The Influence of Accurate Travel Times on a Home Health Care Scheduling Problem*

Matthias Prandtstetter **Andrea Rendl**
Jakob Puchinger
AIT Austrian Institute of Technology
Mobility Department, Dynamic Transportation Systems
Giefinggasse 2, 1210 Vienna, Austria
Email: {matthias.prandtstetter|andrea.rendl|jakob.puchinger}@ait.ac.at

1 Introduction

The home health care (HHC) problem deals with finding an optimal assignment of nurses to nursing services (at patients' homes) such that the overall working (and travel) time is minimized while customer and nurse satisfactions are maximized. We consider a real-world HHC problem setup from a Viennese health care company where we have the following, partly soft side constraints:

- nurses are expected at the patients' homes within certain **time windows**,
- for each job, we have a **preferred start time**,
- each job requires a minimum **qualification** that the nurse must hold (e.g. a nurse for cleaning may not perform a medical service),
- the schedule should meet **preferences** of patients, nurses and employer,
- the nurses' work time must follow **legal regulations** and **their contract** (concerning e.g. resting periods, working hours per week),
- each nurse uses her/his **preferred mode of transport** (either motorized private transport or public transport)

Hence, the HHC problem combines two hard-to-solve combinatorial problems—the vehicle routing problem with time windows and the nurse rostering problem—which indicates that the HHC problem is member of the class of NP-complete problems.

Within this work, we study the influence of appropriate travel times on solving the real-world HHC problem as well as the quality of the obtained roster and tour plans. Therefore, we estimate driving times for motorized private transport via a large set of historical data while public transport travel times are provided by a local public transport data company.

*This work is supported by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) within the strategic programme I2VSplus under grant 826153 as well as our project partners Verkehrsverbund Ost-Region GmbH (ITS Vienna Region), Sozial Global AG and ilogs mobile software GmbH by providing appropriate (anonymized) data.

2 A Hybrid Solution Approach

We employ a hybrid approach for the HHC problem, consisting of two major parts: (1) an *initialization step*, where we generate a (valid) initial schedule, and (2) an *improvement phase*, where we systematically improve the initial solution without loosing validity. This is achieved using a special evaluation function which guarantees that every invalid solution is worse than the worst valid solution.

2.1 Initialization

We use a Constraint Programming (CP) approach to generate feasible initial solutions. In CP, the problem is represented by discrete variables on which arbitrary (nonlinear) constraints are imposed. After filtering the variables' domains wrt the constraints, the variables are systematically searched upon until a solution is found. We extend the standard VRPTW-constraint-model [4] with additional constraints concerning the roster, aspects and multi-modality, and apply a static search strategy, where we first fix half of the tours, and then assign half of the tours to nurses before we fix the remaining tours and nurses.

Since the instances are particularly large (ca. 700 jobs, 500 nurses per day), we need to decompose the problem into subinstances: first, we split the instance by qualification, and iteratively solve each subinstance starting with the highest qualification. Second, we cluster particularly large subinstance by area, where we choose k closely-located jobs and nurses using a quadtree heuristic. If the k-clustered subinstance is not solved within a given time limit τ, we simplify the instance by iteratively removing jobs and (in case of further fails) by adding nurses and increasing the time-limit τ, until a solution is found.

As backup, as well as to evaluate the influence of the initial solution during the later improvement phase, we also provide a random solution which only guarantees that a) all jobs are executed and b) all pre-allocated jobs (e.g. appraisal interviews) remain assigned to the corresponding nurses.

2.2 Variable Neighborhood Search/Descent Approach

For the improvement phase we followed two design criteria: first, we searched for a solution which is flexible enough to be easily adapted to other but yet related rostering/tour planning problems (e.g. rostering for other home health care companies). Second, we needed to find an approach which is powerful enough to tackle real-world instances, i.e. instances of huge size. Therefore, we decided to implement a general *variable neighborhood search* (VNS) scheme [1] incorporating a *variable neighborhood descent* (VND) as local search procedure. One one hand, this metaheuristic turned out to be efficient for many related problems (e.g. [2]). One the other hand, it is very easy to keep the approach as general as possible since problem specific knowledge is only necessary within the objective function (which has to be adapted anyway when applying our approach to problem instances stated by other home health care companies) and in the definition of the neighborhoods of the local search procedure. The neighborhoods are based on rather general moves: first, the so-called *swap nurses* move simply swaps the tours of two nurses with each other. The second move type (*shift mission*) moves one mission from one tour to another tour. For the new tour, the best fitting slot along the tour is chosen. The third move type (*reposition mission*) tries to find a new slot for a selected mission within its current tour.

The initial neighborhood order applied in the VND is set to (1) *swap nurses*, (2) *shift mission*, and (3) *reposition mission*. During the execution of the VND, the neighborhood order is adapted using the same way as presented in [3]. For the shaking phase of VNS we apply i random shift moves in the i-th neighborhood, with $1 \leq i \leq 5$.

3 Preliminary Results and Discussion

We apply our methods on ten (anonymized) real-world one-day-instances (approx. 700 jobs and 500 nurses), as summarized in Table 1. For each instance, we list the objective values after initialization (either using the CP approach or the random solution), after applying only VND (without enclosing VNS) and after a fully executed VNS. Additionally, we present the number of VND iterations after the first full application of VND and the number of total VND iterations after VNS finally ended its search. We consider 4 travel time scenarios: motorized private transport (CAR), public transport (PUBLIC), both transport modes depending on the nurses' preferences (INTERMODAL), and a fixed travel time scenario where the travel time between all jobs is estimated to be 15 minutes (FIFTEEN)[1]. Final results were always obtained within at most two hours runtime[2].

First, we see that the application of VND and VNS/VND significantly reduces the initial objective value, where using the VNS further improves the solutions. Second, generating valid initial solutions with the CP approach has a positive effect on the (total) number of VND iterations. Third, we compare results from using fixed travel times with results using estimated, varying travel times based on realistic data and see that the solution quality is improved with estimated travel times. Furthermore—when having a closer look at the generated tours—more suitable round trips are computed. On average, values for car are better than intermodal transport values, since the objective value includes a term for travel times and the car scenario does not consider times for searching parking slots and walking to/from the destination due to missing data. Finally, we also observe an impact of fixed/varying travel times on our methods: the CP approach finds initial solutions far quicker with varying travel times, since the search procedure is driven by selecting the closest jobs (if all jobs have the same distance this cannot guide search). On the other hand, the VND/VNS approach converges faster with fixed travel times, which probably results from the smaller improvement potential compared to the varying case.

In summary, we observe a notable impact of travel times on the HHC solving process

[1] which is the current procedure at the HHC company due to missing data

[2] initial solution generation takes about 20 secs (CP), 2 min (CP, FIFTEEN) and <0.01 secs (random)

and see that for all travel time scenarios, the CP—VND/VNS approach shows to be the most effective technique and provides the best results.

Table 1: All values are means over ten runs with standard deviations given in parentheses below. Instance 03 is unsolvable due to a data error.

		INTERMODAL					CAR					PUBLIC					FIFTEEN				
		init.	VND		final		init.	VND		final		init.	VND		final		init.	VND		final	
		value	iter.	value	iter.	value	value	iter.	value	iter.	value	value	iter.	value	iter.	value	value	iter.	value	iter.	value
inst_01	CP	0.0885	2011.6	0.0306	3028.2	0.0292	0.0843	1941.0	0.0277	2737.0	0.0270	0.0894	1968.5	0.0330	2924.3	0.0320	0.0954	1141.1	0.0431	1473.8	0.0424
		—	(28.9)	(0.0007)	(459.1)	(0.0004)	—	(58.9)	(0.0006)	(370.9)	(0.0004)	—	(46.3)	(0.0005)	(411.9)	(0.0007)	—	(51.3)	(0.0005)	(186.8)	(0.0007)
	rand.	159.7521	2820.5	0.5316	4035.6	0.0300	159.9421	2659.1	0.0289	3618.6	0.0274	162.1559	2835.4	0.4337	4027.5	0.2320	161.1468	1662.2	1.4460	2174.9	1.1443
		—	(85.1)	(0.5271)	(679.3)	(0.0012)	—	(96.6)	(0.0006)	(360.2)	(0.0007)	—	(75.4)	(0.6989)	(539.1)	(0.4220)	—	(41.7)	(0.5163)	(258.4)	(0.5676)
inst_02	CP	0.0893	1987.4	0.0310	2720.6	0.0302	0.0841	1838.4	0.0284	2569.0	0.0276	0.0901	1894.0	0.0332	2622.4	0.0323	0.0956	1106.8	0.0432	1509.4	0.0424
		—	(53.9)	(0.0006)	(280.7)	(0.0004)	—	(72.9)	(0.0005)	(325.6)	(0.0006)	—	(57.1)	(0.0004)	(292.2)	(0.0006)	—	(28.7)	(0.0010)	(142.0)	(0.0010)
	rand.	149.9526	2724.5	0.5318	3606.8	0.3306	152.1429	2591.4	0.1286	3289.9	0.0278	151.4570	2727.3	0.3334	3556.6	0.2324	151.1473	1608.0	0.5442	1980.6	0.5433
		—	(76.9)	(0.5274)	(317.9)	(0.4831)	—	(43.8)	(0.3165)	(380.6)	(0.0003)	—	(79.4)	(0.4832)	(421.7)	(0.4220)	—	(42.6)	(0.5278)	(189.5)	(0.5276)
inst_03	CP	1.0860	1909.2	1.0293	2776.6	1.0281	1.0812	1832.2	1.0264	2609.0	1.0255	1.0873	1892.0	1.0310	2551.1	1.0303	1.0946	1131.7	1.0403	1454.3	1.0395
		—	(46.4)	(0.0005)	(368.6)	(0.0005)	—	(77.7)	(0.0008)	(432.2)	(0.0006)	—	(34.3)	(0.0007)	(416.4)	(0.0006)	—	(37.0)	(0.0006)	(234.6)	(0.0008)
	rand.	144.4498	2661.0	1.0292	3710.3	1.0280	147.8399	2528.0	1.0266	3201.3	1.0259	146.9539	2708.4	1.0310	3426.8	1.0301	148.1446	1567.8	1.0416	2066.3	1.0402
		—	(95.3)	(0.0008)	(525.2)	(0.0008)	—	(72.7)	(0.0006)	(197.1)	(0.0004)	—	(65.4)	(0.0002)	(394.8)	(0.0006)	—	(46.7)	(0.0006)	(238.7)	(0.0007)
inst_04	CP	0.0896	1949.8	0.0310	2877.1	0.0299	0.0842	1784.2	0.0290	2570.2	0.0281	0.0907	1863.6	0.0335	2811.4	0.0325	0.0960	1132.4	0.0428	1668.6	0.0416
		—	(36.2)	(0.0005)	(422.1)	(0.0006)	—	(63.5)	(0.0005)	(374.3)	(0.0006)	—	(53.1)	(0.0005)	(550.3)	(0.0004)	—	(29.1)	(0.0007)	(191.4)	(0.0007)
	rand.	156.5531	2598.5	0.0325	3361.6	0.0312	156.0438	2543.5	0.0299	3458.7	0.0285	157.1577	2666.0	0.0343	3629.7	0.0329	155.1489	1583.6	0.0449	2033.4	0.0437
		—	(73.9)	(0.0011)	(413.1)	(0.0012)	—	(68.2)	(0.0010)	(442.8)	(0.0008)	—	(77.2)	(0.0007)	(387.9)	(0.0010)	—	(40.2)	(0.0009)	(299.0)	(0.0010)
inst_05	CP	0.0885	2038.5	0.0314	2889.6	0.0303	0.0847	1932.6	0.0288	2562.8	0.0280	0.0895	1962.6	0.0336	2713.3	0.0325	0.0960	1129.7	0.0447	1627.2	0.0435
		—	(39.3)	(0.0005)	(373.0)	(0.0008)	—	(65.6)	(0.0006)	(373.2)	(0.0007)	—	(39.3)	(0.0005)	(261.9)	(0.0006)	—	(52.3)	(0.0009)	(375.3)	(0.0012)
	rand.	164.8494	2925.7	0.0314	3841.7	0.0303	167.8405	2708.7	0.0291	3697.1	0.0281	166.5543	2877.4	0.0332	3908.3	0.0320	168.1462	1722.7	0.0455	2143.5	0.0446
		—	(68.4)	(0.0006)	(300.4)	(0.0006)	—	(67.4)	(0.0006)	(412.0)	(0.0006)	—	(95.5)	(0.0005)	(563.9)	(0.0007)	—	(39.5)	(0.0009)	(233.4)	(0.0010)
inst_06	CP	0.0867	1965.4	0.0301	2865.3	0.0290	0.0825	1851.4	0.0278	2755.0	0.0266	0.0876	1922.3	0.0323	2675.2	0.0314	0.0959	1141.3	0.0422	1477.0	0.0413
		—	(62.9)	(0.0007)	(371.3)	(0.0009)	—	(83.6)	(0.0009)	(365.8)	(0.0008)	—	(53.7)	(0.0006)	(290.6)	(0.0007)	—	(42.3)	(0.0007)	(177.7)	(0.0010)
	rand.	147.3521	2740.1	0.0308	3942.7	0.0295	145.6432	2600.0	0.0282	3411.4	0.0272	143.7569	2730.3	0.0325	3346.3	0.0319	145.1481	1619.2	0.0433	2086.0	0.0420
		—	(32.3)	(0.0007)	(621.6)	(0.0010)	—	(67.1)	(0.0007)	(409.1)	(0.0005)	—	(33.5)	(0.0003)	(319.9)	(0.0005)	—	(42.6)	(0.0010)	(217.3)	(0.0011)
inst_07	CP	0.0904	1992.7	0.0323	2963.0	0.0310	0.0838	1832.4	0.0292	2381.1	0.0286	0.0914	1949.5	0.0344	2897.4	0.0332	0.0944	1193.2	0.0428	1658.3	0.0418
		—	(52.9)	(0.0007)	(383.6)	(0.0008)	—	(52.6)	(0.0005)	(453.3)	(0.0009)	—	(37.8)	(0.0006)	(404.2)	(0.0004)	—	(53.9)	(0.0005)	(313.8)	(0.0006)
	rand.	149.2530	2778.5	0.0322	3504.3	0.0315	149.4421	2592.6	0.0298	3663.5	0.0287	148.4568	2709.3	0.0345	3892.6	0.0332	148.1466	1661.7	0.0448	1971.1	0.0442
		—	(75.3)	(0.0007)	(492.9)	(0.0007)	—	(41.4)	(0.0005)	(596.5)	(0.0009)	—	(86.8)	(0.0009)	(377.0)	(0.0009)	—	(51.4)	(0.0014)	(198.5)	(0.0015)
inst_08	CP	—	1971.1	0.0291	2644.5	0.0277	0.0827	1841.3	0.0265	2563.1	0.0254	0.0882	1922.4	0.0312	2695.9	0.0304	0.0946	1096.1	0.0416	1480.9	0.0406
		—	(49.9)	(0.0005)	(1030.1)	(0.0011)	—	(53.9)	(0.0007)	(537.9)	(0.0009)	—	(49.5)	(0.0006)	(387.2)	(0.0005)	—	(42.7)	(0.0010)	(171.7)	(0.0009)
	rand.	155.9448	2763.2	0.4295	3646.0	0.4284	157.2355	2585.8	0.6274	3296.1	0.6263	157.7491	2710.5	0.0320	3605.2	0.0308	157.1407	1572.2	0.2438	1950.3	0.0426
		—	(77.5)	(0.8433)	(482.2)	(0.8437)	—	(79.6)	(0.9665)	(282.4)	(0.9662)	—	(80.1)	(0.0008)	(448.6)	(0.0007)	—	(35.5)	(0.6327)	(223.7)	(0.0013)
inst_09	CP	0.0886	1865.4	0.0304	2732.8	0.0294	0.0816	1764.2	0.0275	2719.6	0.0263	0.0869	1834.9	0.0326	2778.1	0.0314	0.0958	1135.1	0.0412	1442.3	0.0405
		—	(57.9)	(0.0006)	(413.8)	(0.0008)	—	(46.9)	(0.0007)	(429.3)	(0.0007)	—	(69.5)	(0.0004)	(413.1)	(0.0007)	—	(41.6)	(0.0009)	(177.0)	(0.0008)
	rand.	165.3541	2655.0	0.0309	3482.7	0.0298	164.8446	2548.8	0.0278	3221.0	0.0271	165.8581	2678.1	0.0327	3690.6	0.0315	165.1496	1645.7	0.0435	2126.8	0.0422
		—	(70.9)	(0.0005)	(422.7)	(0.0010)	—	(84.1)	(0.0003)	(350.3)	(0.0008)	—	(71.7)	(0.0010)	(339.6)	(0.0005)	—	(55.0)	(0.0014)	(286.5)	(0.0009)
inst_10	CP	0.0906	2106.5	0.0300	2936.1	0.0288	0.0851	2017.0	0.0270	2694.0	0.0262	0.0914	2088.9	0.0313	2807.5	0.0305	0.0971	1201.4	0.0425	1604.1	0.0418
		—	(61.9)	(0.0006)	(310.2)	(0.0007)	—	(77.4)	(0.0008)	(365.5)	(0.0007)	—	(65.9)	(0.0006)	(373.6)	(0.0008)	—	(45.8)	(0.0008)	(223.2)	(0.0006)
	rand.	166.5479	2942.9	0.0299	3912.7	0.0288	167.1392	2751.1	0.0272	3687.1	0.0262	166.0529	2945.1	0.0321	3967.4	0.0307	167.1440	1712.1	0.0449	2269.1	0.0434
		—	(105.5)	(0.0006)	(393.7)	(0.0006)	—	(36.1)	(0.0004)	(407.2)	(0.0006)	—	(62.4)	(0.0004)	(433.7)	(0.0008)	—	(63.7)	(0.0016)	(296.6)	(0.0012)

References

[1] P. Hansen and N. Mladenović. Variable neighborhood search. In F. W. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publisher, New York, 2003.

[2] S. Pirkwieser and G. R. Raidl. A variable neighborhood search for the periodic vehicle routing problem with time windows. In C. Prodhon et al., editors, *Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing*, Troyes, France, 23–24 Oct. 2008.

[3] M. Prandtstetter, G. R. Raidl, and T. Misar. A hybrid algorithm for computing tours in a spare parts warehouse. In C. Cotta and P. Cowling, editors, *Evolutionary Computation in Combinatorial Optimization - EvoCOP 2009*, volume 5482 of *LNCS*, pages 25–36. Springer, 2009.

[4] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.