Chapter 4

Development and Evaluation of two Dynamic Planning Procedures

In this chapter two dynamic planning approaches are developed: an Insertion based procedure with Multiple Neighborhood Search (Section 4.1) and an Assignment based procedure (Section 4.2). Both procedures are directed - as an intermediate step on the way to the actual real-life planning situation - to the local area capacitated MLPDPTW, for which a detailed problem specification has been given in Section 2.3. In Section 4.3 the procedures' specific characteristics are compared, elaborating the main differences. Afterwards, some test data sets - self-generated as well as taken from the literature - are introduced (Section 4.4). These data sets are used for a comparison of the procedures' performance and also to gain some general insights to dynamic problems (Section 4.5). Finally, one procedure is chosen for adaptation to the actual real-life scenario (Section 4.6).

4.1 Multiple Neighborhood Search

The first procedure (coded in Eclipse 3.4.2 with Java version 1.6) basically consists of two components: *Best Insertion* and *Multiple Neighborhood Search (MNS)*. Best Insertion is used (i) to construct a feasible initial solution out of the available static orders, (ii) to incorporate new dynamically occurring orders as well as (iii) a basis for the improvement procedure. The improvement part is named "Multiple Neighborhood Search" since it investigates several structurally different neighborhoods in order to find solutions with better objective function values.

Figure 4.1 visualizes the *general program framework*: During the planning horizon, new orders arrive dynamically and have to be incorporated by the planning algorithm. At first, a new feasible solution is constructed by Best Insertion, followed by a run of the MNS component. When the replanning run is finished, new instructions are sent to the vehicle fleet in operation.

4.1.1 General Planning Process and Synchronization

Since "plan execution" and "replanning" run simultaneously, some rules for *synchronization* and for the *general planning process* have to be specified. Based on the idea of **rolling horizon planning**, the following approach was chosen (cp. Figure 4.2):

Steffen Schorpp, *Dynamic Fleet Management for International Truck Transportation,*
DOI 10.1007/978-3-8349-6675-9_4,
© Gabler Verlag | Springer Fachmedien Wiesbaden GmbH 2011

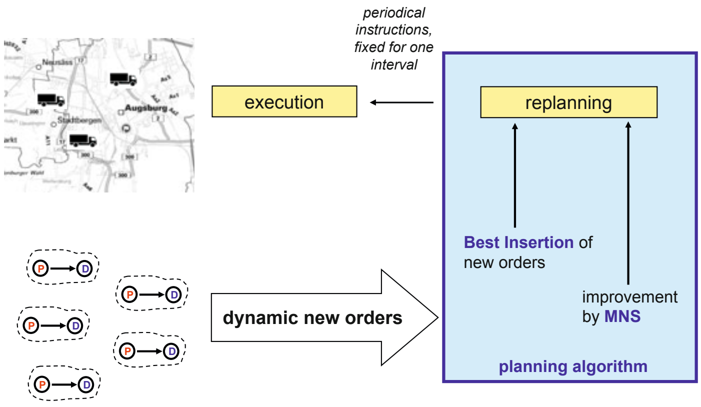

periodical instructions, fixed for one interval
execution
replanning
Best Insertion of new orders
improvement by MNS
planning algorithm
dynamic new orders
P
D

Figure 4.1: MNS: general program framework

The planning horizon is split into time intervals of equal length (here: 10 minutes). At the beginning of each interval, all decisions within the current interval are fixed. Then the current plan is transferred to the vehicles in execution, giving them planning certainty at least for the following 10 minutes. In a next step, it is checked whether new orders have arrived during the last time interval. If this is true, those newly arrived orders are incorporated by Best Insertion. In Figure 4.2, for example, there are three orders, A, B and C, that arrive in the time interval from 8:00 until 8:10. Accordingly, these orders are incorporated at the beginning of the following time interval (8:10 until 8:20) by Best Insertion.

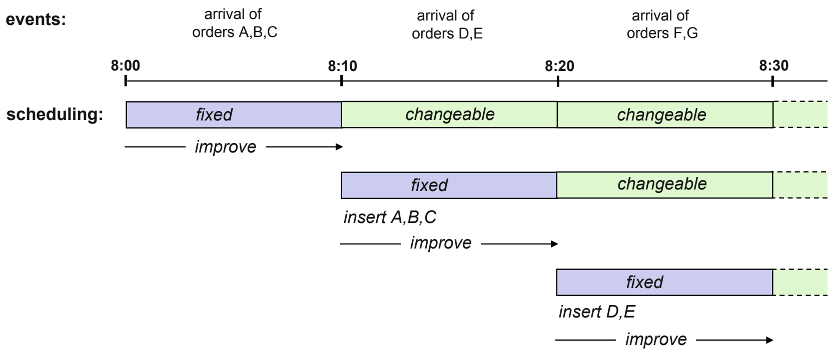


Figure 4.2: MNS: rolling horizon planning

The remaining time in the interval is completely used to run the improvement procedure. Both components, Best Insertion and MNS, have to observe the current interval's *fixed* decisions: changes in scheduling may be applied only in the subsequent intervals (denoted as *changeable*).

In the following, the term *fixed* is specified in more detail as ***time fixed*** and ***vehicle fixed***. The use of *fixed* up to now is synonymous with *time fixed*, which means a fixation of scheduled events, due to a preceding rolling horizon. In addition, some "dependent" parts of the schedule also have to be set to status *time fixed*: Let us suppose the case of time fixation of a departure event. Due to the basic problem specifications, no more re-scheduling is allowed until the associated target location is reached and serviced. Thus, the total scheduling time until servicing of this target location is completed, has to be set as *time fixed*. Therefore, in many cases the *time fixed* horizon will exceed the original 10 minutes.

An event is set to status *vehicle fixed* if it depends on irreversibly made decisions that allow for further changes in scheduling, but do not allow for the event's exchange to another vehicle. This especially covers the situation of an order's Delivery: If the associated Pickup is set as *time-fixed*, then the Delivery task is no longer allowed to be transferred to another vehicle. But nevertheless, it may be subject to re-scheduling within the current vehicle's tour.

If an event has the status *time fixed*, it is automatically *vehicle fixed*, but not vice versa. The difference of *time fixed* and *vehicle fixed* is of special interest for the improvement part.

4.1.2 Best Insertion

In a next step, the **Best Insertion strategy** is considered in detail: in the case of a newly arrived order (with Pickup and Delivery location), the program investigates for each vehicle the cost of all feasible insertion options. Finally, the best insertion option over all vehicles is chosen.

An example is given in Figure 4.3: A vehicle is traveling towards the time fixed Delivery location of order 1, hence there are no more changes allowed before arrival. Its current tour additionally includes the vehicle fixed Delivery of order 2 as well as the unfixed Pickup and Delivery of order 3. For inclusion of the new order's Pickup and Delivery locations, there are four possible positions: three between current tour locations and one at the end. Since Pickup and Delivery may be scheduled at different positions, there are 10 possible scheduling options.

Generally, there are $\frac{n \cdot (n+1)}{2}$ possible scheduling options, with n being the number of non *time fixed* positions. However, depending on vehicle capacity and order size, the number of investigated options can be considerably reduced by excluding infeasible cases.

Another important scheduling aspect is the use of a **waiting strategy** that prevents early arrivals and thus waiting at the target location. Instead, waiting time is scheduled at the current location. The vehicle departure time (and end of the waiting time) is calculated so that it exactly results in an arrival at EPT or EDT at the target location:

departure time (P) = EPT - travel time to Pickup location from current vehicle position
departure time (D) = EDT - travel time to Delivery location from current vehicle position

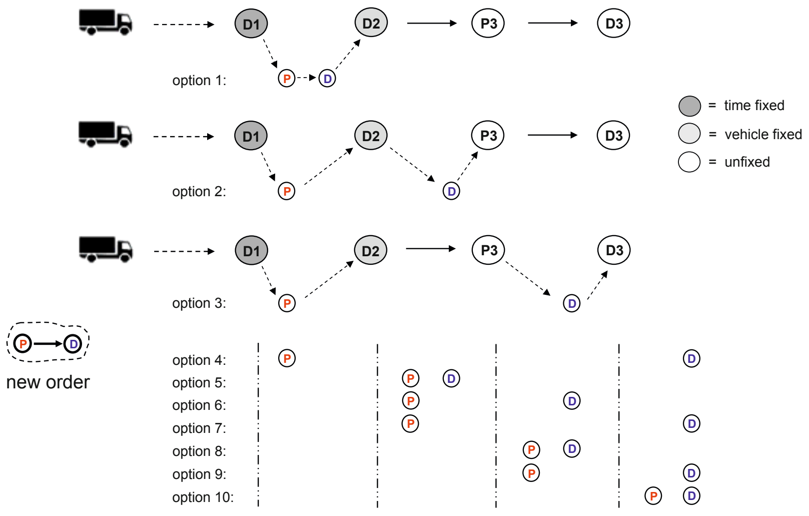


Figure 4.3: MNS: investigated insertion positions

This approach helps to postpone irreversible decisions (time fixation), hence generating more flexibility to react to new possible dynamic information (cp. Section 3.3.2). In cases in which immediate departure results in arrivals within the time window or even delayed, no waiting strategy is applied.

4.1.3 Improvement Neighborhoods

After generation of a feasible solution, the remaining time is used to run the **improvement procedure**, which applies multiple neighborhoods in an alternating manner: *λ-1 Interchange with Tabu List*, *Intraroute Optimal Sequence* and *Complete Solution Rebuild*. The frequency a specific neighborhood comes into operation has to be initially specified by percentage values.

In the following the main ideas behind these neighborhoods are outlined:

λ-1 Interchange with Tabu List selects two promising vehicle tours and investigates the complete λ-1 neighborhood (cp. Osman, 1993), which means that the advantageousness of all possible exchange operations according to the following scheme are evaluated, with the best one being finally chosen:

- an exchangeable request is extracted from each selected tour and re-inserted into the other vehicle's tour ("1 $\Leftrightarrow$ 1 exchange"),
- an exchangeable request is extracted from the first tour only and re-inserted into the second vehicle's tour ("1 $\Rightarrow$ 0 exchange"), and
- an exchangeable request is extracted from the second tour only and re-inserted into

the first vehicle's tour ("$0 \Leftarrow 1$ exchange").

An illustration is given in Figure 4.4.

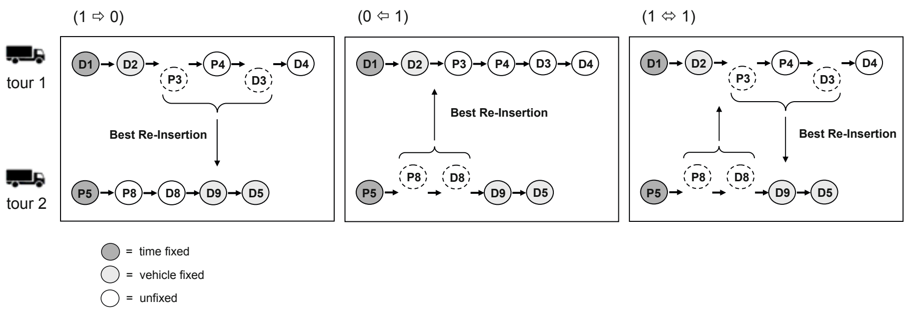
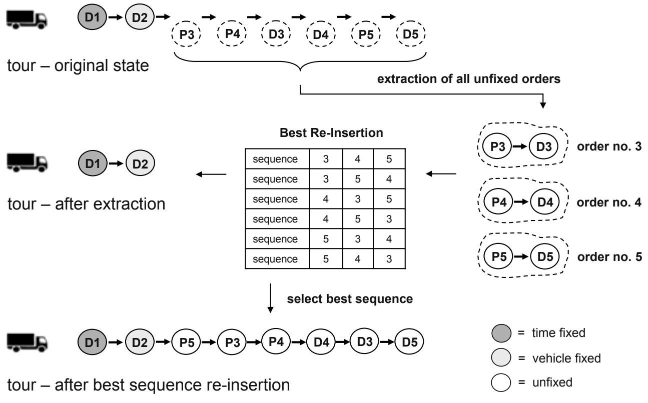


Figure 4.4: Illustration of neighborhood I: λ-1 interchange

The choice of vehicle pairs is performed in the following way: In a preprocessing step, a decreasing cost ranking for all vehicle tours is calculated. When choosing a pair of vehicle tours for exchange operations, preferably a *high-cost* and a *low-cost* vehicle are considered together. This increases the probability of achieving an improvement in objective function value by relieving the busy *high-cost* vehicle.

To avoid the recurring investigation of the same vehicle pairs, after each investigation the associated vehicle pair and the system time is stored in a Tabu List that blocks the vehicle pair for a prespecified time horizon *tabu_time*.



sequence	3	4	5
sequence	3	5	4
sequence	4	3	5
sequence	4	5	3
sequence	5	3	4
sequence	5	4	3

Figure 4.5: Illustration of neighborhood II: intraroute optimal sequence

The second neighborhood **Intraroute Optimal Sequence** extracts all exchangeable requests k within a vehicle's tour. Afterwards, these requests are re-inserted, examining all possible insertion sequences (permutations). Due to an increase in permutations with $k!$, the number of exchangeable requests must be limited: $k = 7$ turned out to be the maximum number that could be handled in acceptable computation time. The choice of vehicles is triggered again by a preprocessing step, in which all vehicles' tours are sorted according to their cost value. The procedure starts with the highest cost vehicle tour. The general idea is visualized in Figure 4.5.

The third neighborhood **Complete Solution Rebuild** does not consider only one or two vehicle tours for exchange operations, it considers all tours. In a first step, the procedure runs through all vehicle tours and extracts every exchangeable request. Afterwards, the exchangeable requests are re-inserted successively at the best insertion positions calculated over all vehicle tours. Again, in a preprocessing step, vehicle tours are ordered according to their cost values, which are used to generate a re-insertion sequence beginning with requests from expensive tours. The approach is illustrated in Figure 4.6.

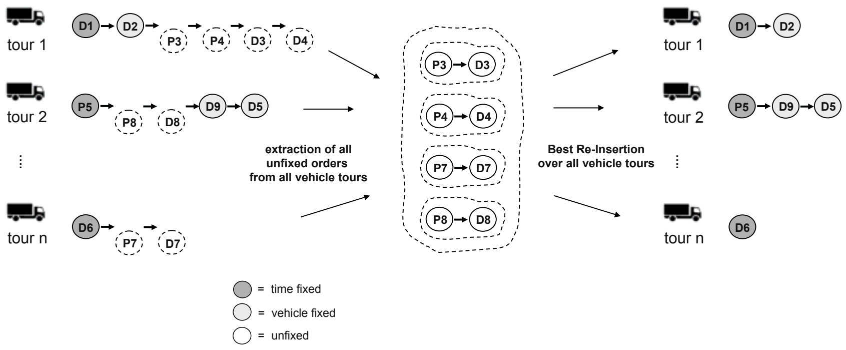


Figure 4.6: Illustration of neighborhood III: complete solution rebuild

Pseudocode notations of all three neighborhoods are given in Appendix A.

Now, all basic components and main ideas of the insertion based MNS procedure have been outlined. In the following, the focus is set on the actual implementation and on the interaction of the specific components. This can be performed best in the context of the utilized simulation framework.

4.1.4 Simulation Framework

Best Insertion and MNS improvement are embedded into a simulation framework that works as follows (cp. Program flow chart, in Figure 4.7):

In a first step, all *input data is read from an Excel file* and stored in the appropriate data classes (vehicles, orders, parameters, etc.). The Excel file contains several worksheets. The structure and contents of the most important *order* and *vehicle* worksheets are explained in the following:

- Table 4.1 shows the typical structure of an order worksheet. The first line indicates

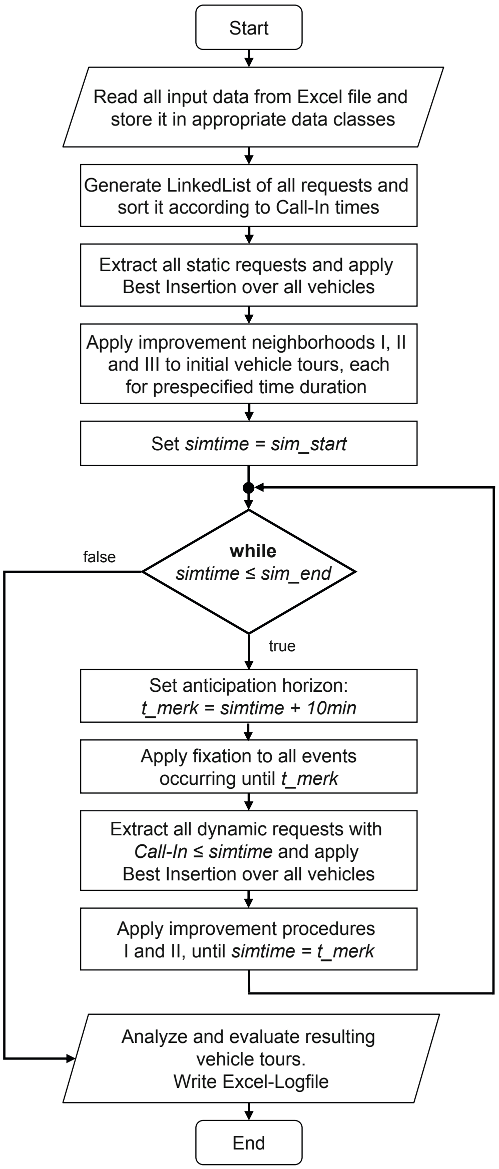

Start
Read all input data from Excel file and store it in appropriate data classes
Generate LinkedList of all requests and sort it according to Call-In times
Extract all static requests and apply Best Insertion over all vehicles
Apply improvement neighborhoods I, II and III to initial vehicle tours, each for prespecified time duration
Set *simtime* = *sim_start*
while *simtime* ≤ *sim_end*
false
true
Set anticipation horizon: *t_merk* = *simtime* + *10min*
Apply fixation to all events occurring until *t_merk*
Extract all dynamic requests with *Call-In* ≤ *simtime* and apply Best Insertion over all vehicles
Apply improvement procedures I and II, until *simtime* = *t_merk*
Analyze and evaluate resulting vehicle tours. Write Excel-Logfile
End

Figure 4.7: Program flow chart: MNS simulation framework

the total *number of orders* to be read, afterwards each line contains a unique order identifier (*no.*), followed by the order attributes *Call-In*, Pickup location (*PL*), Delivery location (*DL*), time window characteristics (*EPT*, *LPT*, *EDT*, *LDT*), required capacity (*weight*, *volume*), *loadtime*, and *unloadtime*.

number of orders:	1000									in min	in min
no.	**Call-In**	**PL**	**DL**	**EPT**	**LPT**	**EDT**	**LDT**	**weight**	**volume**	**loadtime**	**unloadtime**
1	09:00	756	356	09:30	10:15	09:45	11:15	1	1	2	2
2	09:00	989	35	09:30	10:15	09:45	11:15	1	1	2	2
3	09:00	504	99	09:30	10:15	09:45	11:15	1	1	2	2
4	09:01	18	275	09:31	10:16	09:46	11:16	1	1	2	2
5	09:02	415	535	09:32	10:17	09:47	11:17	1	1	2	2
6	09:02	474	488	09:32	10:17	09:47	11:17	1	1	2	2
7	09:03	879	264	09:33	10:18	09:48	11:18	1	1	2	2
8	09:03	766	819	09:33	10:18	09:48	11:18	1	1	2	2
9	09:03	416	161	09:33	10:18	09:48	11:18	1	1	2	2
10	09:03	781	822	09:33	10:18	09:48	11:18	1	1	2	2
11	09:04	5	546	09:34	10:19	09:49	11:19	1	1	2	2
12	09:04	80	311	09:34	10:19	09:49	11:19	1	1	2	2
13	09:05	573	968	09:35	10:20	09:50	11:20	1	1	2	2
14	09:05	320	231	09:35	10:20	09:50	11:20	1	1	2	2
15	09:06	155	442	09:36	10:21	09:51	11:21	1	1	2	2
16	09:06	208	487	09:36	10:21	09:51	11:21	1	1	2	2
17	09:06	52	399	09:36	10:21	09:51	11:21	1	1	2	2
18	09:07	181	762	09:37	10:22	09:52	11:22	1	1	2	2
19	09:07	88	10	09:37	10:22	09:52	11:22	1	1	2	2
20	09:07	585	112	09:37	10:22	09:52	11:22	1	1	2	2
21	09:09	266	729	09:39	10:24	09:54	11:24	1	1	2	2
22	09:09	697	292	09:39	10:24	09:54	11:24	1	1	2	2
23	09:09	768	742	09:39	10:24	09:54	11:24	1	1	2	2
24	09:09	650	263	09:39	10:24	09:54	11:24	1	1	2	2

Table 4.1: Excel input file: orders

- Table 4.2 visualizes the typical structure of a vehicle worksheet: the first line indicates the total *number of* (available) *vehicles*, afterwards each line contains a unique vehicle identifier (*no.*), followed by the *depot location*, the vehicle capacity with regard to weight (*cap. weight*) and volume (*cap. volume*), as well as information on vehicle availability (*available from*, *available to*).

number of vehicles:	50				
no.	**depot location**	**cap. weight**	**cap. volume**	**available from**	**available to**
1	1001	3	3	09:00	19:00
2	1001	3	3	09:00	19:00
3	1001	3	3	09:00	19:00
4	1001	3	3	09:00	19:00
5	1001	3	3	09:00	19:00
6	1001	3	3	09:00	19:00
7	1001	3	3	09:00	19:00
8	1001	3	3	09:00	19:00
9	1001	3	3	09:00	19:00
10	1001	3	3	09:00	19:00
11	1001	3	3	09:00	19:00
12	1001	3	3	09:00	19:00

Table 4.2: Excel input file: vehicles

After reading this data, in a next step all available *orders are sorted according to their Call-In time* and stored in a LinkedList. All *static requests* with Call-In time before the official start of the planning horizon *sim_start* are *extracted* from this LinkedList *and are inserted* into the best positions over all vehicles, thus generating an initial feasible plan. Then all three neighborhoods of the MNS *improvement procedure* are applied during a prespecified calculation time.

After termination of this initial phase, the *actual dynamic simulation is started*. The simulation time that is generated in the program class *clock* is set to the given initial value *sim_start* and runs with a prespecified simulation speed s.

The dynamic simulation reflects the rolling horizon planning that is visualized in Figure 4.2: According to the anticipation horizon (here: 10 minutes), the time step *t_merk* is set to *simtime + 10 minutes.* Afterwards, *fixation is applied* to all events up to *t_merk*, as well as to all dependent activities scheduled later in time. Then, all *dynamic orders* with *Call-In time* $\leq$ *simtime* are extracted from the LinkedList and are *incorporated with Best Insertion.*

Finally, the remaining time up to *t_merk* is used to apply the MNS *improvement procedure* to the new feasible solution. In some pre-tests with the test instances of Section 4.4 *Complete Solution Rebuild* required more than 10 minutes for the generation of a new feasible solution, therefore, in the dynamic program part only the improvement neighborhoods *λ-1 Interchange with Tabu List* and *Intraroute Optimal Sequence* come into operation.

The described steps are iterated until simulation time reaches the time step *sim_end*, which does not coincide with the latest Call-In time: the latest Call-In time is only a lower bound for *sim_end*, in order to make sure that all dynamic requests have been processed. However, since plan execution continues beyond that time, further simulation time results in the investigation of additional improvement options and should result in a better overall solution.

The simulation ends with a *final analysis:* the generated vehicle tours that include all static and dynamic orders are evaluated with the *results being written into an Excel file.* This file includes the following information:

- The first worksheet (cp. Table 4.3) contains some *general information* (investigated test data file, utilized computing time) and *summarized results* for the different objective function criteria (travel time, waiting time, delay, and overtime).

  In addition, the values of the objective function criteria are broken down into several sources, in order to allow for a more detailed investigation: travel time is split into *travel time to Pickup*, *travel time to Delivery* and *travel time back to depot.* Waiting time is apportioned into *waiting empty for return to depot*, *waiting empty otherwise* and *waiting loaded.* Delay is broken down into *delay at Pickup* and *delay at Delivery* locations. As further information, the percentage values of different vehicle activities during operating time are given, as well as average travel time to Pickup and Delivery locations.

- The second worksheet consists of a *detailed vehicle scheduling for each vehicle.* Table 4.4 exemplarily shows the scheduling results for a single vehicle over a planning horizon of approximately 9 hours. Vehicle activity is explained by the columns *activity-log*, *time interval* and *way.*

  In addition, for each approached location, the associated time window and the actual arrival time is given. In the following three columns, the scheduled activity times are assigned to one of the groups: *waiting*, *traveling* and *loading.* Afterwards, potential *delay* is calculated, followed by two columns including information about the vehicle's capacity status *weight* and *volume.* At the end, some supplemental information like capacity utilization and vehicle utilization over the whole simulated planning horizon – *temporal utilization (fraction of traveling and loading)* – is stated.

test data file:	3_40dyn-glv_TW45_Ab30_stat4h_OM2_ev.xls
computation time (in min):	660
total travel time (in min):	19185
travel time to Pickup (in min):	7486
travel time to Delivery (in min):	10999
travel time back to depot (in min):	700
total waiting time (in min):	3248
waiting empty for return to depot (in min):	112
waiting empty otherwise (in min):	2044
waiting loaded (in min):	1092
total delay (in min):	39
delay Pickup (in min):	19
delay Delivery (in min):	20
total overtime (in min):	0

vehicle activity:		
operating time (in min):	26433	
traveling (in min):	19185	72.57 %
waiting (in min):	3248	12.28 %
loading (in min):	4000	15.13 %
further calculations:		
avg. travel time to Pickup (in min):	7:29	
avg. travel time to Delivery (in min):	10:59	

Table 4.3: Excel output file: general summary

vehicle	activity-log	time interval	way	time window	arrival	waiting	traveling	loading	delay	weight	volume
5	waiting in location 1001	09:00:00 -- 09:28:49				28:49					
	traveling to Pickup of order no. 309	09:28:49 -- 09:37:00	1001 -- 325	09:37:00 -- 10:22:00	09:37:00		8:11	2:00		1,00	1,00
	waiting in location 325	09:39:00 -- 09:40:57				1:57					
	traveling to Pickup of order no. 50	09:40:57 -- 09:42:00	325 -- 497	09:42:00 -- 13:42:00	09:42:00		1:03	2:00		2,00	2,00
	traveling to Pickup of order no. 323	09:44:00 -- 09:57:58	497 -- 461	09:45:00 -- 10:30:00	09:57:58		13:58	2:00		3,00	3,00
	traveling to Delivery of order no. 323	09:59:58 -- 10:24:27	461 -- 8	10:00:00 -- 11:30:00	10:24:27		24:29	2:00		2,00	2,00
	waiting in location 8	10:26:27 -- 10:57:58				31:31					
	traveling to Pickup of order no. 155	10:57:58 -- 11:03:00	8 -- 530	11:03:00 -- 15:03:00	11:03:00		5:02	2:00		3,00	3,00
	traveling to Delivery of order no. 309	11:05:00 -- 11:06:47	530 -- 152	09:52:00 -- 11:22:00	11:06:47		1:47	2:00		2,00	2,00
	waiting in location 152	11:08:47 -- 11:12:31				3:44					
	traveling to Delivery of order no. 50	11:12:31 -- 11:19:22	152 -- 720	09:57:00 -- 13:57:00	11:19:22		6:51	2:00		1,00	1,00
	traveling to Pickup of order no. 161	11:21:22 -- 11:24:50	720 -- 358	11:09:00 -- 15:09:00	11:24:50		3:28	2:00		2,00	2,00
	traveling to Pickup of order no. 470	11:26:50 -- 11:31:09	358 -- 442	11:24:00 -- 12:09:00	11:31:09		4:19	2:00		3,00	3,00
	traveling to Delivery of order no. 470	11:33:09 -- 11:46:49	442 -- 28	11:39:00 -- 13:09:00	11:46:49		13:40	2:00		2,00	2,00
	traveling to Pickup of order no. 510	11:48:49 -- 11:56:38	28 -- 616	11:50:00 -- 12:35:00	11:56:38		7:49	2:00		3,00	3,00
	traveling to Delivery of order no. 155	11:58:38 -- 12:09:40	616 -- 352	11:18:00 -- 15:18:00	12:09:40		11:02	2:00		2,00	2,00
	traveling to Pickup of order no. 512	12:11:40 -- 12:15:42	352 -- 16	11:51:00 -- 12:36:00	12:15:42		4:02	2:00		3,00	3,00
	traveling to Delivery of order no. 510	12:17:42 -- 12:45:43	16 -- 440	12:05:00 -- 13:35:00	12:45:43		28:01	2:00		2,00	2,00
	traveling to Delivery of order no. 161	12:47:43 -- 12:52:24	440 -- 712	11:24:00 -- 15:24:00	12:52:24		4:41	2:00		1,00	1,00
	traveling to Pickup of order no. 597	12:54:24 -- 12:58:10	712 -- 738	12:44:00 -- 13:29:00	12:58:10		3:46	2:00		2,00	2,00
	traveling to Delivery of order no. 512	13:00:10 -- 13:03:49	738 -- 928	12:06:00 -- 13:36:00	13:03:49		3:39	2:00		1,00	1,00
	traveling to Pickup of order no. 602	13:05:49 -- 13:08:25	928 -- 897	12:46:00 -- 13:31:00	13:08:25		2:36	2:00		2,00	2,00
	traveling to Pickup of order no. 617	13:10:25 -- 13:13:17	897 -- 635	13:00:00 -- 13:45:00	13:13:17		2:52	2:00		3,00	3,00
	traveling to Delivery of order no. 602	13:15:17 -- 13:20:07	635 -- 958	13:01:00 -- 14:31:00	13:20:07		4:50	2:00		2,00	2,00
	traveling to Pickup of order no. 73	13:22:07 -- 13:24:20	958 -- 552	09:57:00 -- 13:57:00	13:24:20		2:13	2:00		3,00	3,00
	traveling to Delivery of order no. 73	13:26:20 -- 13:35:28	552 -- 383	10:12:00 -- 14:12:00	13:35:28		9:08	2:00		2,00	2,00
	traveling to Pickup of order no. 641	13:37:28 -- 13:49:50	383 -- 549	13:24:00 -- 14:09:00	13:49:50		12:22	2:00		3,00	3,00
	traveling to Delivery of order no. 597	13:51:50 -- 14:10:29	549 -- 254	12:59:00 -- 14:29:00	14:10:29		18:39	2:00		2,00	2,00
	traveling to Delivery of order no. 617	14:12:29 -- 14:23:44	254 -- 209	13:15:00 -- 14:45:00	14:23:44		11:15	2:00		1,00	1,00
	traveling to Pickup of order no. 173	14:25:44 -- 14:27:46	209 -- 63	11:21:00 -- 15:21:00	14:27:46		2:02	2:00		2,00	2,00
	traveling to Pickup of order no. 153	14:29:46 -- 14:32:41	63 -- 5	11:01:00 -- 15:01:00	14:32:41		2:55	2:00		3,00	3,00
	traveling to Delivery of order no. 173	14:34:41 -- 14:41:35	5 -- 189	11:36:00 -- 15:36:00	14:41:35		6:54	2:00		2,00	2,00
	traveling to Delivery of order no. 641	14:43:35 -- 14:55:14	189 -- 237	13:39:00 -- 15:09:00	14:55:14		11:39	2:00		1,00	1,00
	traveling to Pickup of order no. 212	14:57:14 -- 15:02:33	237 -- 685	11:53:00 -- 15:53:00	15:02:33		5:19	2:00		2,00	2,00
	traveling to Delivery of order no. 153	15:04:33 -- 15:07:24	685 -- 664	11:16:00 -- 15:16:00	15:07:24		2:51	2:00		1,00	1,00
	traveling to Pickup of order no. 748	15:09:24 -- 15:16:03	664 -- 648	14:37:00 -- 15:22:00	15:16:03		6:39	2:00		2,00	2,00
	traveling to Delivery of order no. 748	15:18:03 -- 15:19:52	648 -- 452	14:52:00 -- 16:22:00	15:19:52		1:49	2:00		1,00	1,00
	traveling to Delivery of order no. 212	15:21:52 -- 15:26:45	452 -- 704	12:08:00 -- 16:08:00	15:26:45		4:53	2:00		0,00	0,00
	waiting in location 704	15:28:45 -- 15:59:07				30:22					
	traveling to Pickup of order no. 876	15:59:07 -- 16:06:00	704 -- 854	16:06:00 -- 16:51:00	16:06:00		6:53	2:00		1,00	1,00
	traveling to Pickup of order no. 879	16:08:00 -- 16:13:38	854 -- 711	16:08:00 -- 16:53:00	16:13:38		5:38	2:00		2,00	2,00
	traveling to Delivery of order no. 879	16:15:38 -- 16:28:59	711 -- 346	16:23:00 -- 17:53:00	16:28:59		13:21	2:00		1,00	1,00
	traveling to Pickup of order no. 882	16:30:59 -- 16:39:20	346 -- 755	16:10:00 -- 16:55:00	16:39:20		8:21	2:00		2,00	2,00
	traveling to Pickup of order no. 881	16:41:20 -- 16:47:52	755 -- 25	16:09:00 -- 16:54:00	16:47:52		6:32	2:00		3,00	3,00
	traveling to Delivery of order no. 876	16:49:52 -- 16:56:49	25 -- 354	16:21:00 -- 17:51:00	16:56:49		6:57	2:00		2,00	2,00
	traveling to Pickup of order no. 901	16:58:49 -- 17:01:37	354 -- 519	16:25:00 -- 17:10:00	17:01:37		2:48	2:00		3,00	3,00
	traveling to Delivery of order no. 882	17:03:37 -- 17:10:47	519 -- 731	16:25:00 -- 17:55:00	17:10:47		7:10	2:00		2,00	2,00
	traveling to Pickup of order no. 947	17:12:47 -- 17:23:49	731 -- 746	16:56:00 -- 17:41:00	17:23:49		11:02	2:00		3,00	3,00
	traveling to Delivery of order no. 881	17:25:49 -- 17:43:50	746 -- 824	16:24:00 -- 17:54:00	17:43:50		18:01	2:00		2,00	2,00
	traveling to Delivery of order no. 901	17:45:50 -- 18:05:30	824 -- 133	16:40:00 -- 18:10:00	18:05:30		19:40	2:00		1,00	1,00
	traveling to Delivery of order no. 947	18:07:30 -- 18:24:32	133 -- 369	17:11:00 -- 18:41:00	18:24:32		17:02	2:00		0,00	0,00
	traveling back to depot location 1001	18:26:32 -- 18:37:57	369 -- 1001	-- 19:00:00	18:37:57		11:25				
						-------	-------	-------	-------		
	temporal utilization:	**83.32 %**				**96:23**	**389:34**	**92**			
	(fraction of traveling and loading)					**waiting**	**traveling**	**loading**	**delay**		

Table 4.4: Excel output file: vehicle scheduling

- The third worksheet (cp. Table 4.5) includes a view of the *planning results from an order based perspective*. For each order, some given facts, like *Call-In* time and time window information (EPT, LPT, EDT, LDT) is replicated, combined with the planning results: assigned *vehicle*, actual Pickup time (PT) and actual Delivery time (DT). In addition, the resulting delays at Pickup and Delivery location are calculated.

order no.	Call-In	vehicle	EPT	LPT	PT	delay P	EDT	LDT	DT	delay D
299	08:00	43	13:00	17:00	13:00:00		13:15	17:15	14:00:25	
300	08:00	16	13:00	17:00	14:26:47		13:15	17:15	14:55:46	
301	09:01	3	09:31	10:16	09:31:00		09:46	11:16	10:59:37	
302	09:04	14	09:34	10:19	10:17:36		09:49	11:19	11:11:22	
303	09:05	2	09:35	10:20	10:08:34		09:50	11:20	10:20:22	
304	09:05	36	09:35	10:20	09:59:37		09:50	11:20	10:05:26	
305	09:05	10	09:35	10:20	09:35:00		09:50	11:20	09:55:51	
306	09:06	13	09:36	10:21	09:36:00		09:51	11:21	10:23:52	
307	09:06	1	09:36	10:21	09:55:26		09:51	11:21	10:41:36	
308	09:06	3	09:36	10:21	10:11:50		09:51	11:21	10:37:04	
309	09:07	5	09:37	10:22	09:37:00		09:52	11:22	11:06:47	
310	09:07	4	09:37	10:22	10:17:03		09:52	11:22	10:44:18	
311	09:08	17	09:38	10:23	09:38:00		09:53	11:23	11:22:20	
312	09:09	35	09:39	10:24	10:12:21		09:54	11:24	11:00:54	
313	09:11	1	09:41	10:26	10:15:22		09:56	11:26	10:25:36	
314	09:11	19	09:41	10:26	10:10:47		09:56	11:26	11:21:35	
315	09:11	19	09:41	10:26	10:03:53		09:56	11:26	10:40:48	

Table 4.5: Excel output file: planning results from an order based perspective

Now, the general idea and the planning process of the Multiple Neighborhood Search procedure have been explained. Subsequently, a second planning approach that is based on completely different concepts is presented.

4.2 Assignment Based Procedure

The basic idea of the Assignment based procedure (coded in Eclipse 3.4.2 with Java version 1.6) was proposed in Fleischmann et al. (2004) for a local area SLPDPTW. In the following, the original approach is extended for the multi load case.

The procedure's main feature is to trigger an order-to-vehicle assignment by the result of a classical bipartite assignment problem. This allows for a simultaneous consideration of all vehicles V and all open orders O. The objective is to minimize the overall costs of carrying out all requested transportation tasks.

Every re-planning run has to be prepared in such a way that the underlying problem can be solved for an equal number of n orders and n vehicles. Since the number of vehicles and orders will usually not be identical and in order to allow for vehicle waiting and order postponement, some *dummy orders* ($o \in O^d$, denoting the set of dummy orders; with $|O^d| = |V|$) and *dummy vehicles* ($v \in V^d$, denoting the set of dummy vehicles; with $|V^d| = |O|$) are introduced.

4.2.1 Bipartite Assignment Problem

Hence, a bipartite assignment problem has to be solved over all vehicles $v \in \{V \cup V^d\}$ and all orders $o \in \{O \cup O^d\}$. The assignment costs for each order-vehicle pair are calculated as c_{vo}.

A problem formulation is given as follows:

Model of the Bipartite Assignment Problem:

data:

V	$=$	set of real vehicles
V^d	$=$	set of dummy vehicles
O	$=$	set of real orders
O^d	$=$	set of dummy orders
c_{vo}	$=$	assignment cost of vehicle v to order o

variables:

$$x_{vo} = \begin{cases} 1, & \text{if vehicle v is assigned to order o} \\ 0, & \text{otherwise} \end{cases}$$

objective function:

$$minimize \quad \sum_{v \in \{V \cup V^d\}} \sum_{o \in \{O \cup O^d\}} c_{vo}\, x_{vo}$$

s.t.

$$\sum_{v \in \{V \cup V^d\}} x_{vo} = 1 \qquad \forall\, o \in \{O \cup O^d\}$$

$$\sum_{o \in \{O \cup O^d\}} x_{vo} = 1 \qquad \forall\, v \in \{V \cup V^d\}$$

$$x_{vo} \in \{\,0\,,\,1\,\} \qquad \forall\, v \in \{V \cup V^d\}\,,\, \forall\, o \in \{O \cup O^d\}$$

The given problem formulation is recurringly solved by the exact procedure proposed in Jonker and Volgenant (1987). Even in the case of major problem sizes, the calculation time is far less than a second.

A resulting assignment of a real order to a real vehicle becomes effective immediately (in the case of a waiting vehicle) or after completion of the vehicle's current trip (in the case of a traveling vehicle) and results in a trip to the order's Pickup location. As in the MNS procedure, early arrival (before the destination's time window has opened) is prevented by scheduling of some waiting time at the current location.

The length of such a waiting time is calculated identically to the previous procedure so as to achieve exact arrival at the destination's time window opening. Therefore, the real order to real vehicle assignment (result of the bipartite assignment problem) is not fixed until the actual departure of the vehicle has been carried out, thus allowing for re-assignment during such a waiting period.

4.2.2 Assignment Matrix

The underlying assignment matrix contains four types of possible assignments:

- sector I: $v \in V \wedge o \in O^d$ *assignment of real vehicle and dummy order*

$\rightarrow$ waiting empty or execution of next scheduled Delivery (if available)

- sector II: $v \in V \wedge o \in O$ *assignment of real vehicle and real order*
  $\rightarrow$ start traveling to real order's Pickup at next time of availability
- sector III: $v \in V^d \wedge o \in O^d$ *assignment of dummy vehicle and dummy order*
  $\rightarrow$ no impact
- sector IV: $v \in V^d \wedge o \in O$ *assignment of dummy vehicle and real order*
  $\rightarrow$ postponement of real order

The matrix configuration and the associated impact of possible assignments in the different sectors is visualized in Figure 4.8. The size of the first sector is stable since the number of real vehicles and dummy orders is not subject to changes ($|V| = |O^d|$). The other sectors' sizes increase if a new order occurs, and decrease if an order is removed due to an ultimate assignment (*departure to Pickup location*). Generally, the relation $|V^d| = |O|$ has to be maintained in order to keep the total matrix quadratic. Hence, a newly occurring order does not only result in an additional "real order" column but also in an additional "dummy vehicle" row; the departure to a Pickup location does not only result in the deletion of the associated "real order" column but also in the deletion of a "dummy vehicle" row.

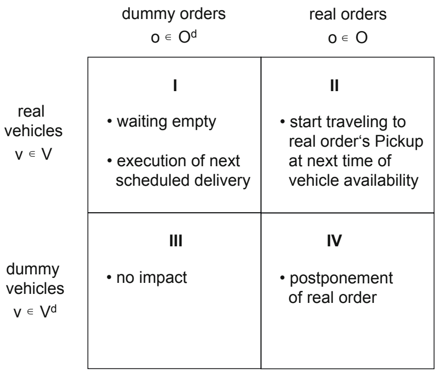

dummy orders
o ∈ O^d
real orders
o ∈ O
real vehicles
v ∈ V
dummy vehicles
v ∈ V^d
I
• waiting empty
• execution of next scheduled delivery
II
• start traveling to real order‘s Pickup at next time of vehicle availability
III
• no impact
IV
• postponement of real order

Figure 4.8: Assignment matrix: general configuration and impact of assignment

In the following, the meaning and impact of assignments in different matrix sectors is discussed in detail. Furthermore, it is explained how the respective cost values c_{vo} are chosen (cp. Figure 4.9):

Sector I

The assignment of a dummy order to a real vehicle may result in two possible effects on the real vehicle: if there are further tasks (Delivery locations) in the vehicle's current schedule, these *tasks are simply executed as planned.* Otherwise, if there are no more tasks available, the vehicle has to *wait empty* at its current location. Such an assignment occurs in situations when the number of real orders is smaller than the number of real vehicles, and also when available orders possess far distant time windows, thus being postponed.

Cost values for *further execution of scheduled Delivery tasks* are set to zero, while *waiting empty* is penalized with a given parameter *c_empty*.

	dummy orders $o \in O^d$	real orders $o \in O$
real vehicles $v \in V$	**I** cost for waiting empty 0, otherwise	**II** Best Insertion cost • travel time • delay • waiting time • overtime
dummy vehicles $v \in V^d$	**III** 0	**IV** urgency cost

Figure 4.9: Assignment matrix: choice of cost values

Sector II

Here, a real order is assigned to a real vehicle. This implies that the *real vehicle starts traveling to the Pickup location of the assigned real order* at the next time of availability. A *waiting* vehicle is available immediately, while a *traveling* vehicle reaches the status *available* for the next time when it arrives at its current destination. As already mentioned, the departure and hence the ultimate order-to-vehicle assignment may be delayed by some waiting time if immediate departure results in waiting time at the destination.

Cost values c_{vo} are chosen according to the costs that result from inserting the order o at the best position of vehicle v's scheduling. These costs are calculated with respect to the overall objective function, including weighted penalty costs for *travel time*, *delay*, *waiting*, and *overtime*.

The applied **Best Insertion** procedure is illustrated in Figure 4.10. It differs slightly from the Best Insertion applied at the MNS procedure, since it only allows for a Pickup's insertion at the first (*unfixed*) position. The associated Delivery, however, may be placed at every subsequent position as long as this complies with the capacity constraints. In Figure 4.10 there is one fixed event (arrival at Delivery 1), so the new order's Pickup is inserted right behind that Delivery. The further exemplary schedule includes three more Deliveries (D2, D4 and D3), which induces four possible insertion positions for the new Delivery: directly after the new Pickup, after D2, after D4, and after D3.

Generally, the number of investigated insertion positions is significantly lower than in the previous MNS case. In total, only $n + 1$ positions have to be checked, with n being the number of unfixed scheduled locations.

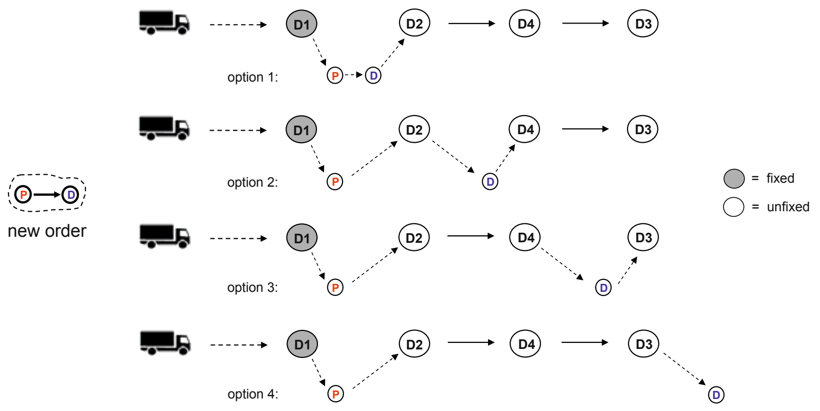


Figure 4.10: Assignment based procedure: investigated insertion positions

Sector III

The existence of this assignment type is a direct consequence of the demanded flexibility. In order to generate the three other assignment sectors for every planning run, an appropriate size of the matrix is required, including dummy columns and dummy rows at any time. The associated assignment cost are set to zero.

Sector IV

Results in this sector indicate the assignment of a dummy vehicle to a real order, which can be interpreted as the order's postponement. Such an assignment decision can have two possible reasons. On the one hand, there may be less real vehicles available than real orders. On the other hand, the order's time window may lie in the far distant future, so that a current assignment in sector II would just produce extensive waiting time.

The associated costs $c\_urgency_o$ are individually calculated for each real order o according to the following formula:

$$c\_urgency_o = \begin{cases} delta_{min,o} - 1 & \text{if } slack\_EPT_o > \text{a} \\ delta_{max,o} + (b - slack\_LPT_o)^2 & \text{if } slack\_EPT_o \leq 0 \ \& \ tw\_width_o < \text{c} \\ delta_{median,o} & \text{otherwise} \end{cases}$$

Before the meaning of these three options is explained, the calculation of the involved variables $slack\_EPT_o$, $slack\_LPT_o$, tw_width_o, $delta_{max,o}$, $delta_{min,o}$, $delta_{median,o}$ is described (a, b and c are parameters):

(i) $slack\_EPT_o = EPT_o - avg.\ travel\,time\ to\ Pickup - simtime$

(ii) $slack\_LPT_o = LPT_o - avg.\ travel\,time\ to\ Pickup - simtime$

$$\text{(iii)} \qquad tw\_width_o = \begin{cases} \frac{LPT_o - EPT_o}{avg.\,travel\,time\,to\,Pickup}, & \text{if simtime} \leq EPT_o \\ \frac{LPT_o - simtime}{avg.\,travel\,time\,to\,Pickup}, & \text{otherwise.} \end{cases}$$

$$\text{(iv)} \qquad delta_{max,o} = max\,\{c_{vo} - c_{vo'},\, v \in V\}, \qquad with\ o' = |O^d|$$

$$\text{(v)} \qquad delta_{min,o} = min\,\{c_{vo} - c_{vo'},\, v \in V\}, \qquad with\ o' = |O^d|$$

$$\text{(vi)} \qquad delta_{median,o} = median\,\{c_{vo} - c_{vo'},\, v \in V\}, \quad with\ o' = |O^d|$$

Variables (i) $slack\_EPT_o$ and (ii) $slack\_LPT_o$ measure the absolute temporal gap from current simulation time to the beginning and ending of the Pickup time window of order o, respectively. The calculation includes the average travel time to a Pickup location, since the actual travel time to the specific Pickup location cannot be calculated exactly (it depends on the vehicle to which the postponed order o will be assigned later on).

Variable (iii) tw_width_o is a measure for the remaining width of the Pickup time window, relative to the average travel time to a Pickup location. The calculation is visualized in Figure 4.11. In the first case, the current simulation time is before EPT, hence the total time window from EPT until LPT is still available. In the example, the average travel time to Pickup fits six times into the remaining time window slot ($tw\_width = 6$). In the second case, current simulation time exceeds EPT, hence the *remaining time window width* is shorter ($tw\_width = 4$).

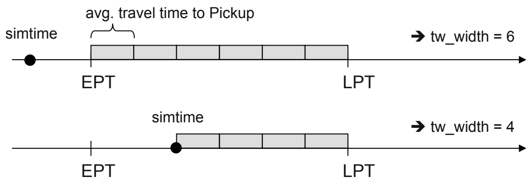


Figure 4.11: Exemplary calculation of *tw_width*

Variables (iv) and (v) are determined in order to "enforce" or to "prevent" a real order's sector II assignment, respectively. If the cost value for order o in sector IV is chosen as greater than $delta_{max,o}$, a certain assignment in sector II is enforced, at least in situations with a higher (or equal) number of real vehicles than real orders. On the other hand, if the cost value for order o is chosen as $delta_{min,o} - 1$, an assignment in sector II can be prevented with certainty. Variable (vi) $delta_{median,o}$ is calculated in order to have an intermediate value that allows "good" real order to real vehicle assignments (the better 50%) and postpones expensive assignments.

An example is given in Figure 4.12. Here, five real vehicles and two real orders are available. This results in a 7×7 - matrix. In sector I, four vehicles are assumed to have further scheduled Deliveries, hence the chosen cost values are zero. Vehicle number 2 has no further scheduled Deliveries, thus the cost values are set to *c_empty* (here: 5). Sector II shows exemplary Best Insertion cost values (e.g. cost values of 30 and 20, if real vehicle number 1 is assigned to real order number 1 and 2, respectively) and sector III includes (as specified) only zero values.

In sector IV, the first order is assumed to be *urgent*, hence the cost values are set to $delta_{max,o} + 1$ (=51), while the second order is assumed to be *not urgent* with a resulting cost value of $delta_{min,o} - 1$ (=14). The grey highlighted fields show an optimal assignment. As intended, the first real order is assigned to a real vehicle (vehicle number 5 at the cost of 10), while the second real order will be postponed (assignment of dummy vehicle number 1 at the cost of 14).

		dummy orders $o \in O^d$					real orders $o \in O$	
		1	2	3	4	5	1	2
real vehicles $v \in V$	1	**0**	0	0	0	0	30	20
	2	5	5	5	5	**5**	40	20
	3	0	**0**	0	0	0	20	60
	4	0	0	**0**	0	0	50	80
	5	0	0	0	0	0	**10**	30
dummy vehicles $v \in V^d$	1	0	0	0	0	0	51	**14**
	2	0	0	0	**0**	0	51	14

Figure 4.12: Assignment matrix with "urgent" (o=1) and "non urgent" (o=2) real order

Now, all auxiliary variables from (i) to (vi) have been defined and illustrated. Getting back to the general formula chosen for $c\_urgency_o$, three *urgency* levels can be differentiated. At the *first level*, the opening of the Pickup time window lies in the far distant future ($slack\_EPT > a$). The value of a can be interpreted as a measure in minutes and has to be chosen with a significant gap to zero: an assignment would result in more than a minutes of waiting. Therefore, such an order is considered *not urgent*, with sector IV costs being chosen *low* ($delta_{min,o} - 1$) to prevent a sector II assignment.

The *second level* depicts the situation of an *urgent* order. The time window is already open ($slack\_EPT_o \leq 0$) and the remaining time window width is smaller than c times of the average travel time to Pickup ($tw\_width_o < c$). Accordingly, the sector IV costs are chosen *high* in order to enforce a sector II assignment.

As a first term, $delta_{max,o}$ is selected, combined with a second term $(b - slack\_LPT_o)^2$. The first term enforces sector II assignments in the case of a smaller or equal number of real orders to real vehicles. It also works if the number of real orders, considered *urgent*, is smaller than or equal the number of real vehicles. The second term, depending on $slack\_LPT_o$, allows for a further differentiation of *urgent* orders if there are too many of them (no. of urgent orders > number of vehicles). The constant b is chosen as a high value in order to ensure $b - slack\_LPT_o > 0$ for all possible scenarios.

The *third level* reflects the intermediate situation between *not urgent* and *urgent*. An assignment in sector II is allowed if it belongs to the better 50% of possible assignments. The other 50% of worse assignments are blocked by the cost value $delta_{median,o}$, which instead attracts an assignment in sector IV. A definite sector II assignment, however, cannot be guaranteed, since it depends on the current number of *urgent* orders. Possibly, all "good" real vehicles are already "occupied".

So far, the configuration of all assignment matrix sectors and the associated cost calculations have been explained. Next, the following questions are answered: by what kind of events a matrix update is triggered and what sections of the matrix are affected in the specific cases.

4.2.3 Events and Matrix Updates

The Assignment based planning procedure basically knows two kinds of events that trigger a replanning run:

- first, the **occurrence of a new order,** and
- second, the event of **departure to the next Pickup or Delivery location**.

For the *first event*, an additional real order column is added to the assignment matrix, including the sector II Best Insertion cost and the sector IV postponement cost. In addition, a dummy vehicle row is added to keep the matrix quadratic (cp. Figure 4.13).

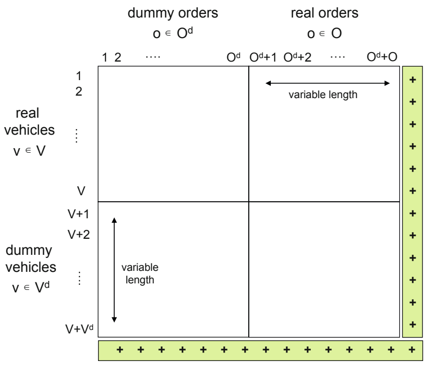


Figure 4.13: Assignment matrix update: occurrence of a new order

For the *second event*, *departure to Pickup* and *departure to Delivery* can be differentiated (cp. Figure 4.14).

In the *Pickup case*, the associated real order that is still part of the matrix must no longer be assigned to another vehicle. Hence, the real order column is completely removed from the matrix together with one row of dummy vehicles. In addition, the Pickup and the Delivery under consideration are actually inserted into the associated vehicle's scheduling, which requires a re-calculation of the complete vehicle row. Finally, all sector IV postponement values have to be updated, due to their dependency on the vehicle's row.

In the *Delivery case*, a fixation is applied to the scheduled Delivery event. This prevents any more Pickups from other open orders from being inserted before this Delivery location.

Hence, the complete vehicle row has to be re-calculated followed by an update of the sector IV postponement values.

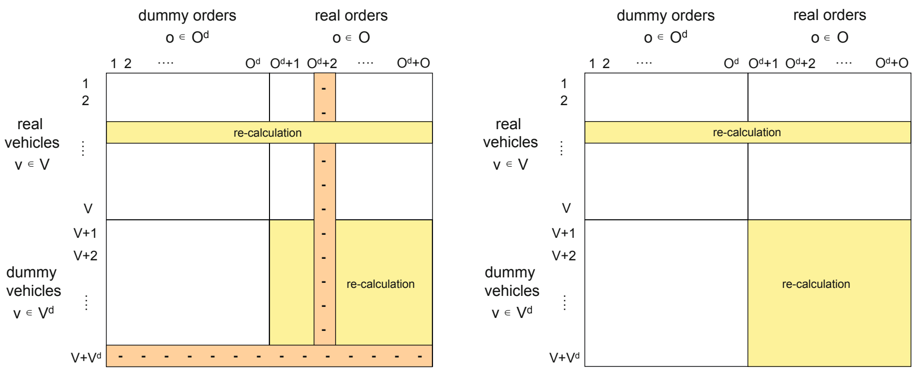


Figure 4.14: Assignment matrix update: start traveling to Pickup (left) or Delivery (right)

For better handling of the applied time discrete simulation, two additional events are defined:

- **new order-to-vehicle assignment/checkup after arrival,** and
- **end of simulation.**

These events do not trigger a matrix replanning, but help to control the simulation. A detailed description of the procedure's workflow, embedded into a time discrete simulation framework, is given subsequently.

4.2.4 Procedure Workflow and Simulation

The general procedure workflow is shown in Figure 4.15. In a first step, all input data are read from an Excel input file and stored in the appropriate data classes. The Excel file configuration is identical to the files used in Section 4.1 for the MNS procedure (see Tables 4.1 and 4.2 for typical worksheets with order and vehicle data). There are only some changes in parameter values that have to be handed to the program: instead of the anticipation horizon, now the parameters *c_empty*, *average travel time to Pickup*, *average travel time to Delivery*, and the *postponement parameters (a, b, c)* have to be specified.

Afterwards, a LinkedList, referred to as *tasklist*, is generated including all order entries with Call-In time and a simulation endtime entry. After sorting this list by increasing event time, the simulation time is set to the time of the first entry. Then all entries with current simulation time are removed from *tasklist*, being subsequently written into another LinkedList *current_events*.

Now, all events are successively removed from *current_events*, inducing different planning steps. For internal processing and sorting, each event has a specific identifier, which is chosen as a single alphabetic character ("A", "B", "C", and "D"). The events in *current_events* are sorted according to the alphabetic order of the internal identifiers, hence all "A" events are processed first, followed by the "B" events, and so on:

- **Occurrence of a new order ("A"-event):** For the new real order, the insertion costs into each real vehicle's scheduling are calculated, then the assignment matrix is updated accordingly. Afterwards, it is checked if there are further type "A"-events in *current_events*. If this is true, the same procedure is applied to the associated new orders. Otherwise, the assignment problem is solved for the updated matrix. The resulting assignments are analyzed by the method *Check Assignment results*, which will be explained later.

- **New order-to-vehicle assignment/Checkup after arrival ("B"-event):** This kind of event may be triggered by a new order-to-vehicle assignment (in such cases, the routine *Check Assignment results* has written a "B"-event for a waiting vehicle into *current_events*), and also if a vehicle arrives at a Pickup or Delivery location. In both cases, vehicles are available immediately.

  In a first query, it is checked if a sector II assignment (real order and real vehicle) is available. If this is true, it is calculated in a second query whether immediate departure would result in any waiting time at the destination. If there is no such waiting time, a type "C" event is written into *current_events*, which initiates the immediate departure to the Pickup. Otherwise, if there is any waiting time, the vehicle status is changed to *waiting*. In addition, the vehicle variables *waiting_order* and *end_wait* are updated. *End_wait* is set exactly to the departure time that results in an arrival at the destination's time window opening. Finally, a type "C" event is written into *tasklist* in order to initiate departure at the right time.

  The second branching covers the case of no available sector II assignment. It is checked if there are further scheduled activities (Deliveries) available for the concerned vehicle. If this is true, the travel time to and possible waiting at this next Delivery location is calculated. The results are handled identically to the *new order Pickup case* above: if there is no waiting time, a type "C" event is written into *current_events*, initiating the immediate departure to the Delivery. If there is waiting time, the associated vehicle attributes are updated (vehicle status = *waiting*, *waiting_order* and *end_wait*, accordingly), followed by a type "C" entry being written into *tasklist* to initiate departure at the right time.

  Finally, there is the case of a vehicle having no sector II assignment and no further scheduled activities. For such a vehicle, the status is changed to *waiting*. In addition, the vehicle attributes *waiting_order* and *end_wait* are set to the default values "-1" and "∞", respectively.

- **Start traveling to Pickup or Delivery location ("C"-event):** This kind of event is ultimately fixing a departure to a Pickup or Delivery location. It may be an immediate departure, or a departure after some waiting time.

  In a first step, it is checked if there is a current sector II assignment. If this is true, the order's Pickup and Delivery are ultimately inserted into the associated vehicle's scheduling. Then the vehicle status is set to *driving* and the attributes *waiting_order* and *end_wait* are set to the default values "-1" and "∞", respectively.

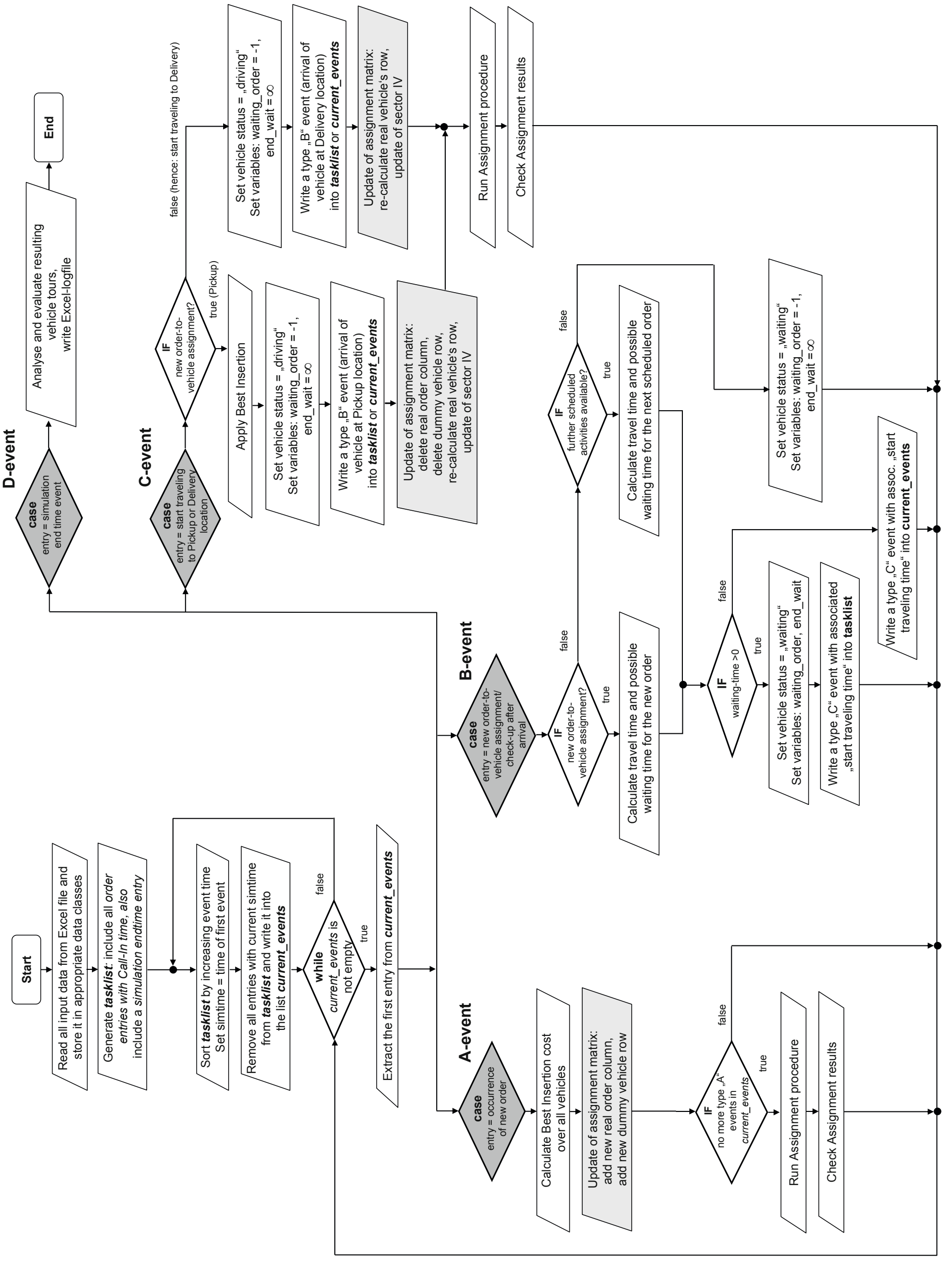

Start
Read all input data from Excel file and store it in appropriate data classes
Generate *tasklist*: include all *order entries with Call-In time, also* include a *simulation endtime entry*
Sort *tasklist* by increasing event time
Set simtime = time of first event
Remove all entries with current simtime from *tasklist* and write it into the list *current_events*
while *current_events* is not empty
true
false
Extract the first entry from *current_events*
A-event
case entry = occurrence of new order
Calculate Best Insertion cost over all vehicles
Update of assignment matrix: add new real order column, add new dummy vehicle row
IF no more type „A" events in *current_events*
true
false
Run Assignment procedure
Check Assignment results
B-event
case entry = new order-to-vehicle assignment/ check-up after arrival
IF new order-to-vehicle assignment?
true
false
Calculate travel time and possible waiting time for the new order
IF further scheduled activities available?
true
false
Calculate travel time and possible waiting time for the next scheduled order
IF waiting-time >0
true
false
Set vehicle status = „waiting"
Set variables: waiting_order, end_wait
Write a type „C" event with associated „start traveling time" into **tasklist**
Write a type „C" event with assoc. „start traveling time" into **current_events**
Set vehicle status = „waiting"
Set variables: waiting_order = -1, end_wait =∞
C-event
case entry = start traveling to Pickup or Delivery location
IF new order-to-vehicle assignment?
true (Pickup)
false (hence: start traveling to Delivery)
Apply Best Insertion
Set vehicle status = „driving"
Set variables: waiting_order = -1, end_wait =∞
Write a type „B" event (arrival of vehicle at Pickup location) into ***tasklist*** or ***current_events***
Update of assignment matrix: delete real order column, delete dummy vehicle row, re-calculate real vehicle's row, update of sector IV
Set vehicle status = „driving"
Set variables: waiting_order = -1, end_wait =∞
Write a type „B" event (arrival of vehicle at Delivery location) into ***tasklist*** or ***current_events***
Update of assignment matrix: re-calculate real vehicle's row, update of sector IV
Run Assignment procedure
Check Assignment results
D-event
case entry = simulation end time event
Analyse and evaluate resulting vehicle tours, write Excel-logfile
End

Figure 4.15: Assignment based procedure: workflow

Afterwards, a type “B” event is written into the *tasklist* with a specified event time equal to the arrival time at the order’s Pickup. In the special case of a zero travel time (Pickup location = current location), the type “B” event has to be written into *current_events* in order to ensure processing at the current simulation time. Finally, the assignment matrix is updated (as already explained above), followed by a run of the assignment procedure and an analysis of the results (method: *Check Assignment results*).

If there is no current sector II assignment, the departure to the next scheduled Delivery is initiated. Vehicle status is set to *driving* and the attributes *waiting_order* and *end_wait* are set to the default values “-1” and “∞”, respectively. Afterwards, a type “B” event is written into *tasklist* (or into *current_events*, in the case of a zero travel time) with a specified event time equal to the arrival time at the associated Delivery location. Finally, the assignment matrix is updated (as explained above), followed by a run of the assignment procedure and an analysis of the results (method: *Check Assignment results*).

- **Simulation end time event (“D”-event):** This kind of event stops the simulation. It is chosen with a sufficient time lag to the last possible Call-In time and also to the latest time windows to ensure the processing of all events. This event triggers analysis and evaluation of the resulting vehicle tours. The results are written into an Excel logfile.

  The output file has the same structure as in the MNS procedure. The first worksheet contains a *summary of results* (cp. Table 4.3), the second worksheet contains a *detailed vehicle scheduling for each vehicle* (cp. Table 4.4), and the third worksheet shows the *planning results from an order based perspective* (cp. Table 4.5).

Now, the method **Check Assignment results** is considered in more detail. It is called each time when a planning run of the solution procedure for the bipartite assignment problem is finished. Its main objective is to analyze the planning results for *waiting* vehicles that could be immediately affected by new assignment decisions.

Vehicles with status *driving*, however, are not considered, since they cannot be immediately affected by the planning results. Current assignment decisions concerning those (driving) vehicles may be updated several times until their next time of availability (arrival at Pickup or Delivery location). Therefore, an early consideration does not make any sense.

The pseudocode of the method *Check Assignment results* is given in Table 4.6.
Basically, three options are checked for each waiting vehicle:

- **Case 1:** *A new real order is assigned to the real vehicle and the vehicle is not waiting for another task.*

  Here, a “B” type event is written into *current_events* to check the new order-to-vehicle assignment and to initiate potential departure.

- **Case 2:** *A new real order is assigned to the real vehicle and the vehicle is waiting for a different task (departure to a new real order’s Pickup or to a Delivery).*

```
For (int i=1, i $\leq$ number of real vehicles, i++) {
    If (vehicle_status(i) == ''waiting'') {
        If (real order is assigned to vehicle(i)) {
            If (assigned order(i) $\neq$ ''waiting_order(i)'') {
                //hence: there was none or a different assignment before
                Write a ''B''-type event into current_events
                If (''end_wait'' $\neq \infty$) { //hence: there was a different assignment before
                    Remove obsolete ''C''-type entry from tasklist or current_events
                    Set variables: ''waiting_order(i)'' = -1 and ''end_wait(i)'' = $\infty$
                }
            }
        }
        Else { //hence: no real order is currently assigned to vehicle i
            If (''waiting_order(i)'' $\neq$ -1 & scheduled task = Pickup) {
                //hence: there was an assignment before
                Remove obsolete ''C''-type entry from tasklist or current_events
                Set variables: ''waiting_order(i)'' = -1 and ''end_wait(i)'' = $\infty$
                If (further scheduled activities (Deliveries) are available for vehicle i) {
                    Apply re-scheduling for vehicle i, based on current simtime
                    Write a ''B''-type event into current_events
                }
            }
        }
    }
}
```

Table 4.6: Pseudocode of method *Check Assignment results*

Here, also a “B” type event is written into *current_events* to check the new order-to-vehicle assignment and to initiate potential departure. In addition, an obsolete “C” type event from the formerly scheduled task (end of the waiting time to arrive punctually at the old order’s Pickup or Delivery location) has to be removed from *tasklist* or *current_events*. The vehicle attributes *waiting_order* and *end_wait* are reset to the default values “-1” and “∞”, respectively.

- **Case 3:** *No real order is assigned to the real vehicle, but there was a real order assignment before.*

  Here, in a first step the obsolete “C” type event from the formerly assigned real order has to be removed from *tasklist* or *current_events*. In addition, the vehicle attributes *waiting_order* and *end_wait* are reset to the default values “-1” and “∞”, respectively.

  Afterwards, it is checked if there are further scheduled Delivery events available for the considered vehicle. If this is true, a re-scheduling is applied based on the current simulation time. Then a type “B” event is written into *current_events* to initiate potential departure to the next Delivery location.

In the following section, a comparison of both procedures that have been introduced in Section 4.1 and 4.2 is carried out.

4.3 A Comparison of Both Procedures

Both procedures, MNS and the Assignment based procedure, can be characterized according to the **configuration framework for dynamic algorithms** that was proposed in Section 2.1. The particular attributes are summarized in Tables 4.7 and 4.8 for MNS and the Assignment based procedure, respectively.

The *first procedure* uses a *classical Best Insertion algorithm combined with improvement measures making use of multiple neighborhoods* as **technique of adjustment**. The *second procedure* applies an *optimal bipartite assignment algorithm* that triggers order-to-vehicle assignment. For calculation of the order-to-vehicle assignment costs, procedure one uses a classical Best Insertion technique that investigates all possible insertion options. In contrast, the second procedure applies a *specific Best Insertion variant* that only allows for the insertion of a Pickup at the first scheduling position.

Insertion based procedure with Multiple Neighborhood Search	
technique of adjustment	- *classical Best Insertion + Multiple Neighborhood Search*
reaction of adjustment	- *constructive,* updating former planning results
frequency of adjustment	- *time driven* replanning
duration of adjustment	- *time limit based* with fixed anticipation horizon
synchronisation of adjustment	- *extensive simultaneity* of plan execution and planning
scope of adjustment	- *restricted,* not all real-life options are allowed

Table 4.7: Characteristics of the first planning approach, according to the *Dynamic Algorithm Configuration Framework*

Assignment Based Procedure	
technique of adjustment	- *optimal bipartite assignment*, based on specific Best Insertion
reaction of adjustment	- *from scratch*
frequency of adjustment	- *event-based* replanning
duration of adjustment	- *zero time*
synchronisation of adjustment	- *extensive simultaneity* of plan execution and planning
scope of adjustment	- *restricted,* not all real-life options are allowed

Table 4.8: Characteristics of the second planning approach, according to the *Dynamic Algorithm Configuration Framework*

The **reaction of adjustment** is developed as follows. In the *first procedure*, new dynamic information is incorporated in a *constructive* way, simply updating the results of the last planning run and not discarding the old solution. The *second procedure* actually performs a *from scratch* re-planning. A new dynamic information may result in a complete change of all prior assignment decisions as long as these decisions have not yet been ultimately *fixed.*

The next characterizing attribute is the **frequency of adjustment**. In the *first procedure*, a new replanning run is triggered periodically by elapsed time (*time-based*). Thus, the time available to the algorithm for inclusion of new information and for improvement operations is known to be at a prespecified constant level, equal to the *anticipation horizon* **(duration of adjustment)**.

The *second procedure*, however, performs an *event-based* replanning technique. This ensures immediate reaction, but, on the other hand, does not guarantee predictable information on the time available for replanning runs. Consequently, an algorithm is chosen that ensures a *zero-time* duration of adjustment.

Synchronization of adjustment is performed in a similar way in both approaches. *Plan execution* and *replanning* run *simultaneously*, with the algorithm being allowed to change all decisions which are not *fixed.* In the *first approach*, an anticipation horizon, which has to be chosen as > 0 time units, is introduced and triggers *fixation* of all events that are due in the horizon, as well as fixation of all dependent events. The associated rolling horizon approach produces planning certainty for the vehicles in execution and defines a clear field of changeable scheduling options to the planning algorithm, thus keeping plan in execution and plan updates consistent.

The *second approach* works without such an anticipation horizon. Since replanning time is near zero, it is assumed that a decision can be executed immediately. *Fixation* is not applied in order to achieve simultaneity, but only in the case of a departure event that induces ultimate *fixation* to parts of the scheduling.

Since technically possible real-life options like "en route diversion" or "transshipment" are not supported in *both procedures*, the **scope of adjustment** must be described as *restricted* twice.

It can be concluded that both procedures use quite differing concepts to cope with dynamic information. In order to prove and compare the performance of these planning approaches, some suitable test data sets are needed.

4.4 Test Data

In the following, two test data sets for the dynamic MLPDP with soft time windows are presented. The first one is *self-generated* with the help of an own data generator and covers the originally intended "capacitated" dynamic MLPDPTW (real-life application: Dial-A-Ride services). The second data set is adopted *from the literature* and was chosen because of its good documentation and availability of detailed results. This data set covers the slightly differing case of an "uncapacitated" dynamic MLPDPTW (real-life application: Express Mail Delivery). Both procedures from Sections 4.1 and 4.2 are capable of solving the uncapacitated problem version by simply setting each vehicle's capacity to infinity.

4.4.1 Self-Generated Test Scenarios

The self-generated data sets (partially based on Ivankina, 2004) basically consist of three components: an underlying *node network*, *dynamic and static orders* and *available vehicles*. Subsequently, different specifications of these three components are presented, which can be combined in all possible ways. A specific selection (node network, order characteristic, vehicle characteristic) is handed as input information to a data generator, which produces the desired test instance.

Node Network

Two node networks are available (cp. Figure 4.16). Each network contains 1000 possible customer locations characterized by an x- and y- coordinate. The distance between these nodes is calculated using the Euclidean metric. Vehicles are located at a central depot with coordinates (x=0, y=0).

- **network 1** is a circle with a radius of 15km. Within the circle, there is a randomly located cluster, covering 20% of total circle area and containing 40% of all nodes.

The remaining 60% of nodes are equally distributed across the remaining circle area.

- **network 2** is also circle shaped, with a radius of 15km. However, the number of randomly located clusters is increased to three, with two clusters covering 10% of the circle area and incorporating 25% of nodes each. The third cluster covers 5% of the circle area and possesses 20% of all nodes. Again, the remaining nodes are equally distributed across the cluster free circle area.

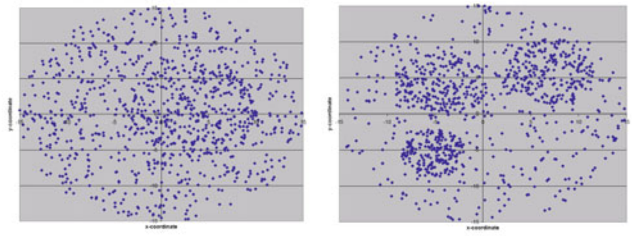


Figure 4.16: Node network 1 (left) and node network 2 (right) - distribution of customers

Static and dynamic orders

The second choice concerns the order characteristic. For every instance, 1000 orders are generated by randomly choosing Pickup and Delivery locations out of the available network nodes. Each order is specified to consume one unit of available vehicle capacity. Furthermore, at every Pickup and Delivery location, a loading/unloading time of 2 minutes has to be scheduled.

In the following, some basic settings for the dynamic and static order generation are explained:

The dynamic orders' Call-In times are chosen as equally distributed in the time interval [09:00, 17:00] with a 30-minute time gap between Call-In and EPT. Hence, the latest time window opening may be scheduled at 17:30 (for an order with Call-In at 17:00). The Pickup time window has a width of 45 minutes (LPT = EPT + 45 min), the Delivery time window opens 15 minutes after EPT and has a width of 90 minutes (LDT = EDT + 90 min). This time window characteristic is denoted (45,15,90).

The static orders are assumed to be known before 9:00. Here, EPT is chosen to be equally distributed in the interval [09:00, 13:00]. In contrast to the dynamic requests, the time windows of static orders have a width of 4 hours, for both, Pickup and Delivery. As in the dynamic case, the Delivery time window opens 15 minutes after EPT. This time window characteristic is denoted (240,15,240)

These basic setting can be changed as follows:

- The *time gap between Call-In and EPT* may be varied. The basic setting with a time gap of 30 minutes is supplemented by five additional options: 0 min, 5 min, 10 min, 15 min, 20 min, and 25 min.

- The *time window width of dynamic orders* may be varied. The basic setting of (45,15,90) is complemented by the options (15,15,30), (30,15,60) and (60,15,120).

In addition, the total share of dynamic and static orders has to be specified. Here, varying degrees of dynamism between 0% and 100% can be chosen in steps of 10%.

Vehicles

At first, some basic assumptions, which are not subject to changes, are explained. Each vehicle starts and ends its tour at the depot that is specified by the underlying network. Regular operating time is defined as in the interval [09:00, 19:00]; afterwards penalty costs for overtime are charged. It is assumed that a vehicle drives at an average speed of 30km/h.

There are two settings that can be changed:

- The *number of vehicles* may be varied in the interval between 42 and 50.
- The *vehicle capacity* may be varied between 1 and 7 units.

With the specified characteristics, the data generator is capable of producing 38 808 different instances (combinations of networks, order and vehicle characteristics). Table 4.9 shows *six main scenarios* and the associated number of instances that have been chosen for detailed consideration in Section 4.5.

Scenarios 1 and 2 allow for an investigation of the impact of varying degrees of dynamism on solution quality. In addition, it can be analyzed whether the use of the different underlying networks 1 and 2 causes any variations in solution quality. In scenario 3, the reaction time that is given to the procedure in order to handle new information is successively decreased. Scenario 4 focuses on the impact of reduced and increased time window opening time. In scenario 5, the number of vehicles is successively reduced.

Finally, scenario 6 allows for an analysis of the two procedures' behavior in the case of varying vehicle capacity. This is of special interest for a comparison with the second test scenario from the literature, which (as already mentioned) deals with the uncapacitated problem version.

4.4.2 Benchmark Data from the Literature

The chosen data set created by Gendreau et al. (2006) is of special interest, since the authors do not only publish average benchmark results, but also detailed objective function values for each of the 15 instances. In the following, the generation process of the instances is explained in detail. Afterwards an analysis is conducted in order to compare the specific characteristics of the data set scenarios. Finally the benchmark results, which were reported by Gendreau et al. (2006) are listed.

The authors generate a 5km×5km unit square as underlying geographical area with depot location at point (2.0km, 2.5km). The area is divided into 4×5 rectangular zones, each

scenario 1 (11 instances)

node network:	1
order characteristics:	*variation of the degree of dynamism from 0% up to 100%,* dynamic time window characteristic (45,15,90), time gap Call-In to EPT 30 min
vehicle characteristics:	50 vehicles, capacity 2

scenario 2 (11 instances)

node network:	2
order characteristics:	*variation of the degree of dynamism from 0% up to 100%,* dynamic time window characteristic (45,15,90), time gap Call-In to EPT 30 min
vehicle characteristics:	50 vehicles, capacity 2

scenario 3 (7 instances)

node network:	2
order characteristics:	degree of dynamism 100%, dynamic time window characteristic (45,15,90), *variation of the time gap between Call-In and EPT*
vehicle characteristics:	50 vehicles, capacity 2

scenario 4 (4 instances)

node network:	2
order characteristics:	degree of dynamism 100%, *variation of dynamic time window characteristic,* time gap Call-In to EPT 30 min
vehicle characteristics:	50 vehicles, capacity 2

scenario 5 (9 instances)

node network:	1
order characteristics:	degree of dynamism 100%, dynamic time window characteristic (45,15,90), time gap Call-In to EPT 30 min
vehicle characteristics:	*variation of number of vehicles from 42 to 50*, capacity 3

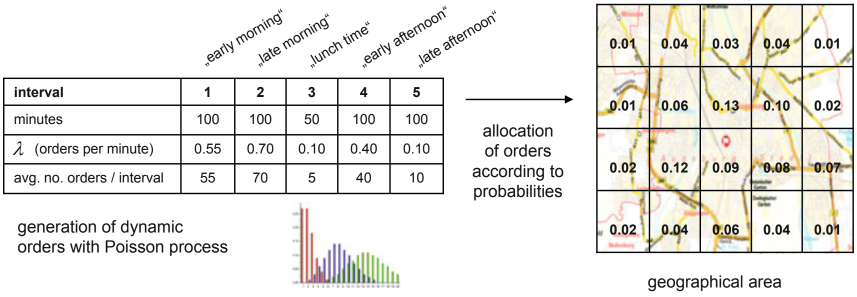
scenario 6 (7 instances)

node network:	2
order characteristics:	degree of dynamism 100%, dynamic time window characteristic (45,15,90), time gap Call-In to EPT 30 min
vehicle characteristics:	50 vehicles, *variation of vehicle capacity from 1 to 7*

Table 4.9: Investigated test scenarios generated with own data generator

possessing a specific probability between 0.01 and 0.13. A fixed number of vehicles (10 or 20) is available at the depot. Those vehicles move with a constant average speed of 30 km/h and have to return to the depot at the end of their trips. Distances within the unit square are calculated with the Euclidean metric.

For generation of dynamic requests (cp. Figure 4.17), the selected planning horizon (450 min or 240 min) is divided into five time periods: early morning, late morning, lunch time, early afternoon, and late afternoon. The lunch time period is half the length of the others, which are of equal length. Thus, for a 450 min planning horizon, a lunch time period has a duration of 50 minutes, while the other time periods have a duration of 100 minutes.



interval	1	2	3	4	5
minutes	100	100	50	100	100
λ (orders per minute)	0.55	0.70	0.10	0.40	0.10
avg. no. orders / interval	55	70	5	40	10

Figure 4.17: Generation process of dynamic requests

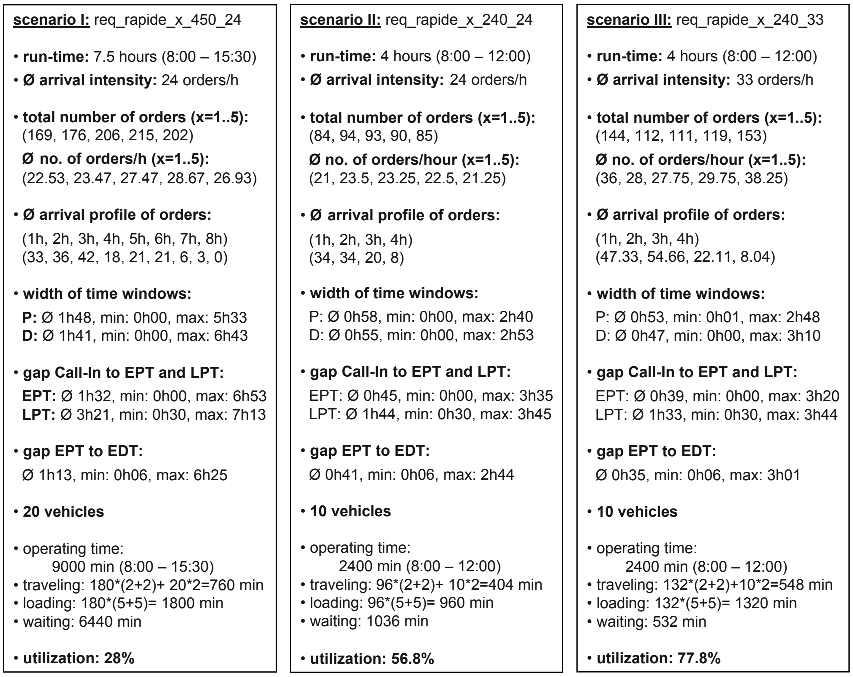
The arrival of new requests is modeled with a Poisson process, having different arrival intensities for the respective time periods. Generally, two sets of intensity parameters λ (requests per minute) are available: (0.75, 1.10, 0.25, 0.40, 0.10) and (0.55, 0.70, 0.10, 0.40, 0.10), which lead on average to 33 and 24 requests per hour, respectively. Each time a new request occurs, its Pickup and Delivery locations are allocated to one of the zones of the geographical area according to the associated probabilities.[9]

In addition, a service time of five minutes has to be scheduled at each location.

Under these basic conditions, the authors create three test scenarios *with increasing stress* for the planning procedure:

- A 450-minute horizon with arrival intensity of 24 requests per hour and a vehicle fleet size of 20 (“req_rapide_x_450_24”),
- a 240-minute horizon with arrival intensity of 24 requests per hour and a vehicle fleet size of 10 (“req_rapide_x_240_24”), and
- a 240-minute horizon with arrival intensity of 33 requests per hour and a vehicle fleet size of 10 (“req_rapide_x_240_33”).

[9] The determination process of exact locations within the zones is not specified in the publication.



scenario I: req_rapide_x_450_24	**scenario II:** req_rapide_x_240_24	**scenario III:** req_rapide_x_240_33
• **run-time:** 7.5 hours (8:00 – 15:30)	• **run-time:** 4 hours (8:00 – 12:00)	• **run-time:** 4 hours (8:00 – 12:00)
• **Ø arrival intensity:** 24 orders/h	• **Ø arrival intensity:** 24 orders/h	• **Ø arrival intensity:** 33 orders/h
• **total number of orders (x=1..5):** (169, 176, 206, 215, 202)	• **total number of orders (x=1..5):** (84, 94, 93, 90, 85)	• **total number of orders (x=1..5):** (144, 112, 111, 119, 153)
Ø no. of orders/h (x=1..5): (22.53, 23.47, 27.47, 28.67, 26.93)	**Ø no. of orders/hour (x=1..5):** (21, 23.5, 23.25, 22.5, 21.25)	**Ø no. of orders/hour (x=1..5):** (36, 28, 27.75, 29.75, 38.25)
• **Ø arrival profile of orders:** (1h, 2h, 3h, 4h, 5h, 6h, 7h, 8h) (33, 36, 42, 18, 21, 21, 6, 3, 0)	• **Ø arrival profile of orders:** (1h, 2h, 3h, 4h) (34, 34, 20, 8)	• **Ø arrival profile of orders:** (1h, 2h, 3h, 4h) (47.33, 54.66, 22.11, 8.04)
• **width of time windows:** **P:** Ø 1h48, min: 0h00, max: 5h33 **D:** Ø 1h41, min: 0h00, max: 6h43	• **width of time windows:** P: Ø 0h58, min: 0h00, max: 2h40 D: Ø 0h55, min: 0h00, max: 2h53	• **width of time windows:** P: Ø 0h53, min: 0h01, max: 2h48 D: Ø 0h47, min: 0h00, max: 3h10
• **gap Call-In to EPT and LPT:** **EPT:** Ø 1h32, min: 0h00, max: 6h53 **LPT:** Ø 3h21, min: 0h30, max: 7h13	• **gap Call-In to EPT and LPT:** EPT: Ø 0h45, min: 0h00, max: 3h35 LPT: Ø 1h44, min: 0h30, max: 3h45	• **gap Call-In to EPT and LPT:** EPT: Ø 0h39, min: 0h00, max: 3h20 LPT: Ø 1h33, min: 0h30, max: 3h44
• **gap EPT to EDT:** Ø 1h13, min: 0h06, max: 6h25	• **gap EPT to EDT:** Ø 0h41, min: 0h06, max: 2h44	• **gap EPT to EDT:** Ø 0h35, min: 0h06, max: 3h01
• **20 vehicles**	• **10 vehicles**	• **10 vehicles**
• operating time: 9000 min (8:00 – 15:30)	• operating time: 2400 min (8:00 – 12:00)	• operating time: 2400 min (8:00 – 12:00)
• traveling: 180*(2+2)+ 20*2=760 min	• traveling: 96*(2+2)+ 10*2=404 min	• traveling: 132*(2+2)+10*2=548 min
• loading: 180*(5+5)= 1800 min	• loading: 96*(5+5)= 960 min	• loading: 132*(5+5)= 1320 min
• waiting: 6440 min	• waiting: 1036 min	• waiting: 532 min
• **utilization: 28%**	• **utilization: 56.8%**	• **utilization: 77.8%**

Figure 4.18: Characteristic of three test scenarios, each including five instances (x=1...5)

For each of these scenarios, five instances are generated (x=1...5), resulting in an overall number of 15 dynamic instances. Figure 4.18 shows a detailed analysis of the associated data sets. The first column scenario entries are exemplarily explained in the following.

At the beginning, general information on the *planning horizon* (7.5 hours) and on the *average order arrival intensity* (24 orders per hour) is repeated. Afterwards, the *total number of orders* and the resulting *average number of orders per hour* are given for the five instances. Due to stochasticity within the generation process, the numbers differ slightly from instance to instance. In a next step, an average *order arrival profile* is shown for all instances: e.g. indicating that in the third hour of the planning horizon, a peak of 42 orders is assumed to arrive, whereas there are only 3 expected orders in the seventh hour.

The *average width of the time windows* is analyzed to be 1h48min and 1h41min for Pickup and Delivery, respectively. In addition, a range in form of minimum and maximum values is given for both, Pickup and Delivery time window widths. Another interesting point is the *gap between Call-In and the Pickup time window (EPT and LPT)*, which is listed subsequently. While average *reaction time* (LPT - Call-In) is 3h21min, the analysis also shows the possibility of a very short *reaction time* of 30 minutes. Furthermore, the *gap between EPT and EDT* is specified with an average length of 1h13min, a minimum value of 0h06min and maximum value of 6h25min.

In the next row, the number of *available vehicles* (20) is listed. This is followed by a calculation of the *approximate utilization* if all requests were served in regular operating time:

The total regular operating time of 20 vehicles (each available for 7.5 hours) is 9000 minutes. The approximated total travel time is 760 minutes and consists of two minutes for each trip to one of the 180 Pickup locations, of two minutes for each trip to the 180 Delivery locations and of two minutes for each of the vehicles to return to the depot at the end of the day.[10] The approximated total travel times[11] and the loading times (1800 min) are subtracted from the total regular operating time, which results in a vehicle waiting or idle time of 6440 minutes. This induces an approximate vehicle utilization of 28%.

The increasing stress from scenario one up to three can be easily reconstructed by increasing utilization values of 56.8% and 77.8%, in scenarios 2 and 3, respectively.

The solution approach that was applied to the proposed test data sets (Parallel Tabu Search with Adaptive Memory) has already been explained in Section 3.3.1.2. The objective function includes *travel time*, *delay* and *overtime*, which are weighted equally. The achieved benchmark results for all 15 instances are listed in Table 4.10: The first column shows the declaration of the associated instance, the subsequent columns include objective function values for travel time, delay and overtime (in minutes). These values are taken as benchmarks.

req_rapide_x_450_24	**travel time**	**delay**	**overtime**
x = 1	539 min	1 min	0 min
x = 2	614 min	3 min	1 min
x = 3	629 min	2 min	1 min
x = 4	700 min	6 min	5 min
x = 5	694 min	0 min	0 min

req_rapide_x_240_24	**travel time**	**delay**	**overtime**
x = 1	336 min	65 min	55 min
x = 2	386 min	68 min	53 min
x = 3	352 min	139 min	98 min
x = 4	359 min	38 min	31 min
x = 5	348 min	75 min	52 min

req_rapide_x_240_33	**travel time**	**delay**	**overtime**
x = 1	473 min	4392 min	699 min
x = 2	402 min	867 min	297 min
x = 3	455 min	853 min	303 min
x = 4	434 min	1104 min	348 min
x = 5	495 min	4121 min	687 min

Table 4.10: Benchmark results for the 15 test instances of Gendreau et al. (2006)

[10] The average travel times to Pickup locations, to Delivery locations and back to the depot are based on an ex-post analysis of the planning results that have been achieved with the procedures from Sections 4.1 and 4.2.

[11] The calculation of total travel time also includes empty travel times, e.g. the trip to the first Pickup location or the last trip back to the depot. The ex-post analysis of the planning results shows that most of the trips to Pickup locations are, however, performed by already loaded vehicles.

4.5 Computational Results and Performance Analysis

This section contains computational results for the self-generated data set and the data set from the literature. The first subsection 4.5.1, which includes the test results for the self-generated data sets, focuses on the relative comparison of both introduced dynamic procedures for varying dynamic test scenarios. The second subsection 4.5.2, which includes the test results for the data sets from the literature, does not only allow for a relative comparison of both procedures, but also for a comparison with other benchmark results. Finally, it is analyzed if the differences of the applied data sets (capacitated vehicles, in the case of the self-generated instances and uncapacitated vehicles, in the case of the instances from the literature) lead to major differences in the procedures' planning performance.

4.5.1 Computational Results for Self-Generated Test Scenarios

In Section 4.4.1, six main scenarios with different variations were generated. All associated instances are solved with *MNS* and with the *Assignment based procedure* on a desktop PC (Intel Core 2 CPU, 2.40 Ghz, 3 GB RAM). MNS simulations are run with simulation speed s = 1 (real time), for the Assignment based procedure an event based simulation is applied. In the following, the results of both procedures are visualized in three diagrams per scenario, which include the three objective function criteria: travel time, delay and overtime.

Table 4.11 shows the parameter and penalty cost settings that are chosen for MNS and for the Assignment based procedure. Parameterization has been accomplished for a basic scenario consisting of node network 2, 100% dynamic requests, time window characteristic (45,15,90), and a vehicle capacity of 2.

MNS		**Assignment**	
internal parameters		**internal parameters**	
initial improvement		avg. time to Pickup:	8 min
duration:	60 min	avg. time to Delivery:	11 min
neighborhoods I:II:III	1:1:1	matrix calculations	
general improvement:		a:	150
neighborhoods I:II	2:1	b:	600
tabu time:	30 min	c:	3
anticipation:	3 min	c_empty:	10 000
penalty costs		**penalty costs**	
c_traveling (per min):	90	c_traveling (per min):	300
c_delay (per min):	10 000	c_delay (per min):	1200
c_wait (per min):	0	c_wait (per min):	60
c_overtime (per min):	10 000	c_overtime (per min):	1200

Table 4.11: Parameter settings for self-generated test data sets

Figures 4.19, 4.20, and 4.21 exemplarily show the parameterization of the MNS anticipation horizon. The anticipation horizon is varied between 1 minute and 10 minutes in steps of 1 minute. Figure 4.19 illustrates the resulting travel times, Figure 4.20 includes

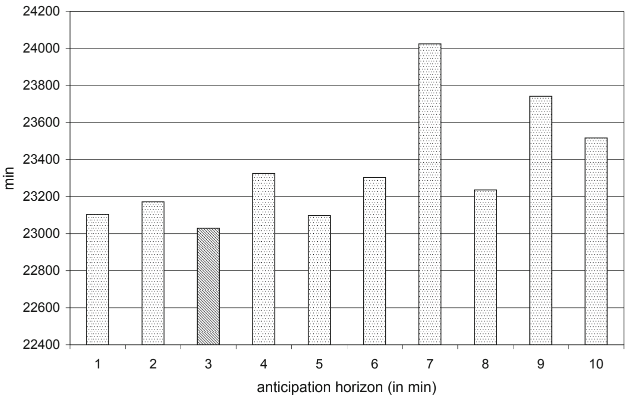


Figure 4.19: Travel time for diff. anticip. horizons (100% dyn. orders, netw. 2, cap. 2)

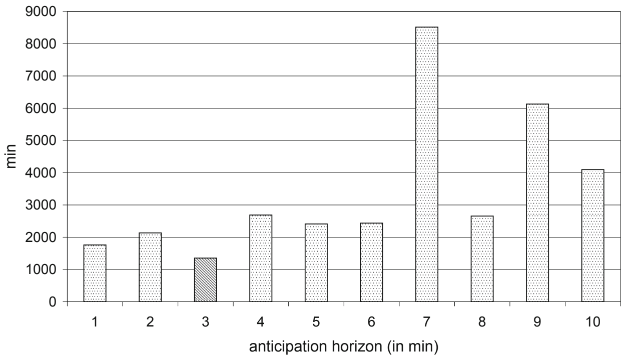


Figure 4.20: Delay for diff. anticip. horizons (100% dyn. orders, netw. 2, cap. 2)

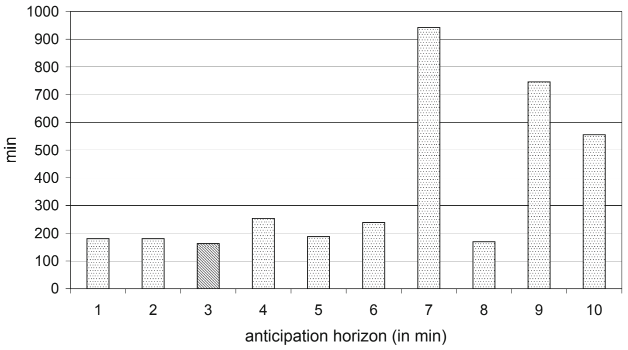


Figure 4.21: Overtime for diff. anticip. horizons (100% dyn. orders, netw. 2, cap. 2)

the resulting delay, and Figure 4.21 contains the resulting overtime. Best results in all three categories are achieved with an anticipation horizon of 3 minutes. Therefore, this parameter setting is chosen for the subsequent tests.

The other parameters are determined in analogical manner for the same basic scenario. At this point, the question of whether it is sufficient to test all instances with the same parameter settings may arise. Especially, in the case of a low degree of dynamism, other parameter settings may perform better. However, since these tests are focused on a relative comparison of both proposed dynamic procedures, such an individual parameterization for every instance is not necessary. On the contrary, results for varying data set characteristics, based on the basic parameterization, allow for additional insights into the procedures' robustness.

Scenarios 1 and 2 investigate the procedures' performance for varying degrees of dynamism and different underlying node networks. The first column of Figure 4.22 shows the results of the objective function criteria travel time, delay and overtime for network 1. The second column of Figure 4.22 illustrates the results for network 2.

In the first case (network 1), in eight of eleven instances (exceptions: degree of dynamism 70%, 80% and 90%) the *Assignment based procedure* generates significantly better results in travel time (-3.8% on average) and delay (-47.5% on average). Overtime remains at a relatively low level for both procedures. Due to an outlyer at a degree of dynamism of 80%, the average overtime of the *Assignment based procedure* is worse compared to *MNS* (+32.4%). In the second case (network 2), the situation is quite similar: for all degrees of dynamism, except 90%, the *Assignment based procedure* performs better. In comparison with *MNS*, an average reduction in travel time (-5.5%), delay (-53.1%) and overtime (-12.7%) can be achieved.

The choice of the underlying node network induces some variations in the objective function values; the general conclusion of a better performing *Assignment based procedure*, however, is identical for both node networks. Therefore, the choice of different node networks (within the range of the available test data) is not assumed to generate significant changes to the overall conclusions.

A general behavior of solution quality in dependency of the degree of dynamism cannot be observed. The intuitive assumption would be a better solution quality with decreasing degree of dynamism, since there is more information available at an earlier time.

There may be several reasons for not discovering such a behavior. A dynamic algorithm is specialized to run on a dynamic instance, thus its performance on a more or less static instance may be worse. In addition, there is the question of the appropriate parameterization of the dynamic algorithms when applying them on static instances. The parameterization for a degree of dynamism of 100% may be not the best choice for a low degree of dynamism.

Another aspect could be related to the workload that has to be handled by the algorithm. In the static case, there is a huge initial workload with a huge solution space. In the dynamic cases, however, the information is revealed little by little, which reduces the

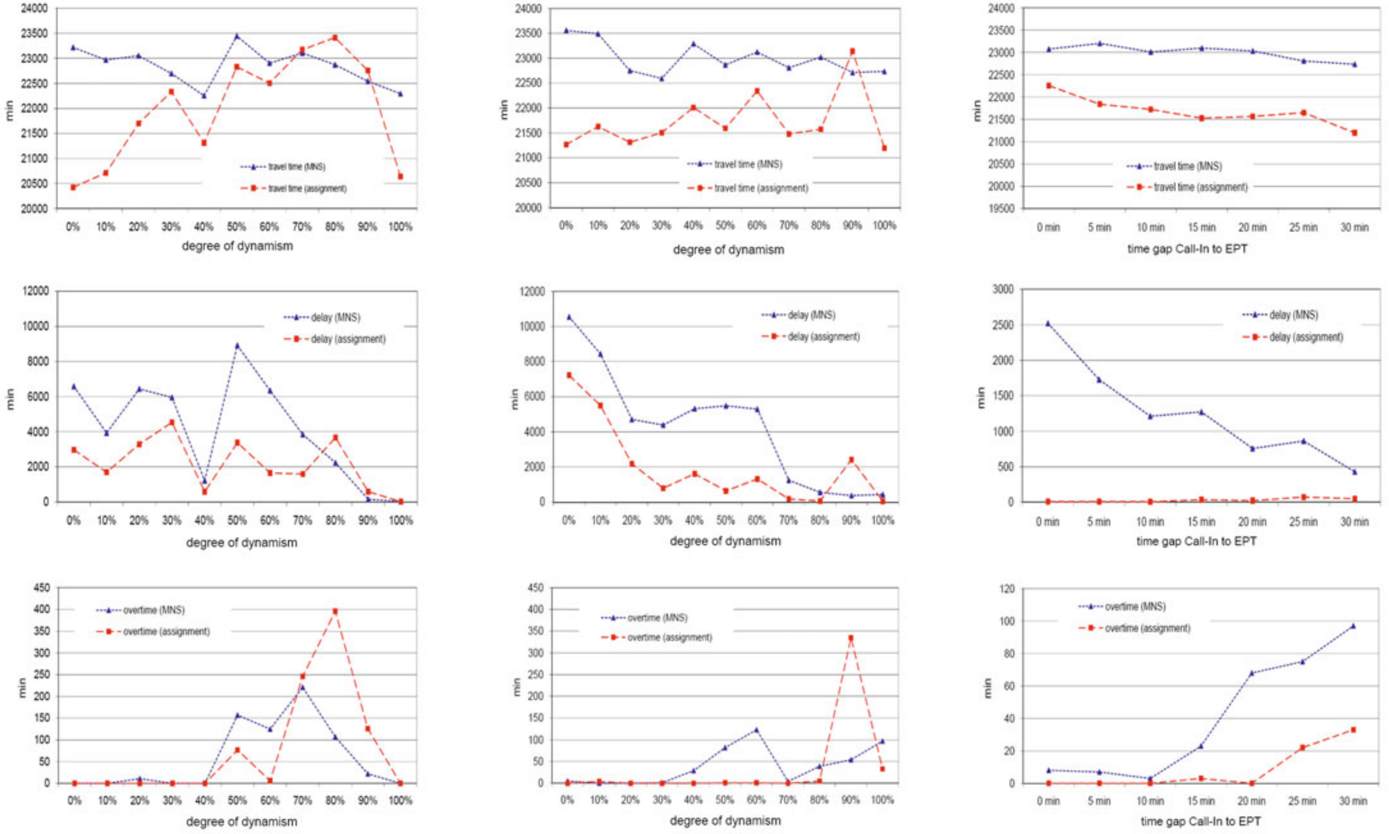


Figure 4.22:
Column 1: travel time/delay/overtime for varying degrees of dynamism (netw. 1, cap. 2)
Column 2: travel time/delay/overtime for varying degrees of dynamism (netw. 2, cap. 2)
Column 3: travel time/delay/overtime for varying gap between Call-In and EPT (100% dyn. orders, netw. 2, cap. 2)

possible alternatives. In this way, the revealment of dynamic information may "guide" the dynamic algorithm, by retaining some currently unimportant information. Similar experiences have been noted by other authors. Larsen et al. (2004) find "high variability over the entire dod spectrum" and conclude that "lower dod problems are harder to solve, because they involve larger instances than their higher dod counterparts".

In **scenario 3**, the time gap between Call-In and Earliest Pickup Time (EPT) is varied between 0 minutes and 30 minutes, in steps of 5 minutes. The underlying basic scenario contains the nodes of network 2, vehicles with a capacity of 2, the time window characteristic (45,15,90), and 100% dynamic customers.

The results are shown in the third column of Figure 4.22. Travel time decreases slightly for both procedures when the time gap is increased. The delay of the *MNS* procedure can be reduced significantly when a higher time gap is chosen. The delay of the *Assignment based procedure* remains on a permanently low level. The improved results which are achieved with longer time gaps are plausible, since there is a longer reaction time available to the algorithm. This especially benefits the improvement process of *MNS*.

Reductions in travel time and delay come along with a small increase in overtime for both procedures. This negative effect, however, is outweighed by far through the improvements in the other objective function criteria, and can be attributed to an internal trade off that accepts a small worsening in overtime in order to achieve significant better results for travel time and delay. In all instances of scenario 3, the better performance of the *Assignment based procedure* is beyond question.

Scenario 4 deals with a variation of the time window characteristic. The original characteristic (45,15,90) is changed to the shorter time window characteristics (15,15,30) and (30,15,60), as well as to the longer time window characteristic (60,15,120). As further settings, node network 2, vehicles with capacity 2 and 100% dynamic customers are chosen.

By trend, it can be observed that smaller time windows result in an increase in travel time, delay and overtime (cp. Figure 4.23, column 1). When comparing the performance of both dynamic procedures, an interesting behavior can be found. While the *Assignment based procedure* produces better results for the two "longer" time window characteristics, the picture changes for the two "shorter" characteristics. Here, the application of *MNS* produces better results in all three objective function criteria.

In **scenario 5**, the impact of a reduced number of available vehicles is investigated. Starting at the original number of 50 vehicles, the number is reduced successively to 42. As further settings, node network 1, vehicles with capacity 3, time window characteristic (45,15,90), and 100% dynamic customers are chosen.

The *Assignment based procedure's* advantage in solution quality persists up to a level of 46 available vehicles (cp. Figure 4.23, column 2). Afterwards, for a smaller number of vehicles, *MNS* achieves better solution quality. For a number of 44 down to 42 vehicles, there is a dramatic worsening of the *Assignment* results in delay and overtime. An increase of delay and overtime can also be recognized for *MNS*, but with a much more moderate rise.

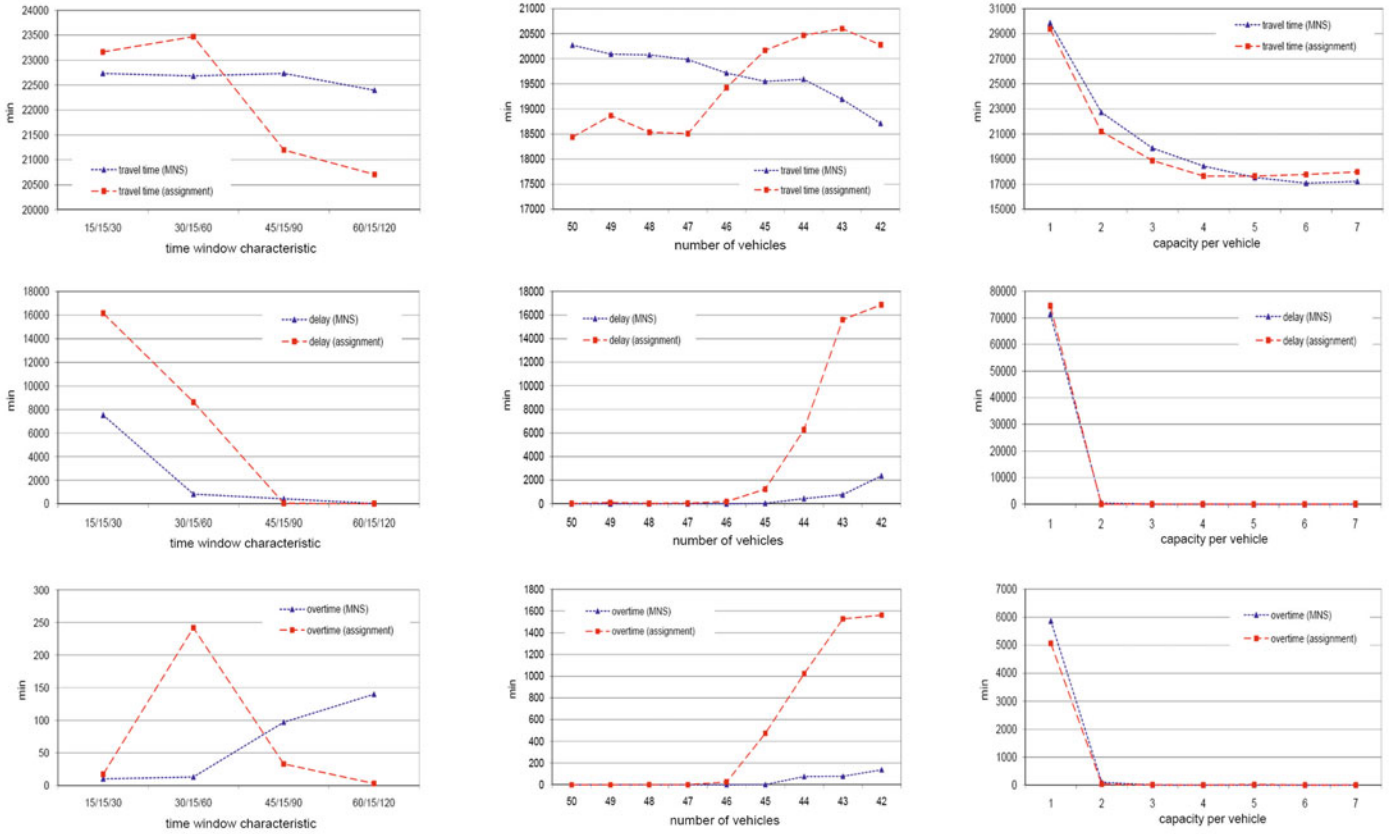


Figure 4.23:
Column 1: travel time/delay/overtime for varying time window characteristics (100% dyn. orders, netw. 2, cap. 2)
Column 2: travel time/delay/overtime for varying number of vehicles (100% dyn. orders, etw. 1, cap. 3)
Column 3: travel time/delay/overtime for varying vehicle capacity (100% dyn. orders, netw. 2)

For the travel time criterion, an antithetic behavior of the procedures' results can be observed. With the reduction of the number of vehicles, the travel time generated by the *Assignment based procedure* shows an increasing trend. Travel time generated by *MNS*, however, shrinks with a decrease of available vehicles.

The first behavior is the intuitive one. However, there are some reasons the behavior of the *MNS* results can be attributed to:

- With a sufficient number of vehicles, an urgent order is preferably transported by an immediately available vehicle to ensure on time arrival. For this purpose, some extra travel time may be accepted. However, if there is a situation of many *urgent* orders, minimization of travel time may become the decisive aspect. Suppose two orders, whose immediate execution results in an identically high level of penalty costs for delay. MNS will schedule these orders according to travel time criteria, since there is no more possible differentiation based on the urgency costs.

- There is also a second reason for the decrease in travel time: if the number of vehicles is reduced, there is also a direct reduction in total travel time, since less vehicles have to return from their last tour position back to the depot.

Generally, it can be concluded that *MNS* has the better capability to cope with a situation of a scarce number of vehicles.

Finally, in **scenario 6**, the vehicle capacity is varied in the interval from 1 to 7 (cp. Figure 4.23, column 3). As further settings, node network 2, vehicles with capacity 2, time window characteristic (45,15,90), and 100% dynamic customers are chosen.

Considering the travel time results, a better performance of the *Assignment based procedure* can be observed at the capacity levels from 1 to 4; afterwards, at capacity levels from 5 to 7, *MNS* produces better results. Delay and overtime results of *Assignment* and *MNS* are not very different. From capacity levels 2 up to 7, the results are nearly identical. Only at a capacity level of 1 there is a small variation: while *MNS* achieves a better result in category delay, the *Assignment based procedure* produces a smaller overtime.

In summary, better results for the self-generated test data sets are achieved with the *Assignment based procedure*. Nevertheless, in some scenarios (with major deviations from the parameterized basic scenario), the *Assignment based procedure* shows a less robust behavior with a significant decline in objective function value. Especially in the situation of vehicle scarcity, the results in delay and overtime reach inacceptable levels.

When compared to the *Assignment based procedure*, *MNS* generates inferior results on average. However, for some degrees of dynamism (around 90%), the produced results are competitive or even better. In addition, a good adaptability to changing conditions is exhibited: especially in situations of short time windows, scarcity of vehicles, and also for less capacity restricted situations proper results can be observed.

4.5.2 Computational Results for Benchmark Data from the Literature

In Section 4.4.2, the three test scenarios published by Gendreau et al. (2006) are presented and analyzed in detail. All associated 15 instances are solved with MNS and with the Assignment based procedure on a desktop PC (Intel Core 2 CPU, 2.40 Ghz, 3 GB RAM). MNS simulations are run with simulation speed s = 1 (realtime). In the following, the results of both procedures are visualized in three diagrams per scenario, which include the three objective function criteria: travel time, delay and overtime.

Table 4.12 shows the parameter and penalty cost settings that are chosen for MNS and for the Assignment based procedure. Parameterization was accomplished for scenario req_rapide_1_240_33.

MNS		**Assignment**	
internal parameters		**internal parameters**	
initial improvement		avg. time to Pickup:	2 min
duration:	60 min	avg. time to Delivery:	2 min
neighborhoods I:II:III	1:1:1	matrix calculations	
general improvement:		a:	150
neighborhoods I:II	2:1	b:	600
tabu time:	30 min	c:	3
anticipation:	1 min	c_empty:	10 000
penalty costs		**penalty costs**	
c_traveling (per min):	90	c_traveling (per min):	100
c_delay (per min):	10 000	c_delay (per min):	300
c_wait (per min):	0	c_wait (per min):	3
c_overtime (per min):	10 000	c_overtime (per min):	3

Table 4.12: Parameter settings for test data sets from the literature

The results for the first **scenario req_rapide_x_450_24** with the lowest stress level are visualized in Figures 4.24, 4.25 and 4.26. In all five instances (x = 1..5), a better performance of *MNS*, compared with the *Assignment based procedure*, can be observed: on average, travel time is reduced by 20%, delay by 68% and overtime by 59%.

In comparison with the *benchmark* results from literature, however, the results generated by *MNS* are outperformed itself. Especially in travel time, the benchmark results are on average 11% better than MNS. At least in categories delay and overtime, MNS achieves three improvements of the benchmark results: for instances x = 1 and 2, delay and overtime are reduced to zero; for instance x = 4, the benchmark result is undercut by 50% in delay and by 60% in overtime.

The results for the second **scenario req_rapide_x_240_24** - with medium stress level - are visualized in Figures 4.27, 4.28 and 4.29. A comparison of *MNS* and the *Assignment based procedure* again results in significant advantages of MNS: on average, travel time is reduced by 15%, delay by 38% and overtime by 29%. The Assignment based procedure only achieves slightly better delay and overtime values in one instance (x = 5); travel time, however, remains on a definitely inferior level.

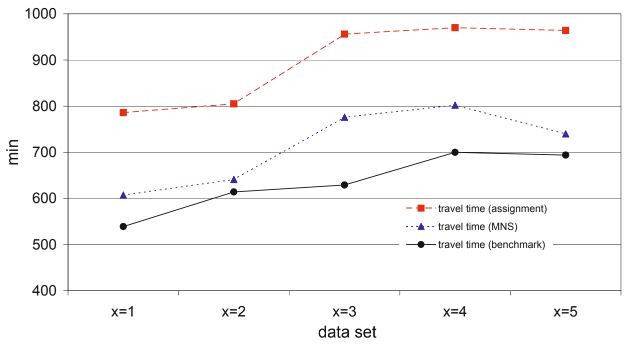


Figure 4.24: Travel time for test scenarios *req_rapide_x_450_24*

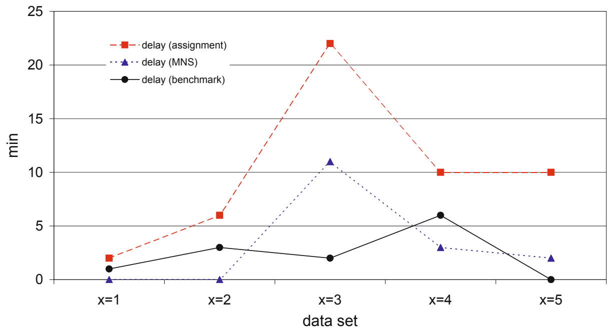


Figure 4.25: Delay for test scenarios *req_rapide_x_450_24*

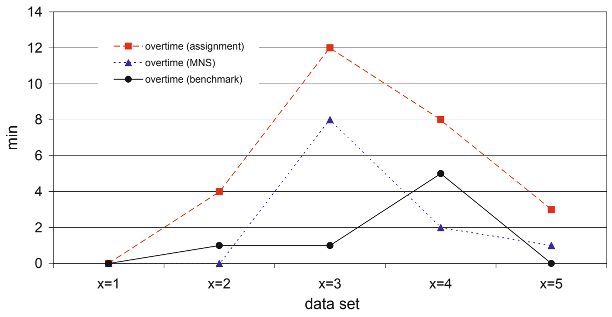


Figure 4.26: Overtime for test scenarios *req_rapide_x_450_24*

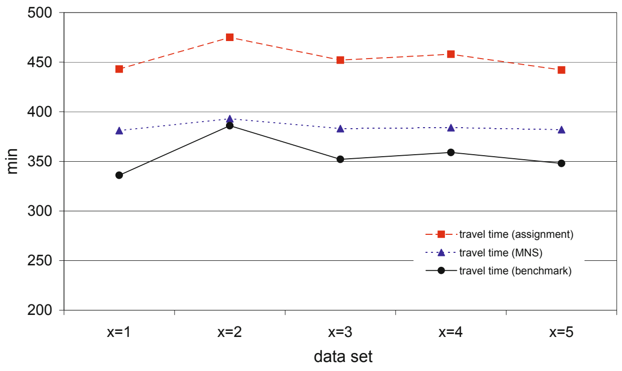


Figure 4.27: Travel time for test scenarios *req_rapide_x_240_24*

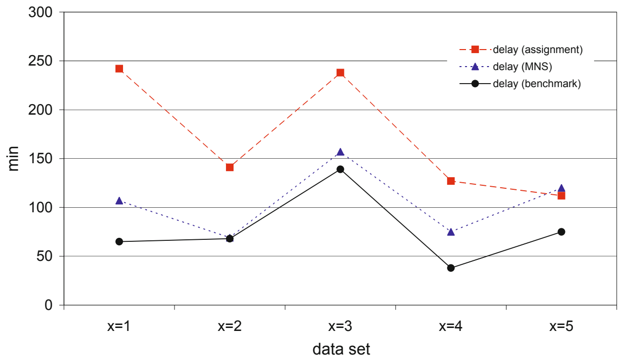


Figure 4.28: Delay for test scenarios *req_rapide_x_240_24*

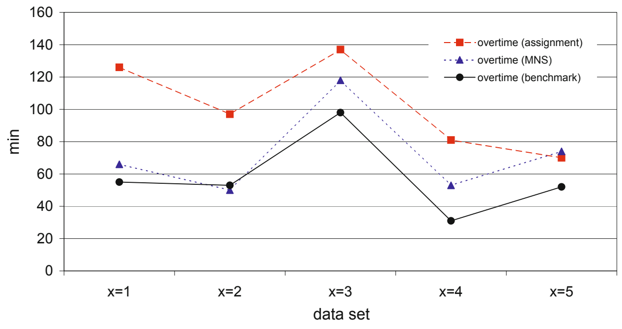


Figure 4.29: Overtime for test scenarios *req_rapide_x_240_24*

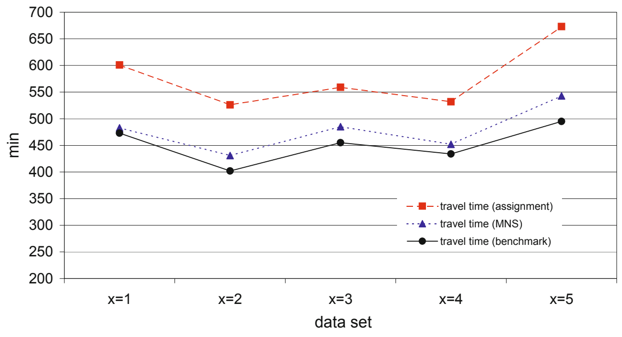


Figure 4.30: Travel time for test scenarios *req_rapide_x_240_33*

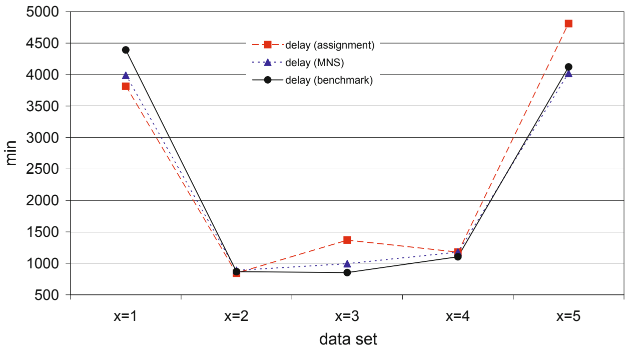


Figure 4.31: Delay for test scenarios *req_rapide_x_240_33*

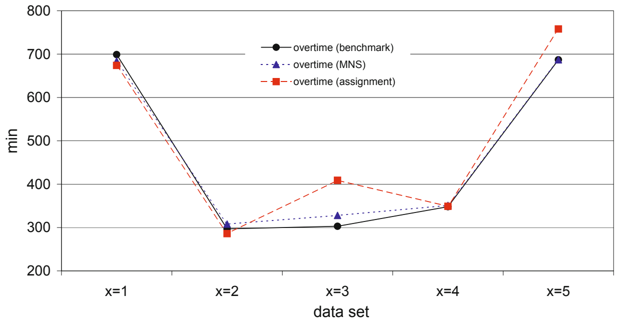


Figure 4.32: Overtime for test scenarios *req_rapide_x_240_33*

As in the first scenario, the *benchmark* results cannot be reached on average. The travel times reported by Gendreau et al. (2006) are still 7% better than the *MNS* results. For instance x = 2, the MNS results for travel time, delay and overtime are at least on a par with the benchmark.

The last **scenario req_rapide_x_240_33** exhibits the highest stress level. The associated results are presented in Figures 4.30, 4.31 and 4.32. The comparison of *MNS* and *Assignment* results still shows better performance of MNS, but MNS's advantage is diminishing. On average, travel time is better by 17%, delay by 7% and overtime by 4%. In three scenarios (x = 1,2 and 4), the Assignment results in delay and overtime reach the same or a better level than MNS.

In comparison with the *benchmark* results, *MNS* generates absolutely competitive results (cp. Figure 4.33). The average delay is 2.4% better (!) than the results published by Gendreau et al. (2006). Average travel time and overtime are only 6% and 1% worse, respectively. For instances x = 1,2,4, and 5, MNS achieves the same or even better results in delay and overtime.

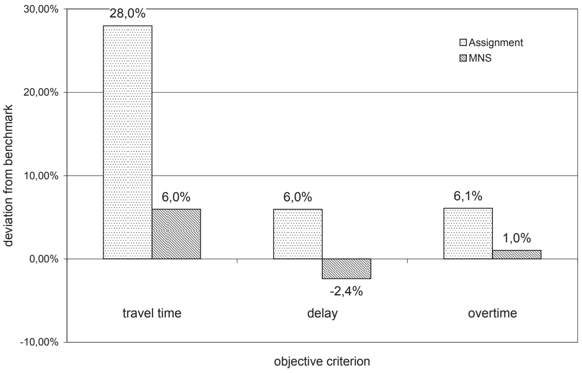


Figure 4.33: Relative comparison of results for test scenarios *req_rapide_x_240_33*

In summary, for the comparison of MNS and the Assignment based procedure, a clearly better behavior of MNS can be observed. For all 15 instances, MNS produces much better results in travel time and for 12 instances, there are also better results in delay and overtime. The best results of the Assignment based procedure are achieved for scenario 3, for which the parameterization was performed. The gap of the results between MNS and Assignment increases all the more as the test data differ from the parameterized instance. Consequently, the solution quality produced by Assignment drops from scenario 3, over scenario 2, to scenario 1. MNS is much more capable of persisting on a data set with differences to the parameterized one.

Beyond that, MNS is able to attain results on a par with the benchmark results. For the parametrized scenario 3, competitive results or even better results are generated. To put these results in perspective, it should be mentioned that Gendreau et al. (2006) applied much more computation power (parallel computing network with 16 processors) to achieve the benchmark results.

4.6 Selection of a Procedure for the Real-Life Application

At this point, the results of both data sets have been presented. Now, the question of what procedure to adapt to the real-life scenario in Chapter 5 arises. In the following, the pros and cons of both procedures are discussed, and one is finally selected.

Comparing the solution quality, the conclusions for the two test data sets do not coincide exactly. For the self-generated data sets, better **performance (in the objective function criteria travel time, delay and overtime)** of the Assignment based procedure is observed, while for the benchmark data from the literature, the contrary is true (better performance of MNS). How can that be explained?

The most plausible reason is the use of capacitated vehicles in the first and of uncapacitated vehicles in the second test data set. MNS seems to be better in the uncapacitated context and the Assignment based procedure in the capacitated case. This assumption, is supported by the findings of scenario 6 of the self-generated test data set. Here, MNS is able to achieve superior results in the case of extended vehicle capacities of 6 and 7 units, whereas better results for the low capacity instances (1 up to 4 units) are achieved by Assignment.

In terms of **robustness**, an advantage of MNS can be detected for both data sets. While the Assignment based procedure generates better results on average for the self-generated test scenarios, some data variations (e.g., reduction of available vehicles) result in an inacceptable worse solution quality of the Assignment results. In such cases, MNS shows a clearly better adaptability. The same behavior can be observed for the data set from the literature. The more different the scenario is from the scenario used for parameterization, the more the solution quality decreases in comparison with MNS and the benchmark results.

A further aspect is the **parameterization** process itself. The time discrete simulation of the Assignment based procedure is shorter than the MNS simulation runs that are accomplished in real time. Nevertheless, the parameterization of the Assignment based procedure necessitates more effort than the parameterization of MNS for both test data sets. This is due to a higher number of parameters in Assignment, a strong mutual dependency of the Assignment parameters and a high sensitivity of solution quality to changes in parameter settings.

In addition, there is also the **complexity** aspect. Which procedure is better suited for illustration in practice? This is a subjective assessment. In this work, lower complexity is associated with MNS. This is attributed to its very intuitive planning steps and the relatively insensitive parameters (compared to Assignment).

The discussed pros and cons are summarized in Table 4.13: Plus and minus signs indicate better and worse assessment in the associated category, respectively.

Except for the category *performance on self-generated data set (capacitated),* MNS achieves all pluses. This result makes the decision of which procedure to adapt to the real-life sce-

	MNS	Assignment
self-generated test data set (capacitated)		
performance (objective function criteria):	−	+
robustness:	+	−
test data set from literature (uncapacitated)		
performance (objective function criteria):	+	−
robustness:	+	−
parameterization effort	+	−
complexity of procedure	+	−

Table 4.13: MNS or Assignment? - pros and cons

nario rather difficult. In particular, because the real-life scenario is a strictly capacitated problem (SLPDPTW).

Thus, the application of the Assignment based procedure, which performed very well for such capacitated scenarios, seems to promise "better results". On the other hand, there may be some problems with robustness, especially in situations with changing time window characteristics, in situations of vehicle scarcity, or with general inhomogeneity in the real-life data.

MNS, in contrast, is expected to produce only "good" results, inferior to the Assignment ones. However, MNS promises a higher level of robustness. The risk of a complete failure, due to unknown variations of the data set, is reduced. The aspects of an easier parameterization and of a better illustration to people in practice should also not be dismissed.

Finally, after a detailed weighing up of the arguments, the safe way is preferred, selecting MNS for adaptation to the real-life scenario.