A Fast Heuristic Algorithm for the Train Unit Assignment Problem

Valentina Cacchiani, Alberto Caprara*, and Paolo Toth

DEIS, University of Bologna,
Viale Risorgimento 2, I-40136 Bologna, Italy
valentina.cacchiani@unibo.it,alberto.caprara@unibo.it,paolo.toth@unibo.it

Abstract

In this paper we study a railway optimization problem known as the Train Unit Assignment Problem. A train unit consists of a self-contained train with an engine and a set of wagons with passenger seats. Given a set of timetabled train trips, each with a required number of passenger seats, and a set of train units, each with a given number of available seats, the problem calls for the *best* assignment of the train units to the trips, possibly combining more than one train unit for a given trip, that fulfills the seat requests. We propose a heuristic algorithm based on the computation of a lower bound obtained by solving an Integer Linear Programming model that gives the optimal solution in a "peak period" of the day. The performance of the heuristic algorithm is computationally evaluated on real-world instances provided by a regional Italian Train Operator. The results are compared with those of existing methods from the literature, showing that the new method is able to obtain solutions of good quality in much shorter computing times.

1998 ACM Subject Classification G.1.6 Integer Programming, G.2.3 Applications

Keywords and phrases Train Unit Assignment, Heuristic Algorithm, ILP model, Real-world instances

Digital Object Identifier 10.4230/OASIcs.ATMOS.2012.1

1 Introduction

It is well-known that the optimization of a railway system is very hard and is generally performed in separate phases, each one corresponding to a very difficult problem itself (see e.g. [11], [12]). In this paper, we focus on the so-called *Train Unit Assignment Problem* (TUAP), which is also known as *Rolling Stock Planning Problem*. A train unit consists of a self-contained train with an engine and a set of wagons with passenger seats. Given a set of timetabled train trips, each with a required number of passenger seats, and a set of train units, each with a given number of available seats, the problem calls for the *best* assignment of the train units to the trips, possibly combining more than one train unit for a given trip, that minimizes the number of used train units and satisfies a set of real-world constraints. The constraints are the following: covering constraints (i.e. the request of passenger seats must be satisfied for each trip), maximum combination constraints (i.e., for each trip a maximum number of train units can be combined in order to cover the trip), sequencing constraints (i.e. two trips can be performed in sequence by a train unit if and only if there is enough time for the train unit for traveling from the arrival station of the first trip to the

* Alberto Caprara passed away unexpectedly on April 21, 2012. At that time this work was almost completed. The other two authors wrote the paper finding inspiration in his suggestions and ideas.

12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'12).
Editors: Daniel Delling, Leo Liberti; pp. 1–9

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

departure station of the second trip), and availability constraints (i.e. no more units than the ones available can be used). TUAP is strongly NP-hard, as it has been proven in [8].

1.1 Literature review

There is a considerable amount of literature on Rolling Stock Planning: locomotives and cars have to be assigned to trips (see e.g. [4], [14], [15], [26], [28]) instead of self-contained train-units (see e.g. [1], [2], [3], [18], [25], [27]); the order of the train-units assigned to a trip has to be considered (see e.g. [18], [21], [22], [25]); different objective functions can be considered for the problem, for example the goal can be to obtain a robust solution in order to take care of possible disruptions (see e.g. [7], [24]). For surveys on the specific problem as well as on the use of combinatorial optimization in railway planning see, e.g., [5, 9, 11, 12, 13, 16, 17, 20].

1.2 Outline of the paper

We consider the version of TUAP studied in works [6], [8] and [10]. In [8] an Integer Linear Programming (ILP) model for the problem with one variable for each possible daily schedule of each train unit is proposed. A diving heuristic based on the Linear Programming (LP) relaxation of this model is developed, combined with a refinement procedure. In the recent work [10], a heuristic based on the Lagrangian relaxation of an alternative ILP model is developed. The relaxed solution is computed by solving a sequence of assignment problems. In this work we propose a new heuristic algorithm, which is based on the computation of a lower bound obtained by solving an ILP model that gives the optimal solution in a "peak period" of the day.

The paper is organized as follows: in Section 2 we formally describe the problem and in Section 3 we describe the heuristic algorithm we propose. In Section 4 we present computational experiments on real-world instances and compare the results with those of existing methods from the literature. Finally, in Section 5 we draw some conclusions and guidelines for future research.

2 Problem Description

The problem input specifies a set of n timetabled *train trips* to be executed every day of a given time period, and a set of p *train unit types*. Each trip $j \in \{1, \dots, n\}$ is defined by a required number r_j of passenger seats, and a maximum number u_j of train units that can be assigned to the trip. In the specific application we consider, we have $u_j = 2$ for $j = 1, \dots, n$, i.e., each trip can be assigned to at most two train units. Each trip is also characterized by its departure and arrival times and departure and arrival stations. Each train unit type $k \in \{1, \dots, p\}$ is defined by a number d^k of available train units, and an associated capacity s^k (number of available seats). We say that a trip j is *covered* if the overall capacity of the train units assigned to the trip is at least r_j.

We introduce a directed acyclic graph $G = (V, A)$, where each node corresponds to a trip, i.e., $V = \{1, \dots, n\}$, and arc $(i, j) \in A$ exists if and only if a train unit (of any type) can be assigned to i and then to j within the same day. In other words, arc (i, j) exists whenever both trips i and j can be assigned to a train unit, and the time between the arrival of trip i and the departure of trip j allows the train unit to travel from the arrival station of trip i to the departure station of trip j within the same day. It is to note that there is an overnight break of a few hours, i.e. no trips have to be covered during the night. In addition, it is

possible for any train unit to perform a "transfer" (i.e. to travel between any pair of stations) within the night break. Therefore, it is not necessarily the case that every used train unit performs the same set of trips every day. Indeed, after having performed a sequence of trips on one day, a train unit can perform on the following day a trip sequence assigned to another train unit of the same type (possibly performing a transfer within the night break). In other words, number the, say, q trip sequences assigned to the train units of a given type as $1, \ldots, q$ in an arbitrary way. These q trip sequences can be performed by q train units of that type, all performing a different sequence on each day, and each one performing the q sequences in the cyclic order $1, \ldots, q$ over a period of q days.

3 Heuristic Algorithm

In this section, we describe the proposed heuristic algorithm. It is composed of a constructive phase and a local search phase, that are described in Sections 3.2 and 3.3, respectively. The constructive phase is based on the computation of a lower bound: it is obtained by solving at optimality the problem restricted to a "peak period". The description of the lower bound computation is given in Section 3.1. Before starting the constructive phase, we apply a straightforward preprocessing on the input instance: the seat request for each trip j is redefined so that it matches the minimum sum of train unit capacities that can cover request r_j. The associated optimization problem, which is a cardinality-constrained bounded subset sum problem (see, e.g., Martello and Toth [23]), can easily be solved by enumeration given the small values of p and d^k in practical cases.

3.1 Lower Bound

The lower bounding procedure we use is described in [8], but, for sake of clarity, we briefly recall it. The main idea is to find a set of trips that are pairwise "incompatible", i.e., that cannot be performed by the same train unit (of whatever type) in the same day, and then optimally solve the subinstance restricted to these trips: the obtained optimal value is a lower bound on the optimal TUAP value. These incompatible trips correspond to a "peak period" of the day, when many simultaneous trips need to be covered with a large seat request. In order to determine a set of incompatible trips, we compute a *maximum-weight stable set* in the auxiliary undirected graph (V, E) defined by neglecting the arc orientation in graph G and adding to each node j (corresponding to trip j) a weight equal to r_j. The auxiliary graph is a *comparability graph* for which a maximum-weight stable set can be computed efficiently by using flow techniques (see, e.g., [19]). Once the set of incompatible trips has been determined, we find the optimal solution of the TUAP instance restricted to these trips, by solving the following ILP model, where S represents the set of trips in the maximum-weight stable set and w_j^k is an integer variable representing the number of times that trip j is assigned to a train unit of type k:

$$\min \sum_{k=1}^{p} \sum_{j \in S} w_j^k, \tag{1}$$

$$\sum_{j \in S} w_j^k \leq d^k, \qquad k = 1, \ldots, p, \tag{2}$$

$$\sum_{k=1}^{p} s^k w_j^k \geq r_j, \qquad j \in S, \tag{3}$$

$$\sum_{k=1}^{p} w_j^k \leq u_j, \qquad j \in S, \tag{4}$$

$$w_j^k \geq 0, \text{ integer }, \qquad k = 1, \ldots, p, \ j \in S. \tag{5}$$

The objective function calls for the minimization of the number of train units needed to perform the set S of incompatible trips. Constraints (2) impose not to use more than d^k train units of type k. Constraints (3) ensure the covering of the trips in S. Finally, constraints (4) guarantee to combine at most u_j train units for covering trip j.

Of course, the value of the optimal solution of the ILP model (1)-(5) represents a valid lower bound for the original TUAP instance given in input.

3.2 Constructive Phase

For each train unit type k we impose an upper limit $\bar{d}^k$ on the number of available units equal to the corresponding number of units used in the optimal solution of the ILP model (1)-(5). I.e. our aim is to construct a feasible solution for the original TUAP instance with the same value of the lower bound: obviously, if we manage to find it, this is an optimal solution. Otherwise, we apply the local search phase after the constructive phase.

As observed in [8], TUAP can be solved efficiently if there is a unique train unit type. This means that, if we have a subset of m trips assigned to a train unit type, the computation of the best possible sequencing of these trips can be done in polynomial time ($O(m^3)$). In particular, the problem corresponds to an Assignment Problem that can be solved, for a train unit of type k, by using a directed graph $\bar{G}^k = (\bar{V}^k, \bar{A}^k)$. It has a subset $\bar{V}^k$ of nodes in V corresponding to the trips assigned to the current train unit type k, and the set of arcs $\bar{A}^k$ defined as follows: all the arcs in A between nodes in $\bar{V}^k$ belong to $\bar{A}^k$ and have cost equal to 0; in addition, we define an arc between any two nodes in $\bar{V}^k$ such that they are not connected in A and we set its cost to 1. In this way, each arc $(i, j) \in \bar{A}^k$ with cost equal to 0 (respectively, equal to 1) corresponds to a pair of trips i and j that can be covered in sequence by a train unit of type k in the same day (respectively, in two consecutive days, with a night within). As a consequence, the cost of the solution of the Assignment Problem corresponds to the number of consecutive days (i.e. to the number of different units) required by the train units of type k to cover all the trips currently assigned to it.

Since the sequencing of the trips for each train unit type can be solved efficiently, the constructive phase decides how to assign each trip to one or two train unit types and then solves, for each train unit type, an Assignment Problem to determine the optimal sequence of the trips. The assignment is accepted only if the corresponding number of units of type k does not exceed $\bar{d}^k$.

What remains to be determined is a good policy for assigning a trip to the *best* train unit type. We consider the trips in the maximum-weight stable set as the first ones to be

assigned and consider the remaining ones in chronological order. For each trip i, we perform the following steps:

1. If $r_i > s^{k_{max}}$, where k_{max} corresponds to the train unit type with the largest capacity, then assign i to two train unit types k_1 and k_2 ($k_1, k_2 = 1, \ldots, p$), possibly with $k_1 = k_2$, such that $s^{k_1} + s^{k_2} - r_i$ is minimum and non negative. Otherwise, assign i to a train unit type k ($k = 1, \ldots, p$) such that $s^k - r_i$ is minimum and non negative.
2. Compute the optimal solution of the Assignment Problem corresponding to the trips assigned up to now to the considered train unit type k (or train unit types k_1 and k_2). Check if the value of the optimal solution of the Assignment Problem corresponding to each considered train unit type k does not exceed the number of available train units $\bar{d}^k$.
3. If this is the case, accept the assignment; otherwise try to assign trip i to a train unit type k (or even to two train unit types k_1 and k_2) with larger capacity and go to 2.
4. If no train unit type leads to a feasible assignment for trip i, leave trip i uncovered.

At the end of the constructive procedure, either we have a feasible solution with the same value of the lower bound (i.e. optimal), or we have a set of uncovered trips. In the latter case, we move to the local search phase described in Section 3.3.

3.3 Local Search Phase

Note that in the proposed heuristic algorithm, the solution of the Assignment Problem already gives the best sequencing of the trips assigned to a given train unit type. Thus, it is not useful to exchange a trip from a unit to another one of the same type, but it is only necessary to consider exchanges of trips to different train unit types.

The local search phase consists of two procedures. The first one applies an exchange move between a trip that was assigned to a given train unit type (or to two train unit types) to a different train unit type (or to two different train unit types), with the aim of allowing the insertion of an uncovered trip. More specifically, for each uncovered trip i, we execute the following procedure:

1. Consider a trip j that was assigned to a train unit type k (or to two train unit types k_1 and k_2), with $s^k \geq r_i$ (or $s^{k_1} + s^{k_2} \geq r_i$) and such that its timetable overlaps the timetable of trip i.
2. Remove trip j from train unit type k (or k_1 and k_2), and proceed as in the constructive phase in order to try to assign the uncovered trip i to the train unit type k (or k_1 and k_2).
3. If it is possible to perform this assignment without exceeding the lower bound value of train unit type k (or k_1 and k_2), then proceed as in the constructive phase in order to try to assign trip j to a different train unit type (leaving all the other assignments unchanged). Otherwise, re-assign trip j to train unit type k (or k_1 and k_2). If all the assigned trips have been considered as a possible exchange move for trip i then stop (trip i remains uncovered), otherwise go to 1.

The second procedure of the local search phase tries to compute a feasible solution taking into account all the available train units and not only those corresponding to the computation of the lower bound. The trips that are still uncovered at the end of the exchange phase are assigned to one or two currently unused train units, by performing steps 1 to 4 of the constructive phase (see Section 3.2). Note that, in our case study, due to the large number of train units available (i.e. those used in the practitioners' solutions), it was always possible to find a feasible solution with the proposed heuristic algorithm.

Table 1 Comparison of the proposed heuristic algorithm with the LP-diving heuristic presented in [8] and the Lagrangian heuristic presented in [10] on real-world instances.

				New heur.			[8] heur.			[10] heur.		
Inst.	n	p	D	LB	value	time	LB	value	time	LB	value	time
1	85	1	2	2	**2**	0	2	**2**	0	2	**2**	0
2	120	1	4	4	**4**	0	4	**4**	0	4	**4**	0
3	221	1	18	17	**17**	3	17	**17**	288	15	**17**	4
4	127	2	27	25	**25**	1	25	**25**	17	20	**25**	4
5	283	2	22	20	**20**	4	20	**20**	1912	17	**20**	18
A	528	8	72	62	66	28	62	**63**	1932	27	70	288
B	662	10	76	53	57	31	53	**54**	2309	19	59	600
C	660	10	75	53	55	31	53	**53**	1878	16	57	572
D	196	3	19	13	14	1	13	**13**	280	9	**13**	12
E	143	4	32	26	27	1	26	**26**	18	12	**26**	7
F	366	3	26	23	24	11	23	**23**	1013	11	24	49
G	348	3	45	39	42	6	39	**39**	1590	22	45	52
H	137	3	21	20	**20**	0	20	**20**	25	13	**20**	7
Avg. %Gap					2.99			0.26			3.55	
Avg. Times						9.00			866.3			124.1

4 Computational Experiments

We present computational experiments on a set of 13 real-world instances provided by a regional Italian Train Operator, and compare the results of the proposed heuristic algorithm with those of the approaches presented in [8] and in [10]. The heuristic algorithm presented in [8] is composed of a fixing phase and a refinement phase and leads to a good improvement over the practitioners' solutions but requires large computing times. The heuristic algorithm presented in [10] is composed of a constructive phase and a local search phase similar to the one described in Section 3.3. The considered algorithms are coded in C and the tests are performed on a PC Pentium 4, 3.2 GHz, 2 GB RAM.

In Table 1, we present the comparison of the proposed heuristic algorithm (New heur.) with the existing methods. For each method we report the lower bound value (LB), the solution value (value) and the computing time expressed in seconds (time). The first four columns in the table represent, respectively, the instances name, the number of trips n, the number of train unit types p and the global number of available train units D. We report in bold the best solutions found and we underline the values that correspond to optimal solutions. The last two rows report, for each method, the average percentage gap (computed by considering for each instance the value of the solution found by the corresponding method and the best lower bound), and the average computing time on the 13 instances.

Table 1 shows that the proposed heuristic algorithm is very fast in obtaining solutions of good quality. Compared with the algorithm proposed in [8], the new heuristic has a larger percentage gap but the average computing time is much shorter. Compared with the algorithm proposed in [10], the new heuristic has a smaller percentage gap, a computing time again much shorter, and obtains 4 better solutions (on the larger instances) while the heuristic proposed in [10] obtains only 2 better solutions.

In Table 2, we present a comparison between the three methods when a time limit of 30 seconds or 1 minute is imposed. For each method we report the solution values found within

Table 2 Comparison of the proposed heuristic algorithm with the LP-diving heuristic presented in [8] and the Lagrangian heuristic presented in [10] on real-world instances with a time limit imposed.

				Time limit = 30 seconds			Time limit = 1 minute		
				New heur.	[8] heur.	[10] heur.	New heur.	[8] heur.	[10] heur.
Inst.	n	p	D	value	value	value	value	value	value
1	85	1	2	**2**	**2**	**2**	**2**	**2**	**2**
2	120	1	4	**4**	**4**	**4**	**4**	**4**	**4**
3	221	1	18	**17**	**17**	**17**	**17**	**17**	**17**
4	127	2	27	**25**	**25**	**25**	**25**	**25**	**25**
5	283	2	22	**20**	22	**20**	**20**	22	**20**
A	528	8	72	**66**	-	-	**66**	-	-
B	662	10	76	**57**	-	-	**57**	-	61
C	660	10	75	**56**	-	-	**55**	-	64
D	196	3	19	14	-	**13**	14	16	**13**
E	143	4	32	27	**26**	**26**	27	**26**	**26**
F	366	3	26	**24**	25	**24**	**24**	25	**24**
G	348	3	45	**42**	-	-	**42**	-	45
H	137	3	21	**20**	**20**	**20**	**20**	**20**	**20**
Avg. %Gap w.r.t [8]				0.98	2.14		1.67	3.98	
Avg. %Gap w.r.t [10]				1.67		0.46	2.73		3.98
# found				13	8	9	13	9	12

the imposed time limit, showing in bold the best solutions found and underlining the optimal solutions. The last three rows report, for each method, the number of feasible solutions found, and the average percentage gap, computed by considering only the instances for which the new heuristic and the method presented in [8] ([10], respectively) were able to find a feasible solution. The results show that, by imposing a time limit of 30 seconds, the proposed heuristic is able to find a feasible solution for all the instances, while the approaches proposed in [8] and in [10] obtain a feasible solution for 8 and 9 instances, respectively. When a time limit of 1 minute is imposed, the methods proposed in [8] and [10] still fail in finding a feasible solution for 4 and 1 instances, respectively. We can see that the average percentage gap of the proposed heuristic is always better than the one obtained by the heuristic presented in [8]. With respect to the heuristic presented in [10], the new heuristic finds a larger average percentage gap when a time limit of 30 seconds is imposed (but, with this time limit, the method presented in [10] is not able to find a feasible solution for 4 instances), while a smaller average percentage gap is obtained by the new heuristic when a time limit of 1 minute is imposed.

5 Conclusions and Future Research

We have proposed a heuristic algorithm for the Train Unit Assignment Problem, based on the computation of a lower bound obtained by solving an Integer Linear Programming model that gives the optimal solution in a “peak period" of the day. Compared with the results obtained by existing methods, the new heuristic algorithm is able to obtain solutions of good quality in much shorter computing times. Therefore, the new heuristic algorithm can also be used in a real-time setting. Future research will be devoted to improve the performance

of the proposed algorithm, by executing it iteratively, considering, at each iteration, the unassigned trips (of the previous iteration) as the first to be assigned together with the trips in the maximum-weight stable set. In addition, larger realistic instances, presented in [10], will be tested. Finally, variants of the problem, related to Fixed Job Scheduling Problems, will be considered.

References

1 E.W.J. Abbink, B.W.V. van den Berg, L.G. Kroon, and M. Salomon: "Allocation of Railway Rolling Stock for Passenger Trains". Transportation Science **38** (2004) 33–41

2 A. Alfieri, R. Groot, L.G. Kroon, and A. Schrijver: "Efficient Circulation of Railway Rolling Stock". ERIM Research Report, ERS-2002-110-LIS, Erasmus Universiteit Rotterdam, The Netherlands, (2002)

3 N. Ben-Khedher, J. Kintanar, C. Queille, and W. Stripling: "Schedule Optimization at SNCF: From Conception to Day of Departure". Interfaces **28** (1998) 6–23

4 J. Brucker, J.L. Hurink, and T. Rolfes: "Routing of Railway Carriages: A Case Study". Osnabrücker Schriften zur Mathematik, Reihe P, Heft **205** (1998)

5 M.R. Bussieck, T. Winter, and U.T. Zimmermann: "Discrete Optimization in Public Rail Transport". Mathematical Programming **79** (1997) 415–444

6 V. Cacchiani: "Models and Algorithms for Combinatorial Optimization Problems arising in Railway Applications". 4OR A Quarterly Journal of Operations Research **7(1)** (2009) 109–112

7 V. Cacchiani, A. Caprara, L. Galli, L. Kroon, G. Maroti, and P. Toth: "Railway Rolling Stock Planning: Robustness Against Large Disruptions". Transportation Science **46(2)** (2012) 217–232

8 V. Cacchiani, A. Caprara, and P. Toth: "Solving a Real-World Train Unit Assignment Problem". Mathematical Programming Series B **124** (2010) 207–231

9 V. Cacchiani, A. Caprara, and P. Toth: "Models and Algorithms for the Train Unit Assignment Problem". In A.R. Mahjoub et al. (Eds.): *ISCO 2012„ LNCS 7422*, Springer-Verlag Berlin Heidelberg, (2012) 24–35

10 V. Cacchiani, A. Caprara, and P. Toth: "A Lagrangian Heuristic for a Train-Unit Assignment Problem". Discrete Applied Mathematics doi: 10.1016/j.dam.2011.10.035

11 A. Caprara: "Almost 20 Years of Combinatorial Optimization for Railway Planning: from Lagrangian Relaxation to Column Generation". In Thomas Erlebach and Marco Lübbecke (eds.), *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*, Schloss Dagstuhl, Germany, (2010) 1–12

12 A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth: "Passenger Railway Optimization". In C. Barnhart and G. Laporte (eds.). Handbooks in OR & MS **12**, Elsevier Science, (2006)

13 A. Caprara, L. Kroon, and P. Toth: "Optimization Problems in Passenger Railway Systems". Wiley Encyclopedia of Operations Research and Management Science **6** (2011) 3896–3905

14 J.-F. Cordeau, G. Desaulniers, N. Lingaya, F. Soumis, and J. Desrosiers: "Simultaneous Locomotive and Car Assignment at VIA Rail Canada". Transportation Research **35** (2002) 767–787

15 J.-F. Cordeau, F. Soumis, and J. Desrosiers: "Simultaneous Assignment of Locomotives and Cars to Passenger Trains". Operations Research **49** (2001) 531–548

16 J.-F. Cordeau, P. Toth, and D. Vigo: "A Survey of Optimization Models for Train Routing and Scheduling". Transportation Science **32** (1998) 380–404

17 J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis: "Time Constrained Routing and Scheduling", in M.O. Ball et al. (eds.), Handbooks in OR & MS **8**, Elsevier Science, (1995) 35–139

18 P.-J. Fioole, L.G. Kroon, G. Maróti, and A. Schrijver: "A Rolling Stock Circulation Model for Combining and Splitting of Passenger Trains". European Journal of Operational Research **174** (2006) 1281–1297

19 M. Grötschel, L. Lovász, and A. Schrijver: "Geometric Algorithms and Combinatorial Optimization". Springer-Verlag (1988)

20 D. Huisman, L.G. Kroon, R.M. Lentink, and M.J.C.M. Vromans: "Operations Research in Passenger Railway Transportation". Statistica Neerlandica **59** (2005) 467–497

21 N. Lingaya, J.-F. Cordeau, G. Desaulniers, J. Desrosiers, and F. Soumis: "Operational Car Assignment at VIA Rail Canada". Transportation Research **36** (2002) 755–778

22 G. Maróti: "Operations research models for railway rolling stock planning". PhD Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2006)

23 S. Martello and P. Toth: Knapsack Problems: Algorithms and Computer Implementations. John Wiley and Sons (1990)

24 L.K. Nielsen, L. Kroon, and G. Maróti: "A rolling horizon approach for disruption management of railway rolling stock". European Journal of Operational Research **220** (2012) 496–509

25 M. Peeters, and L.G. Kroon: "Circulation of Railway Rolling Stock: a Branch-and-Price Approach". ERIM Research Report, ERS-2003-055-LIS, Erasmus Universiteit Rotterdam, The Netherlands, (2003)

26 S. Rouillon, G. Desaulniers, and F. Soumis: "An Extended Branch-and-Bound Method for Locomotive Assignment". Transportation Research **40** (2006) 404–423

27 A. Schrijver: "Minimum Circulation of Railway Stock". CWI Quarterly **6** (1993) 205–217

28 K. Ziarati, F. Soumis, J. Desrosiers, and M.M. Solomon: "A branch-first, cut-second approach for locomotive assignment". Management Science (1999) 1156–1168