an active one
- Those can be generated as follows

Decoding procedure

2.H1 Determine an operation o' from A with the earliest possible completion time, $t' + p' \leq t_{ik} + p_{ik}$ for all $o_{ik} \in A$.

2.H2 Determine the machine M' of o' and build the set B from all operations in A which are processed on M', $B := \{o_{ik} \in A \mid \mu_{i(k)} = M'\}$.

2.H3 Determine an operation o'' from B with the earliest possible starting time, $t'' = t_{ik}$ for all $o_{ik} \in B$.

2.H4 Delete operations in B in accordance to parameter δ such that $B := \{o_{ik} \in B \mid t_{ik} < t'' + \delta((t' + p') - t'')\}$.

2.H5 Select the operation o^*_{ik} from B which occurs leftmost in the permutation and delete it from A, $A := A \backslash \{o^*_{ik}\}$.

Active versus non-delay

- Obviously, non-delay schedules are the most restrictive ones
- In non-delay schedules, some optimal schedules are excluded from consideration
- **In best constellations**, where the GA has the potential to converge to optimality, **active schedules are able to attain optimal constellations**
- **In best constellations**, where the GA has the potential to converge to optimality, **non-delay schedules are able to attain optimal constellations**
- On the other hand, the algorithm will **converge faster if non-delay scheduling is used**

Active versus non-delay

- **In worse constellations**, where the GA has not the potential to converge to optimality, **active schedules are frequently outperformed by non-delay schedules**
- On the other hand, the algorithm will **converge faster if non-delay scheduling is used**

Parameterization

- Fitness measure is defined as the mean flow time
- Selection probability is determined by the inverted mean flow time
- Population size 100
- Crossover operation is applied in 80 percent of the cases
- With a probability of 0.2, the mutation operation is used
- Stopping criteria

  If T steps are executed without any improvement, the algorithm stops. Within the tests, T is confined to the number contained in a problem instance

The online procedure

- The hybrid scheduler is used throughout the control process
- After being decoded, the respective permutation is rewritten with the sequence of operations as they have actually been scheduled
- This hybrid scheduler should overcome the limitations of genetic procedures
  - Tests have shown that the capabilities of genetic procedures for converging to near-optimality vanish with increasing problem size
  - Consequently, computational analyses should reveal an efficient tuning parameter setting

Experimental setup

- Workload is defined as the number of operations in the system that await processing

Prescribed inter-arrival times: $\lambda = \frac{\bar{P}}{m \cdot U}$

with:

$\bar{P}$: Mean processing time

m : Number of machines

U: Desired utilization rate

Simulation process

1. The manufacturing system consists of 6 machines.
2. Each job passes 4 to 6 machines resulting in 5 operations on average.
3. The machine order of operations within a job is generated from a uniform probability distribution.
4. The processing times of operations are uniformly distributed in the range of $[1, 19]$ which leads to a mean processing time of $\overline{P} = 5 \cdot 10$.
5. We generate exponentially distributed inter-arrival times based on λ by using various utilization rates U.

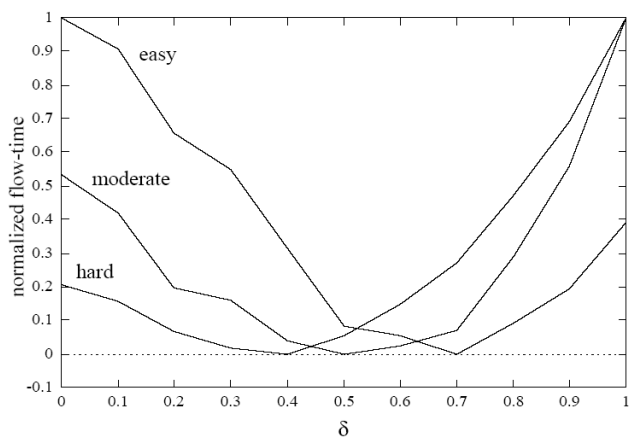
Computational results

- At first, tests for parameterization of the hybrid scheduler are executed
- For this purpose, three groups of experiments are generated
  - Easy
  - Moderate
  - Hard

Test problems – Parameter constellations

Table 1: Three cluster of test problem instances.

cluster	no. of jobs n	util. rate U
easy	10 and 20	0.5 and 0.6
moderate	30 and 40	0.7 and 0.8
hard	50 and 60	0.9 and 1.0



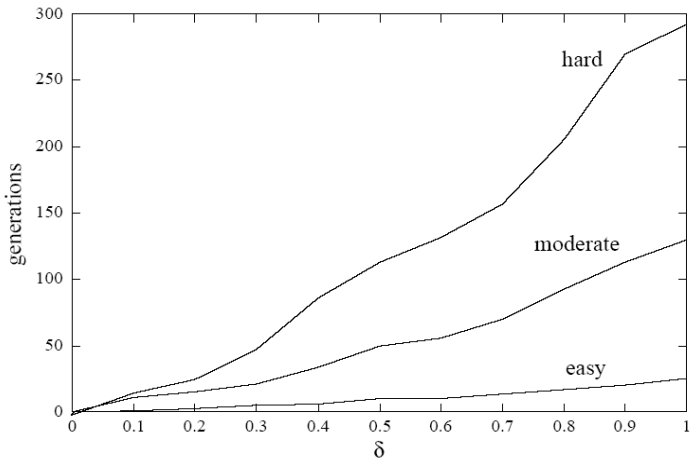


Figure 5: Influence of hybrid-decoding on GA generations.

Test 1 – Conclusions

- Easy:
  Best value for δ is 0.7
- Moderate:
  Best value for δ is 0.5
- Hard:
  Best value for δ is 0.4
- Findings:
  - Worst level of solution quality for easy problems is produced with non-delay scheduling, while active scheduling still results in an acceptable solution quality
  - For hard instances, it is just the other way round
  - Active schedulers are more time consuming

Test 1 – Conclusions

- Summarized δ is set to 0.5 as a reasonable compromise
- If computational time becomes a dramatic bottleneck for large-sized instances, it can be reduced
- Note that the corresponding drawbacks are limited since the use of non-delay schedulers still leads to quite promising results for large-sized instances

Real-time control by rescheduling

- Two versions of the hybrid GA are iteratively used to solve the intermediate static problems of the dynamic process
- Rand version
  - Starts with a randomly generated population
  - Returns the best solution found
  - Is restarted after each snapshot taken at the arrival of a new job
- Biased version
  - Obtains its initial population by modifying the last one of its preceding application according to the new constellation
  - This is done by:

1. The operations which have been started before t, including those which are still processed, are deleted in all permutations of the population.
2. The operations of new jobs are inserted at random positions in all permutations of the population respecting the job's operation order.

Biased GA rescheduling

- Job release times are updated using the “snapshot” equations
- Already started operations are neglected and erased
- Requests whose last operation is erased vanish
- Now the initial population contains fragments of favorable solutions found in the previous GA run
- Therefore, a faster convergence behavior can be expected

Real-time control experiments

- Both GA variants are applied within a dynamic scenario
- Inter-arrival times are generated by a Poisson process (see section 4.2 of the cited paper)
- To generate usable results, 50 simulations with 1,000 jobs each have been conducted
- In the SPT approach, the non-delay schedule is used, while the request scheduled next is always characterized by the smallest imminent processing time

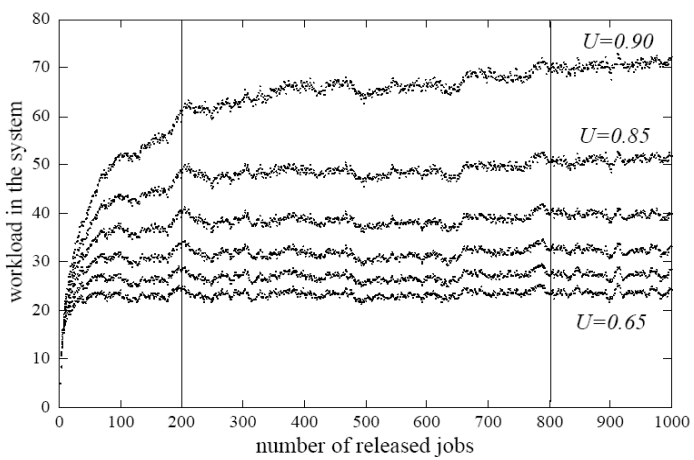


Figure 6: System workload under SPT-based control.

Conclusions

- After about 200 job releases, the system reaches a stable load level except for U=0.9
- However, the larger U becomes, the longer it takes to reach the stable level
- First jobs distort the total results as significantly as the last
- Consequently, in order to reduce distorting effects, the jobs 1-200 and 801-1,000 are neglected by the evaluation

Computational environment

- Tests are executed on a Pentium 200 MHz personal computer system
- Approaches
  - GA Rand
  - GA Bias
  - SPT
- Nothing is said concerning anticipation horizons…
- Therefore, dominance of computation

The attained solution quality

Table 2: Results of non-deterministic scheduling.

util. rate	mean flow-time $\overline{F}$			generations		runtime (sec)	
	SPT	rand	bias	rand	bias	rand	bias
0.65	91.8	85.8	85.9	10.8	8.8	111.7	81.1
0.70	100.4	93.2	93.5	14.9	11.1	189.6	127.6
0.75	111.5	103.5	103.1	21.4	14.6	351.0	214.6
0.80	128.1	118.7	117.4	33.5	20.3	747.6	407.5
0.85	154.9	142.6	140.6	56.7	31.6	1876.4	953.0
0.90	200.4	184.3	179.4	103.8	52.4	5928.1	2705.5

Results

- **SPT**
  - Flow times strongly increase with the utilization rate
  - U=0.65: Average waiting time of jobs: 42
  - U=0.9: Average waiting of jobs: 150
- **GA**
  - Flow times increase moderately with the utilization rate
  - About quadratic time complexity
  - Clearly outperform SPT
  - On average, 2.7 seconds computational time per call

Randomized versus Biased GA

- Bias GA clearly outperforms Rand GA
- In particular, if the required computational time is included in this rating
- On average, there are 73.5 operations in the system
- Biased GA generates 52.4 generations on average for U=0.9
- For **this problem size**, the required zero response time is held

Literature of Section 7

- Arthur, D. ,Vassilvitskii, S.: k-means++: The advantages of careful seeding. In the *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms,* pp. 1027-1035, 2007.
- Bieding, T., Görtz, S., Klose, A.: On-line routing per mobile phone: a case on subsequent deliveries of newspapers. In: van Nunen, J.A.E.E., Speranza, M.G., Bertazzi, L. (Eds.), Innovations in Distribution Logistics, Lecture Notes in Economics and Mathematical Systems, vol. 619. Springer, Berlin Heidelberg, pp. 29–51, 2009.
- Bierwirth, C.; Mattfeld, D.C.: Production scheduling and rescheduling with genetic algorithms. Evolutionary Computation, 7(1), 1-18, 1999.
- Bock, S. Echtzeitfähige Steuerung von Speditionsnetzwerken. DUV, Wiesbaden 2005.
- Bock, S.: Real-time control of freight forwarder transportation networks integrating multimodal transports and multiple transshipments. European Journal of Operational Research Vol.200, 733-746, 2010.
- Deutsch, P., Gailly, J.L., 1996. RFC1950: ZLIB compressed data format specification version 3.3. RFC editor United States. http://www.ietf.org/rfc/rfc1950.txt.

Literature of Section 7

- Ferrucci, F.: Pro-active dynamic Vehicle Routing. Real-time Control and Request-Forecasting Approaches to Improve Customer Service. Physica/Springer, Berlin, Heidelberg 2013.
- Ferrucci, F.; Bock, S.; Gendreau, M.: A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. European Journal of Operational Research 225(1), 130-141, 2013.
- Ferrucci, F.; Bock, S.: Real-time Control of Express Pickup and Delivery Processes in a Dynamic Environment. Transportation Research Part B Methodological Vol.63, S.1-14, Mai 2014.
- Ferrucci, F.; Bock, S.: A General Approach For Controlling Vehicle En-route Diversions in Dynamic Vehicle Routing Problems. Transportation Research Part B Methodological, Vol.77, S.76-87, 2015.
- Ferrucci, F.; Bock, S.: Pro-active real-time routing in applications with multiple request patterns. European Journal of Operational Research, Vol. 253(2), S.356-371, 2016.
- Gendreau, M., Guertin, F., Potvin, J.-Y., Taillard, E. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. Transportation Science 33, 381-390, 1999.

Literature of Section 7

- Glover, F. Tabu Search Part 1 and 2. ORSA Journal on Computing 1 (1989) pp. 190-206 and Vol. 2 (1990), 4-32, 1990.
- Glover, F., Laguna, M. Tabu Search. Kluwer Acad. Publishers, 1997.
- Hvattum, L. M., Løkketangen, A., Laporte, G. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic. Transportation Science 40 (4), 421-438, 2006.
- Hvattum, L. M., Løkketangen, A., Laporte, G. A Branch-and-Regret Heuristic for Stochastic and Dynamic Vehicle Routing Problems. Networks 49 (4), 330–340, 2007.
- Ichoua, S., Gendreau, M., Potvin, J.-Y. Diversion Issues in Real-Time Vehicle Dispatching. Transportation Science 34, 426-438, 2000.
- Ichoua, S., Gendreau, M., Potvin, J.-Y. Exploiting Knowledge About Future Demands for Real-Time Vehicle Dispatching. Transportation Science 40 (2), 211–225, 2006.

Literature of Section 7

- Kvam, P.H., Vidakovic, B.: Nonparametric Statistics with Applications to Science and Engineering. Wiley Series in Probability and Statistics. Wiley, Hoboken, NJ., 2007
- Larsen, A., Madsen, O.B., Solomon, M.M.: Partially dynamic vehicle routing models and algorithms. Journal of the Operational Research Society 53 (6), 637–646, 2002.
- Larsen, A., Madsen, O.B., Solomon, M.M.: Classification of dynamic vehicle routing systems. In: Zeimpekis, V., Giaglis, G.M., Minis, I., Tarantilis, C.D. (Eds.): Dynamic Fleet Management, Operations Research/Computer Science Interfaces Series, vol. 38. Springer Science+Business Media LLC, Boston, MA, pp. 19–40, 2007.
- MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: LeCam, L.M., Neyman, J. (Eds.), Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1. University of California Press, pp. 281–297, 1967.

Literature of Section 7

- Moon, T.K.: Error Correction Coding: Mathematical Methods and Algorithms. Wiley-Interscience, Hoboken, NJ., 2005.
- Reeves, C.R. Modern heuristic techniques for combinatorial problems. Wiley, New York, 1993.
- Séguin, R., J.-Y. Potvin, M. Gendreau, T.G. Crainic, P. Marcotte. Real-time decision problems: An operational research perspective, *Journal of the Operational Research Society* 48, 162-175, 1997.
- Van Hentenryck, P., Bent, R. Online Stochastic Combinatorial Optimization. The MIT Press 2009. ISBN:0262513471, 9780262513470.