Hybridization strategies for routing problems with synchronization constraints

Sophie N. Parragh[1,2], Karl F. Doerner[3]

[1] INESC Porto / IBM CAS Portugal
Rua Dr. Roberto Frias, 378 – 4200-465 Porto, Portugal

[2] Department of Business Administration
University of Vienna
Bruenner Str. 72 – 1210 Vienna, Austria
sophie.parragh@univie.ac.at

[3] Production and Logistics Management
Johannes Kepler University Linz
Altenberger Strasse 69 – 4040 Linz, Austria
karl.doerner@jku.at

Abstract

Route synchronization is an issue encountered in several different real life scenarios. However, the literature addressing route synchronization issues is relatively scarce. In this paper, we propose three hybridization strategies to deal with pairwise route synchronization constraints in the presence of time windows.

1 Introduction

In this paper we address the issue of pairwise route synchronization; that is, a customer or task location has to be visited by two vehicles at exactly the same time. This issue is commonly encountered in the field of service technician routing and scheduling and in the area of mobile care, where two technicians or nurses are required to complete a given task. It is also encountered, e.g., in forest harvesting operations [1]. The literature explicitly dealing with pairwise route synchronization is relatively scarce, see, e.g. [1, 5]. In this work, we consider a generic problem based on the vehicle routing problem with time windows (VRPTW), which we denote VRPTW with pairwise route synchronization (VRPTWPS), and the service technician routing and scheduling problem with pairwise route synchronization (STRSPPS). The latter is motivated by a real world problem from an Austrian infrastructure service provider. In both problem situations we minimize total routing costs. Each customer or task is associated with a time window and either with a demand (VRPTWPS) or certain skill requirements (STRSPPS). A given fleet of vehicles with given capacities (VRPTWPS) or a given group of technicians with given skills (STRSPPS) is available to serve the customers or tasks.

2 Solution framework

We propose a hybrid method in order to deal with pairwise route synchronization constraints. The metaheuristic component consists of an adaptive large neighborhood search (ALNS) algorithm which is based on the ALNS developed in [3] to solve a service technician routing and scheduling problem without synchronization requirements. In general, ALNS works as follows. Starting with a first feasible solution, in every iteration, a destroy and a repair operator are employed to the current incumbent solution, hopefully yielding a solution of improved quality. In ALNS, as first proposed by Ropke and Pisinger [4], a set of destroy and a set of repair operators are used and an adaptive scheme guides their selection. While in the original version destroy and repair operators are selected and remunerated separately for good performance, Kovacs et al. [3] use operator pairs. The employed destroy operators are referred to as random removal, worst removal, related removal, and cluster removal; the employed repair operators

are a greedy insertion, a sequential insertion [7], and several regret insertion heuristics. In addition, the ALNS is embedded into a simulated annealing framework [2]. For further details we refer to [3].

In order to deal with the above described synchronization issues, we propose to integrate an exact component into the ALNS of [3]. The aim of this component is to improve the positioning of the synchronized tasks within their routes. During the execution of the ALNS, time windows of synchronized tasks are narrowed to points in time. This approach guarantees that both vehicles arrive at the same time at the synchronized task. In order to improve this point in time, we propose to integrate an exact component. This exact component is used every i_{freq} number of iterations and every time a new best solution is found. The general scheme is the following. For each synchronized task or customer, we first determine the two routes serving it. Then, we reset the time window of the task to its original value and we simultaneously optimize its positioning on both routes; that is, the total routing costs of the two routes are minimized respecting the different time window constraints. Since the customer or task to route assignment decision is not touched, it is not necessary to consider capacity or skill requirements. If a better positioning can be found, the time window is narrowed again in accordance with the new positioning, and the ALNS resumes. The underlying optimization problem can be modeled and solved in several ways. We identified three such approaches and compare their performance on a set of benchmark instances.

MIPit approach In the first approach, we formulate the underlying optimization problem in terms of a mixed integer program (MIP). Let r_1 and r_2 denote the routes of synchronized task s and let s_1 and s_2 denote the synchronized task inserted into r_1 and r_2, respectively. Then, we generate an arc set A that consists of all arcs $\in r_1$ and $\in r_2$ plus all arcs connecting any of the nodes in r_1 with s_1, and all those in r_2 with s_2. On this basis, the problem can be formulated in terms of an arc flow formulation and solved by a mixed integer problem solver.

DPit approach The second approach relies on dynamic programming (DP). Here, the fundamental concept is the way the graph for finding the shortest path, while taking care of synchronization issues, is defined. We propose to construct a graph of $|r_1| \times |r_2| + 1$ nodes. Each of the nodes in the graph corresponds to one node of r_1 and one node of r_2. If the nodes were given in a grid like layout, horizontal arcs would correspond to moving to the successor node in r_2 while staying at the same node in r_1; vertical arcs would correspond to moving to the successor node in r_1 while staying at the same node in r_2. Diagonal arcs would finally correspond to moving to the successor node in both routes. Diagonal nodes may also be connected via a detour through the synchronized node. Thus, each node is connected to at most 4 nodes: the next horizontal, the next diagonal, the next vertical and the synchronized node. At all regular nodes cost and time consumptions are updated for each route individually; at the synchronized node, the beginning of service times are synchronized. We solve the shortest path problem on this graph by means of a label setting algorithm.

CGit approach The third approach is inspired by column generation (CG). Instead of an arc flow formulation, we use a set-partitioning type one. In a first step, for each of the two routes considered, all feasible insertion positions of s are determined and the according routes are put into two route pools Ω_{r_1} and Ω_{r_2}, referring to the route pools of route r_1 and route r_2, respectively. For each insertion position, we also calculate the beginning of service time of the synchronized task and its forward time slack F [6]. We then solve a set partitioning problem where exactly one route has to be chosen from each of the two sets, while the beginning of service of s_1 and s_2 have to be synchronized; that is, their current beginning of service times can at most be increased by F.

3 Results

In order to test the different hybridization strategies on the VRPTWPS, we use Solomon's benchmark data set [7]. Following [5], we assume that every 10th customer demands a synchronized visit. This also

	MIPit			DPit			CGit		
i_{freq}	10	25	100	10	25	100	10	25	100
C1	3602.11	2668.35	1253.17	24.67	25.82	22.07	534.35	256.46	168.13
C2	3603.96	3570.04	608.35	138.26	101.12	66.94	1484.18	626.64	368.35
R1	3602.36	3084.40	2169.05	24.77	23.16	20.49	647.20	252.06	163.19
R2	3603.67	3603.27	3581.53	238.70	150.68	104.71	2311.25	943.00	629.62
RC1	3534.19	3452.42	2409.29	24.05	23.37	21.38	537.56	247.53	160.99
RC2	3603.21	3604.25	3603.49	100.63	77.11	63.28	541.82	1934.38	854.56
Avg	3591.58	3330.45	2270.81	91.85	66.87	49.81	1009.9	710.01	390.81

Table 1: Run times in seconds for different i_{freq} settings using MIPit, DPit and CGit on data sets based on [7]

	DPit				
	No sync	Fixed	10	25	100
C1	828.31	1251.91	**1194.54**	1200.01	1207.57
C2	589.86	940.05	**892.07**	894.10	896.34
R1	1185.89	1389.80	**1361.62**	1362.01	1366.61
R2	884.11	1080.33	1029.98	**1029.77**	1031.77
RC1	1355.00	1625.26	**1556.26**	1563.86	1565.12
RC2	1013.55	1282.92	**1218.31**	1222.79	1223.71
Avg.	976.12	1261.71	**1208.80**	1212.09	1215.19

Table 2: Results for DPit with different i_{freq} settings on data sets based on [7]

corresponds to the assumptions made in [1] where 10% of the customers demand synchronization.

The proposed algorithm is implemented in C++ and CPLEX 12.1 together with Concert Technology 2.9 is used to solve the MIPs. We allow a maximum of 25×10^3 iterations or one hour of total run time. Table 1 provides the run times for the different exact components with different i_{freq} settings. DPit is definitely the most competitive method in terms of run time. Table 2 provides average solution values for DPit, compared to the average results obtained with the ALNS on the original data set without synchronization requirements ('No sync') and a setting without any intermediate improvements and synchronized tasks fixed to points in time ('Fixed'). All values are average values across all instances of the data set and 5 random runs per instance.

Improvements of about 4% on average can be obtained with DPit ($i_{freq} = 10$) compared to the 'Fixed' setting. The synchronization requirement leads on average to about 24% increased routing costs (comparing 'No sync' and DPit with $i_{freq} = 10$). Similar results are obtained for the STRSPPS.

Acknowledgments This research is supported by the Austrian Science Fund (FWF) under grant # T514-N13. This support is gratefully acknowledged.

References

[1] D. Bredström and M. Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *Eur J Oper Res*, 191:19–31, 2008.

[2] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[3] A. A. Kovacs, S. N. Parragh, K. F. Doerner, and R.F. Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. Technical report, University of Vienna, Faculty of Business, Economics and Statistics, 2010.

[4] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transport Sci*, 40:455–472, 2006.

[5] L.-M. Rousseau, M. Gendreau, and G. Pesant. The synchronized vehicle dispatching problem. Technical report, CIRRELT, 2010.

[6] M. W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA J Comput*, 4:146–154, 1992.

[7] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper Res*, 35:254–265, 1987.