Transportation Research Part C 119 (2020) 102751

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Determination of infrastructure capacity in railway networks without the need for a fixed timetable

Lars Wittrup Jensen [a,c,*], Marie Schmidt [b], Otto Anker Nielsen [a]

[a] DTU Management, Technical University of Denmark, Bygningstorvet 116B, 2800 Lyngby, Denmark
[b] Department of Technology and Operations Management, Rotterdam School of Management, Erasmus University, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
[c] Ramboll Danmark A/S, Hannemanns Alle 53, 2300 Copenhagen S, Denmark

ARTICLE INFO

Keywords:
Railways
Absolute capacity
Capacity analysis
Tabu search
Branch-and-bound

ABSTRACT

Railway capacity is a scarce and expensive resource that has to be utilised in the best way possible. Many methods exist for medium- and short-term planning and analysis of railway capacity. Yet, there is a lack of methods for long term (strategic) planning that provides high quality estimates of capacity in complex railway networks. In this paper, we define capacity of a railway network by a maximal set of trains that can be operated in a predefined period. This maximal set distinguish between different train types and the number of each type, and thus the heterogeneity of the operation.

We formalise the capacity determination as an optimisation problem with the objective of finding the maximal set of trains that minimises the average waiting time between trains of the same type. We test for feasibility of sets of trains by computing the time a set of trains occupies the infrastructure. A fixed timetable is not needed for the feasibility tests as samples of train sequences are examined. Furthermore, it is possible to determine a span of capacity depending on service constraints by adjusting input parameters.

We propose and test two different solution approaches to determine the capacity of railway infrastructure: an exact branch-and-bound algorithm and a tabu search heuristic. The applicability of the capacity determination framework is demonstrated on the railway network of Southwestern Denmark, with and without infrastructure upgrades. We analyse computation time, solution performance and sensitivity. The results illustrate the ability to obtain quality solutions for a real network within the order of minutes. As precise infrastructure characteristics are considered in junctions, the method is able to quantify the effect of the infrastructure upgrades, contrary to other assessment methods used in the long term (strategic) planning phase.

1. Introduction

Increasing railway traffic calls for a better utilisation of the existing infrastructure. However, the infrastructure capacity of existing or future railway networks is not easily determined or even defined.

Intuitively, we would like to define railway infrastructure capacity as the number of trains that can traverse the railway network in a given time period. However, this number does not only depend on the infrastructure itself, but also on characteristics of

* Corresponding author at: Ramboll Danmark A/S, Hannemanns Alle 53, 2300 Copenhagen S, Denmark.
E-mail address: lars.wittrup.jensen@gmail.com (L.W. Jensen).

https://doi.org/10.1016/j.trc.2020.102751
Received 26 November 2019; Received in revised form 15 July 2020; Accepted 8 August 2020
Available online 26 August 2020
0968-090X/

the rolling stock and operational requirements. Some factors that need to be considered in capacity determination of railway infrastructure are:

1. Signalling equipment and layout as well as operational rules impose minimum headways unique to each railway segment
2. Train services operated on a line usually have different stop patterns and are operated according to a timetable to serve passenger demand
3. Rolling stock types have different acceleration and braking capabilities as well as maximum speed

For a very simple type of operation where the effect of the above factors is minimised, the capacity is relatively straightforward to determine. E.g. if the minimum headway time of a given railway segment is known and the operation is homogeneous with only one train type and stopping pattern used, the maximum theoretical capacity of the given segment can be calculated as the time period considered divided by the minimum headway.

However, most of the time, operations are not homogeneous as demand must be met in the best way possible. To serve passengers travelling longer distances, skip-stop and/or high-speed services are introduced. These services usually operate on the same tracks as services stopping at all stations. In addition to different passenger services, freight trains may also be operated on the same tracks. As a result, trains might run with different speeds. Such variances in speed from train to train form a heterogeneous train operation that leads to longer line headway times. I.e. a fast train must depart later *not* to catch up with a slower train. Thus, only a reduced number of trains per hour can be operated with heterogeneous operation compared to a network with homogeneous train operation. Furthermore, the reduced number of trains is heavily dependent on the order of trains. E.g. bundling fast trains and slow trains will lead to a higher capacity than distributing fast and slow trains. On the other hand, bundling is not desirable from a passenger point of view as departures are distributed unevenly.

Existing methods to estimate the capacity of a railway system either neglect these complicating factors, or rely on additional information like timetables or fixed train sequences.

In this paper, we define the infrastructure capacity of a railway network by means of a maximal set of trains that can be operated on the infrastructure within a given time period given input parameters describing heterogeneity and maximum capacity occupation. Among all sets that fulfil these properties, we choose the one that minimises the average waiting time between trains of the same type. We check the feasibility of sets of trains by computing the time a set of trains occupies the infrastructure using the approach presented in Jensen et al. (2017). To solve the optimisation problem and find the capacity-representing set of trains, we use branch-and-bound and tabu search.

The approach presented in this paper is designed for capacity estimation in the strategic planning phase. This is the early planning phase in the planning of railway infrastructure and operations. Subsequent phases include the tactical planning phase and the operational phase. As an approach designed for the strategic phase, our approach uses only the infrastructure network and the type of trains that would potentially be run on the network as input. This makes it different from some other capacity estimation approaches designed for later planning stages, which can base their estimation on operational details like frequencies, train sequences, or even timetables that only become known later in the planning process. Compared to other approaches for the strategic planning phase, the approach presented in this paper still takes important infrastructure constraints at junctions into account contrary to other strategic assessments methods.

In Section 2, we review literature to determine capacity assessment approaches for both the strategic and the tactical planning phase. In Section 3, we introduce the capacity determination problem, and in Section 4, we present our solution approaches. We conclude the paper with a case study and computational results in Section 5

2. Literature review

2.1. Capacity terminology

Within the railway field, there already exist several methods to determine and analyse railway capacity. From publication to publication, capacity terms used may differ. Therefore, we first give some definitions of the main capacity terms used in this paper before reviewing the literature. Our definition of infrastructure capacity in Section 1, can be divided into theoretical, maximum and practical capacity. We define these terms as:

> Theoretical capacity: the maximum number of trains that can traverse a given part of the network in a predefined time period. The operation must be completely homogeneous with a single ideal train type with no supplements of any kind.
>
> Maximum capacity: the maximal set of trains that can be handled in a given time period while considering the actual train mix where train types may have different stopping patterns and maximum speeds resulting in heterogeneous operation. Buffer times are not included in headways between trains.
>
> Practical capacity: the maximal capacity that can actually be used in a given time period to obtain a robust and stable operation. Buffer times are added to the minimum headway time between trains to obtain planning headways that provide stability and robustness.

Furthermore, we use the term *capacity occupation* throughout the paper to describe the time a set of trains in a given sequence occupies the infrastructure (i.e. the makespan of the train sequence). This term is also known as infrastructure occupation, but has many synonyms in the literature as described by Bešinović and Goverde (2018). The ratio between capacity occupation and a predefined time period, in which the set of trains should operate, is defined as the *capacity occupation rate*. The term capacity

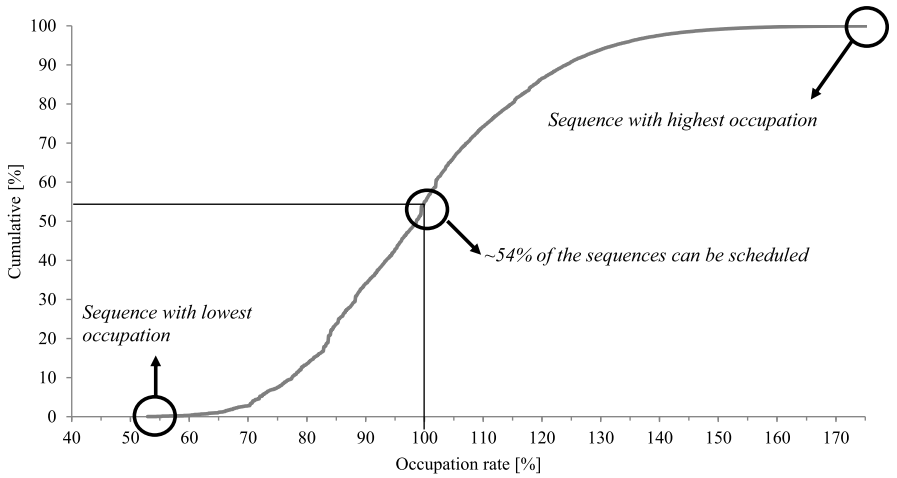


Fig. 1. Example cumulative distribution of capacity occupation.

consumption is also frequently used synonymously with capacity occupation. However, this term is defined by UIC (2004) as the time a train sequence (compressed timetable) occupies the infrastructure including additional buffer time to absorb delay. Thus, capacity consumption is not synonymous with capacity occupation (or infrastructure occupation). We therefore avoid using the term in this paper, except when referring to existing literature in this section.

2.2. Capacity analysis methods

We now review literature related to analyses of railway capacity. One of the most well-known methods for analyses of capacity is the UIC 406 method (UIC, 2013), which is designed for use in the (late) tactical planning phase, when the timetable is already known. This method can be used to calculate the capacity occupation on line segments and routes based on a given timetable. The timetable sets out the order of trains and the UIC 406 method is therefore able to give estimates on the capacity consumed by a given set of trains in a specific order. The simplicity of the method has entailed its use throughout Europe.

Jensen et al. (2017) expand the UIC 406 method to determine the capacity consumption and infrastructure/capacity occupation (as defined by UIC, 2004) of a set of trains in a railway network rather than segments of the network. Since the capacity occupation by the trains depends on the sequence in which they traverse the network, Jensen et al. (2017) describe the capacity occupation not by a single value but by a distribution which describes which percentage of the sequences that can be scheduled within a given capacity occupation rate. An example is given in Fig. 1.

To compute the capacity occupation of a set of trains, Jensen et al. (2017) generate and schedule all train sequences if the set of trains is comparatively small, or a sample of them if the instance is larger. Train sequences yielding capacity occupation rates of 100% or close will not be feasible in practice due to the stochastic behaviour of railway systems (resulting in delays and thereby higher occupation). To account for the stochastic behaviour, they build upon the basic model to include robustness considerations by stochastic simulation (see Jensen et al., 2017) and thus obtain capacity consumption values. In this paper, we use the methodology presented in Jensen et al. (2017) to estimate the capacity occupation for sets of trains as a feasibility check in our solution approach.

2.3. Timetabling methods

Since a timetable decides the usage of railway capacity, timetabling methods can be used to estimate the feasibility of scheduling a given set of trains. Within optimisation and scheduling of railway operations, the timetabling problem is used to generate a feasible (and possibly optimal) timetable when solved. Both *nominal* and *robust* versions of timetabling are considered in the literature, see e.g. the overview given in Cacchiani and Toth (2012). In the nominal problem, the goal is to schedule all the trains subject to the given constraints. In the robust problem, the goal is to schedule the trains such that a robust timetable is obtained. That is, a timetable that can absorb smaller delays using the supplements included in the timetable to prevent delay propagation. The most common model for finding periodic timetables is the *periodic event scheduling problem (PESP)* where train departure and arrival times repeat after a certain time period (i.e. cyclic timetables). Another model for timetabling is proposed by Polinder et al. (2020) who defines the Strategic Passenger-Oriented Timetabling (SPOT) problem. Compared to PESP the proposed model is aimed at strategic planning instead of tactical planning to provide a timetable outline using less input.

While timetabling methods cannot directly be used to estimate the capacity, they can be combined with methods to calculate capacity occupation with a given timetable to estimate capacity of the system (e.g. the UIC 406 method). An example of this combination of timetabling and methods for calculation of capacity occupation is proposed by Zhang and Nie (2016) who use a minimum cycle time calculation (MCTC) model based on the PESP to assess the capacity occupation for a given train line plan

taking train mixes (different train types) and overtaking into account. While the approach by Zhang and Nie (2016) provides quality results, the computation times are long in the order of days, and the input required makes the approach more suitable for the tactical planning level. This exemplifies the challenges with using timetabling methods for strategic capacity evaluation. In general, the resulting capacity will rely heavily on the input given to the timetabling method and thus the structure of the generated timetable. This timetable will be quite specific which is usually not desirable in the strategic planning phase. Furthermore, long computation times of timetabling methods may make it undesirable to run them several times in a loop to find the maximum capacity.

In addition to timetabling methods, several models and tools exist that can be used to saturate existing timetables (which can be empty) with additional trains, locate bottlenecks in the network and estimate stability or robustness of a generated timetable. Such models can be used to estimate the number of trains that can traverse a given network under a set of operational constraints. However, as also described above, these models are more suitable on a tactical level as they require a significant amount of input. For an overview of these models and tools we refer to Abril et al. (2008).

2.4. Capacity determination in early planning phases

We now describe approaches which address the problem of capacity determination in the strategic phase; the subject of this paper. An overview of the reviewed approaches is given in Table 1. The capacity determination approaches described in the following all define the capacity in a similar form, which may be summed up as the maximum number of trains that can traverse the network or corridor in a given time period. Some approaches distinguish between different train types while others use an average train type. Furthermore, some include a buffer in the determined capacity to mitigate delay propagation (the practical capacity), some measure the quality of the determined capacity, while others calculate the determined capacity as either the maximum capacity or theoretical capacity without considering delays.

De Kort et al. (2003) uses a probabilistic approach using max-plus algebra to determine the capacity of a simple railway infrastructure with single track tunnels. The approach is based on the bottleneck approach which determines the critical section (bottleneck) that limits the capacity of the system considered. The approach does not explicitly account for different train types. However, it is possible to implement this by using the probability that a train will be of a given type.

Inspired by the work of De Kort et al. (2003) and Mussone and Wolfler Calvo (2013) present a framework to maximise the total number of trains in the railway system based on train conflict probabilities, where different train types can be considered. De Kort et al. (2003) ans Mussone and Wolfler Calvo (2013) are able to include knock-on delays (delay propagation) and route conflicts (junction constraints) in the railway system in a simple way. The approaches by De Kort et al. (2003) and Mussone and Wolfler Calvo (2013) do not depend on the exact train sequence, but rather consider a weighted average. Thus, in a network with heterogeneous operation the average maximum capacity is obtained, and this capacity might therefore be increased by bundling trains.

Burdett and Kozan (2006) describe and discuss terms for absolute capacity (maximum capacity) and utilisation levels based on sectional running times of train types, dwell times and proportional mix of (different) train types. Based on these, they propose an optimisation model to maximise the theoretical number of trains that can traverse a railway network. The model used to determine the capacity for a network is non-linear and is therefore only solved to a local maximum (Burdett and Kozan, 2006). The approach by Burdett and Kozan (2006) is quite flexible and is able to capture a large number of operational characteristics. Furthermore, the approach is able to give a lower and an upper bound for the determined capacity. Bevrani et al. (2017b) use the model by Burdett and Kozan (2006) in a case study in Iran to provide decision support on future capacity extensions. Burdett (2015b) extend the earlier work in Burdett and Kozan (2006) from a single objective to a multi-objective model to include a variable train mix instead of a fixed train mix. The multi-objective problem is formulated to regulate competition between train services, trains types and network corridors resulting in Pareto optimal solutions. Burdett (2015a) extends the work of Burdett and Kozan (2006) further by incorporating complex train paths. Burdett (2016) describes a model to obtain a given capacity while minimising spending and a model where capacity is optimised within a given budget based on the work in Burdett (2015b). Similar to the model by Burdett (2015b), Bevrani et al. (2017a) develop a capacity assessment model for multi-modal transport systems. Common for the approaches described is the representation of the network as segments each with a capacity constraint. However, capacity constraints in junctions, due to route conflicts, are not included. This may lead to an overestimation in the determined network capacity.

Jensen (2015a) develops a framework to determine the number of trains that can traverse a network within a given time period. Both the maximum capacity and the practical capacity can be determined. Jensen (2015a) represents an early version of the present paper. Compared to this early work, we have improved the problem formulation to provide results that are more realistic and practically useable compared to the initial test results in Jensen (2015a). Furthermore, the greedy heuristic used in Jensen (2015a) has been replaced by branch-and-bound and tabu search.

Related to determination of capacity on a strategic level is also the work of Harrod (2009). Harrod (2009) uses a linear programming model to assess the effect on capacity from introducing passenger trains on single track freight lines resulting in speed differentials between trains. A weighted sum of train trips as well as delays incurred are included in the objective function to indicate the quality of the determined capacity. The primary focus is on how fast passenger trains affect the operation of freight trains. The specific model used is limited to various single track line configurations.

Other capacity determination methods take a macroscopic view that requires less data and yields fast computation times, but produces less precise results. Examples are Yaghini et al. (2014) where line capacity is evaluated using an integer programming model. The problem is formulated as a multi-commodity network design model. The inputs of the model are the characteristic of each train type and railway line attributes. The model determines railway capacity based on train type mixes and thus takes

Table 1
Summary of capacity determination approaches and their main characteristics (denoted by •) as described in the text. Parentheses imply only partial applicability.

Method	Timetable independent	Different train types	Practical capacity[a]	Span of capacity	Junction constraints	Execution time[c]
Timetabling + capacity evaluation		•	(•)		(•)	Slow
Zhang and Nie (2016)	(•)	•	(•)	(•)		Slow
De Kort et al. (2003)	•	(•)	(•)			Fast
Mussone and Wolfler Calvo (2013)	•	•	(•)		•	Fast
Burdett et al.[b]	•	•	(•)	•		Fast
Yaghini et al. (2014)	•	(•)	(•)			Fast
Reinhardt et al. (2017)	•		(•)			Fast
The approach in this paper	•	•	(•)	•	•	Fast

See also the following notes.

[a]As none of the methods uses integrated stochastic simulation, only partial applicability is assigned.

[b]Covers the literature: Burdett and Kozan (2006), Burdett (2015a,b, 2016) and Bevrani et al. (2017a,b). Common for the approaches is the base capacity model used which is extended and improved in various ways. Due to the base model used, the characteristics are essentially the same across the listed literature albeit with different purposes and contributions.

[c]Estimated comparison of model execution times. Slow: in the order of hours or days, Fast: in the order of seconds or minutes.

heterogeneity into account. Other examples are Reinhardt et al. (2017) who propose a model for strategic long-term assessment of railway freight capacity using a macroscopic model.

Table 1 gives a comparison and summary of the capacity determination methods described in this section. In conclusion, the capacity determination framework, we propose in this paper complements and extends existing research by determining a span of capacity (e.g. practical and maximum capacity as well as bundled and distributed operation) taking different train types and thus train heterogeneity into account. Additionally, essential and precise infrastructure characteristics are considered, especially at junctions, and no timetable is needed thus making the approach suitable for fast and high quality assessments in the strategic planning phase. Furthermore, the capacity determination framework can easily be altered to adapt for later planning phases by using an approach for calculation of capacity occupation that takes additional input available later in the planning process and thus provide solutions that are adapted to more complex constraints.

3. The capacity determination problem

We now introduce the capacity determination problem that is solved using the approaches described in Section 4. Let N be a network that models railway infrastructure, represented by a mesoscopic infrastructure model where nodes and edges respectively represent switches and tracks used in normal operation. This paper develops a framework to determine the capacity of N, which, colloquially speaking, is the number of trains that can traverse the network in a given period of time T.

3.1. Infrastructure capacity definition

If N is a simple set of railway segments and all trains that run on N have the same speed profile, the number of trains that can traverse the network in time period T can be determined by dividing T by the maximum headway between two trains.

However, defining the capacity of a network N is much less straightforward if the trains that are supposed to run on N are heterogeneous in the sense that they have different speed profiles and/or cross N on different routes.

To see this, consider a heterogeneous set of trains S. Each train follows a predefined route on N. Furthermore, minimum headway times and running times are defined for each edge (including junctions) and train type.

We would like to refer to the time it takes for these trains to traverse the network as the capacity occupation by the set S. The capacity occupation in relation to the time period is referred to as the capacity occupation rate. The capacity occupation by S depends crucially on the sequence in which the trains traverse the network. E.g. even if all trains in the set run on a single railway segment and we have two fast trains and two slow trains on this segment, then sending the two fast trains first and then the two slow trains will lead to a lower occupation than scheduling first a slow train, then a fast one, then a slow one, and a fast one. This follows from the extra time that has to be planned to make sure that the fast trains do not catch up with the slow ones.

Thus, for *each sequence* s of trains, we call the time that this sequence needs to traverse the network if scheduled as close as operationally feasible, the *capacity occupation by sequence* s.

For a *set of trains* S *and a percentage* p, we follow the approach of Jensen et al. (2017) and define the *capacity occupation by the set of trains* S as the time in which p percent of all sequences of trains in S can traverse the network. For $p = 0$ we consider only the sequence with minimum capacity occupation in which slow and fast trains are bundled. For $p = 100$, the worst-case sequence with maximum makespan determines the capacity occupation. A natural choice would be $p = 50$, which means that we determine the capacity occupation by S as the time span in which at least for 50% of the possible sequences from S it would be possible to traverse the network.

We represent the *capacity* of a network by a maximal set of trains which can be operated on it within a time period length T. More precisely, the $(p, C_{\max})$-capacity of a network is represented by a maximal set of trains with a capacity occupation rate

$C(X,p)$ of at most $C_{\max}$ where X is a vector representing the number of trains per train type in S. Thus, besides the parameter p, which describes the percentage of train sequences we consider to determine the capacity occupation by a set S, we parametrise the capacity by an additional parameter $C_{\max} \in [0,1]$. $C_{\max}$ is thus the maximum capacity occupation rate that we allow for the period T. With $C_{\max} = 1$, we may fully use the period length, T. If $C_{\max} < 1$, there is some buffer, to account for delays during operations. The requirement 'maximal' means that if we add one more train to the set, its capacity occupation rate is higher than the maximum capacity occupation rate, $C(X,p) \leq C_{\max}$.

Of course, there is a multitude of different maximal sets of trains with capacity occupation rate $C(X,p)$ of at most $C_{\max}$. In particular, for each train type that runs on the network, we have such a set. However, in our opinion, real-life capacity of a network is best represented by a capacity-representing set of trains that is close(r) to real-life usage of the network.

We thus require that each train type that would run on the network is contained at least once in the capacity-representing set, and that the number of trains of a certain type is the same in both directions. Among all maximal sets of trains that fulfil these requirements, we choose one that favours solutions with high service quality for passengers and freight: we minimise the weighted average waiting time between two departures of the same train type.

3.2. Problem formulation

The $(p, C_{\max})$-capacity can thus be described as solution to the following optimisation problem:

$$\begin{aligned}
&\text{Minimise} && \frac{T}{2} \cdot \sum_{i=1}^{2n} \frac{w_i}{x_i} && \text{[Minimise weighted average waiting time]}\\
&\text{Subject to} && x_i \geq 1 \ \forall i \in \{1,2,\dots,n\} && \text{[Minimum threshold on trains]}\\
& && C(X,p) \leq C_{\max} && \text{[Maximum capacity occupation rate]}\\
& && x_i = x_{n+i} && \text{[Directional balance]}
\end{aligned}$$

$X = (x_1, \dots, x_i, \dots, x_{2n})$ are the decision variables, i.e. a vector representing the number of trains of each train type x_i present in the set of trains S. x_i is the number of trains of train type i in one direction for $i = 1, \dots, n$, while x_{n+i} is the number of trains of train type i in the opposite direction for $i = 1, \dots, n$. The constraint $x_i = x_{n+i}$ ensures balance (or symmetry) between the two directions. $C(X,p)$ is the function that calculates the p-capacity occupation rate in a given time period T given a set of trains S represented by the vector X, and a percentile p. To give an indication on the waiting time per train type, we estimate waiting time as if departures of trains of the same type were equally spread over the period, and assume that arrival of passengers and freight is uniformly distributed over the period. (For more information on the difference of actual and hidden waiting time see Ingvardson et al., 2018). Then the estimated waiting time between train of type i can be calculated as $1/2 \cdot T/x_i$. By w_i we denote the weight of train type i with $w_i > 0$, where a higher value defines a higher weight. The train weights make it possible to prioritise certain train types over others, e.g., because they need to be operated more often. Note that the factor T/2 in the objective function is constant, and could thus be omitted. However, we included it in the objective function, so that the objective value can be interpreted as the weighted average waiting time. Ultimately, we obtain a convex objective function that deflates the value of a train as more trains are added per train type.

Although the capacity determination problem as is presented by the mathematical model above looks simple, a closed-form expression cannot be constructed for the evaluation of the capacity occupation rate $C(X,p)$ in networks with heterogeneous operation. As described earlier in Section 3, this is caused by the heterogeneous operation resulting in different capacity occupation rates depending on the sequence in which the trains traverse the network. As described in Section 4.3, we use the approach by Jensen et al. (2017) to estimate $C(X,p)$ by sampling random permutations of train sequences. Depending on the size of the set of trains S, a large sample size may be needed resulting in long computation times each time the capacity occupation rate $C(X,p)$ is evaluated.

4. Solution approaches

In this section, we describe two approaches to determine the capacity of a network, expressed by a capacity representing set of trains S, by finding an (optimal) solution to the optimisation problem described in Section 3.

The first approach to solve the optimisation problem is a branch-and-bound approach, as such approaches are able to handle non-linear problems and are well-suited for discrete problems. Branch-and-bound (with local search heuristic) is also used in many train dispatching and timetabling problems, see Karoonsoontawong and Taptana (2017). A branch-and-bound approach can, however, be time-consuming as the number of solutions in the problem instances grows exponentially with the number of trains. In addition to the branch-and-bound approach, we also propose in the following a tabu search meta-heuristic to handle larger problem instances.

As described in the previous section, the constraint $C(X,p) \leq C_{\max}$ in our problem is expensive to evaluate as the capacity occupation rate $C(X,p)$ in general has to be calculated each time a new solution is examined. To mitigate this, we define the following properties, that is used in the approaches to reduce the computational running time:

Property 1. If an extra train is added to a network, the capacity occupation will never decrease compared to the capacity occupation in the previous solution (without the extra train)

Property 2. Likewise, if a train is removed, the capacity occupation will never increase

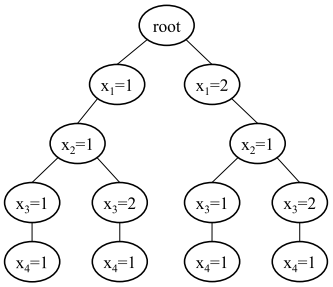


Fig. 2. Example of solution (enumeration) tree for the problem.

In Sections 4.1 and 4.2 we describe the branch-and-bound algorithm and the tabu search. Both approaches make use of a module for evaluation of the capacity occupation rate $C(X, p)$. We name this evaluation module the *capacity occupation evaluation (COE) module*. This module is discussed in Section 4.3. Furthermore, a module based on binary search to compute lower and upper thresholds per train type for both approaches is described in Appendix. Together the approaches and modules comprise our capacity determination framework.

4.1. Branch-and-bound

To obtain an exact solution to the optimisation problem, we use a branch-and-bound algorithm tailored for our problem. Each node in the solution (enumeration) tree represents a fixed number of trains of train type x_i, see Fig. 2 for an example. Using branch-and-bound, we construct only the parts of the tree containing feasible solutions having a lower bound better than the incumbent solution (the current best solution). Starting from the root node of the tree, we branch depth-first by adding nodes (and connecting edges) to a stack (LIFO) data structure. This ensures that leaf nodes in the tree (representing complete solutions) are reached early in the process providing new feasible solutions and thus potentially better upper bounds faster. Branching is only done if the subtree may contain the optimal solution.

Algorithm 1: Pseudocode for branch-and-bound algorithm.

Data: x_i^{min}, x_i^{max} $\forall i \in \{1, 2, ..., n\}$: lower and upper threshold for trains x per train type i;
w_i $\forall i \in \{1, 2, ..., n\}$: train type weights;
C_{max}: threshold for capacity occupation, p: percentile;
Result: Solution $X^* = (x_1^*...x_i^*...x_n^*)$ maximum number of trains that can be added
`/* Start solution */`
$nodeStack \leftarrow$ add node v with $x_i = x_i^{min}$ $\forall i \in \{1, 2, ..., n\}$;
while *nodeStack is not empty* **do**
  $v \leftarrow$ take top solution node of $nodeStack$;
  if *v is a leaf node in the solution tree* **then**
    `/* v and preceding nodes in the solution tree comprising a complete solution */`
    Update incumbent, X^*, if solution, v, is better (wrt. objective function);
  else
    `/* v comprises a partial solution, check if branch nodes are feasible using binary search */`
    $\hat{x}_{i+1} \leftarrow binarySearch(x_{i+1}^{min}, x_{i+1}^{max})$; `// Find the feasible branch with the highest number of trains` $\hat{x}_{i+1}$
    Update incumbent, X^*, if a better solution was found during $binarySearch$;
    for $x = x_{i+1}^{min}$ **to** $\hat{x}_{i+1}$ **do**
      `/* Add feasible branch nodes to` $nodeStack$ `if the best solution in the branch has a better objective than the incumbent */`
      if $getLowerBound(x\ as\ child\ of\ v) < objective(X^*)$ **then**
        Add new node as child to v with $x_{i+1} = x$ to $nodeStack$;
return X^*

In the construction of the tree, nodes can be pruned by either infeasibility (the capacity occupation rate for the set of trains $C(X, p)$ exceeds the maximum capacity occupation rate $C_{\max}$) or bound ($f(X) > f(X^*)$ where X is the candidate solution and X^* is the current best solution). To minimise the number of times the capacity occupation evaluation (COE) module is executed to check for feasibility of the constraint $C(X, p) \leq C_{\max}$, the following three features have been implemented:

- Binary search in branches to determine feasible and infeasible branches (as described in pseudocode in the second part of Algorithm 3). E.g. for a tree node, we do not need to check a child (branch) for feasibility if the child node has one train less than another child node that has already been deemed feasible. This follows from the properties described in Section 4.

- List with non-dominated feasible solutions, where a solution a dominates another solution b if $x_i^a \geq x_i^b \forall i$ and $\exists i^*$ s.t. $x_{i^*}^a > x_{i^*}^b$ with x_i denoting the number of trains per train type i
- List with non-dominated infeasible solutions, where solution a dominates solution b if $x_i^a \leq x_i^b \forall i$ and $\exists i^*$ s.t. $x_{i^*}^a < x_{i^*}^b$ with x_i denoting the number of trains per train type i

The three features listed above effectively reduce the number of times the COE-module is executed to evaluate $C(X, p)$ by taking advantage of the two properties defined in Section 4. The two lists come at the expense of increased memory usage and CPU time to maintain and search the lists. However, the additional CPU time and memory used are substantially smaller than the CPU time and memory required to run the COE-module additional times.

4.2. Tabu search

The branch-and-bound algorithm described in Section 4.1 provides optimal solutions to problem instances at the expense of computational time. As described earlier, the computational effort makes the branch-and-bound algorithm unsuitable for large problem instances.

To handle large problem instances within reasonable time, we employ a tabu search metaheuristic that can quickly provide good solutions to even large problems. Tabu search is characterised by the use of adaptive memory structures to escape local optima in order to find global optima. This adaptive memory allows the efficient search of the solution space (Glover and Laguna, 2013).

To reduce the number of times $C(X, p)$ is evaluated using the capacity occupation evaluation (COE) module, we reuse the strategy of saving dominating feasible and infeasible solution from the branch-and-bound search. This fits well with the tabu search algorithm, as the feasibility list can also be used as long-term memory for the tabu search. The only exception being that all feasible solutions are saved, not just the dominating feasible solutions.

Our customised tabu search algorithm is given in Algorithm 2 and consists of four main memory structures or methods:

- The tabu list (short-term memory)
- The neighbourhood search in the form of $generateNeighbourhoodCandidate$
- The diversification strategy in the form of $generateDiverseCandidate$
- The list with all feasible solutions generated (long-term memory) used by $generateNeighbourhoodCandidate$ and $generateDiverseCandidate$

Algorithm 2: Tabu search

Data: $x_i^{min}\ \forall i \in \{1, 2, ..., n\}$: lower threshold for trains x per train type i;
$w_i\ \forall i \in \{1, 2, ..., n\}$: train type weights;
Result: Solution $X^* = (x_1^* ... x_i^* ... x_n^*)$
$x_i \leftarrow x_i^{min}\ \ \forall i \in \{1, 2, ..., n\}$;
while $iterationCount < maxIterations$ ***and*** $timer < maxTime$ **do**
  $X \leftarrow generateNeighbourhoodCandidate$;
  if $objective(X) < objective(X^*)$ **then**
    $X^* \leftarrow X$;
  $tabuList.add(X)$;
  $tabuList.removeOldestSolutions(maxTabuSize)$;
  if *generateNeighbourhoodCandidate failed* ***or*** *iteration = restartInterval* **then**
    $X \leftarrow generateDiverseCandidate$;
return X^*;

We now describe each of the above items followed by an example.

The tabu list prevents the neighbourhood search in choosing a candidate solution that is tabu (in the tabu list). Every candidate solution found in each iteration is put in the tabu list (see Algorithm 2). The tabu list has a maximum size. For our case in Section 5, we use a list size of 25 solutions. When the list size is exceeded, the oldest solution is removed from the tabu list. This solution is then no longer tabu and can potentially be revisited by the search. The tabu list thus serves as short-term memory, and ensures that the search is guided towards new areas of the solution space.

The $generateNeighbourhoodCandidate$ returns a candidate solution X^S from the neighbourhood of the current solution X. The neighbourhood of a solution consists of all solutions for which the following conditions hold true:

- for each train type i the number of trains is at most increased or decreased by a maximum of 1, i.e. $|x_i - x_i^S| \leq 1$
- the sum of trains in the neighbourhood solution is restricted to a maximum increase or decrease of 1 train, i.e. $\sum_{i=1}^{n} |x_i - x_i^S| \leq 1$

To return the best feasible solution in the neighbourhood that is not tabu, we use a greedy (descent) method. The greedy method returns the feasible solution in the neighbourhood that has the lowest objective and is not tabu. If there are no feasible solutions or the solutions are tabu in the neighbourhood, the $generateNeighbourhoodCandidate$ fails.

	Best solution $(x_1, x_2, x_3, x_4, x_5, x_6)$ Objective value	Solution generated $(x_1, x_2, x_3, x_4, x_5, x_6)$ Objective value	Algorithm/list used	Tabu list (max. 5 entries)	Infeasibility list	Feasibility list
Iteration 1		(1,1,1,1,1,1) 480	Minimum solution	Add (1,1,1,1,1,1)		Add (1,1,1,1,1,1)
Iteration 2	(1,1,1,1,1,1) 480	(1,1,1,1,1,2) 405	GNC	Add (1,1,1,1,1,2)		Add (1,1,1,1,1,2)
Iteration 3	(1,1,1,1,1,2) 405	(1,1,1,1,2,2) 360	GNC	Add (1,1,1,1,2,2)		Add (1,1,1,1,2,2)
Iteration 4	(1,1,1,1,2,2) 360	(1,2,1,1,2,2) 330	GNC	Add (1,2,1,1,2,2)	Add (1,1,1,2,2,2)	Add (1,2,1,1,2,2)
Iteration 5	(1,2,1,1,2,2) 330	(1,3,1,1,2,2) 320	GNC Infeasibility list	Add (1,3,1,1,2,2)	Add (1,2,2,1,2,2)	Add (1,3,1,1,2,2)
Iteration 6	(1,3,1,1,2,2) 320	None	GNC Infeasibility list, Tabu		Add (1,3,2,2,1,2) Add (2,2,1,1,2,2)	
Iteration 7	(1,3,1,1,2,2) 320	(1,2,1,1,1,2) 375	GDC	Add (1,2,1,1,1,2) Remove (1,1,1,1,1,1)		
Iteration 8	(1,3,1,1,2,2) 320	(1,1,2,1,2,2) 330	GNC Tabu	Add (1,1,2,1,2,2) Remove (1,1,1,1,1,2)		Add (1,1,2,1,2,2)
Iteration 9	(1,3,1,1,2,2) 320	(2,1,2,1,2,2) 315	GNC	Add (2,1,2,1,2,2) Remove (1,1,1,1,2,2)		Add (2,1,2,1,2,2)
Iteration 10	(2,1,2,1,2,2) 315	Termination	Stop criterion met			

Fig. 3. Example of iterations in the tabu search for six train types with weights $w_1 = 1, w_2 = 2, w_3 = 2, w_4 = 3, w_5 = 3, and\ w_6 = 5$. GNC: $generateNeighbourhoodCandidate$ and GDC: $generateDiverseCandidate$. Infeasibility list or Tabu: the best solution generated is either evaluated as infeasible, in the infeasible list or in the tabu list.

After a predefined number of iterations, the $generateDiverseCandidate$ is executed to restart the tabu search algorithm. The $generateDiverseCandidate$ is also executed if $generateNeighbourhoodCandidate$ fails (i.e. the search is trapped in a local minimum). The purpose of $generateDiverseCandidate$ is to construct a new diverse candidate solution from existing solutions in order to explore new parts of the solution space. Based on Glover and Laguna (2013), our diversification strategy makes use of the feasibility list as long-term frequency-based memory. All feasible solutions encountered during the search are included in this list with a frequency parameter that describes how many times the same solution is encountered.

The diversification works by constructing a candidate solution made out of different parts of random length. The number of parts for this division is case specific. For our case, a division into three parts has been used. Each of these parts is populated with trains from a random solution selected out of the three feasible solutions which have been encountered the least (i.e. lowest solution frequency) from the feasible solutions list. This restricted random selection, in contrast to a uniform random selection of feasible solutions, ensures that the search is not cycling, yet still directed towards unexplored solutions.

An example of the tabu search and solutions generated by $generateNeighbourhoodCandidate$ and $generateDiverseCandidate$ is given in Fig. 3 for six arbitrary train types with a tabu list size of 5.

In the example $generateNeighbourhoodCandidate$ is able to generate the best solution in the neighbourhood in iterations 1–3 and 9. In both iteration 4 and 5, the best solution in the neighbourhood is found to be infeasible (i.e. $C(X, p)$ exceeds $C_{\max}$) by the COE-module or the infeasibility list (i.e. the capacity occupation is above the threshold, $C_{\max} \cdot T$). In iteration 6, a local minimum is reached as $generateNeighbourhoodCandidate$ cannot find a feasible solution that is not tabu in the neighbourhood. Thus, $generateDiverseCandidate$ is executed and returns the solution $(1, 2, 1, 1, 1, 2)$ based on three randomly selected feasible solution $((1, 2,), (, 1, 1,), (, 1, 2))$. In iterations 7–9, $generateNeighbourhoodCandidate$ incrementally improves this solution to obtain a solution that is marginally better than the best solution found before $generateDiverseCandidate$ was executed. In iteration 10, the search is stopped as the predefined stop criterion is met.

4.3. Computing the capacity occupation of a set of trains

To evaluate the capacity occupation rate $C(X, p)$ for a set of trains in our capacity determination framework, we use the approach described in Jensen et al. (2017) as the basis for our capacity occupation evaluation (COE) module (see also Section 2). To estimate the capacity occupation rate $C(X, p)$ of a network with respect to a set of trains, we generate all possible train sequences if the set of trains is small, or use only a sample if the set is (too) big, since the amount of possible permutations of the set of trains into train sequences grows factorial with the number of trains. Furthermore, trains sequences which are not desirable from a demand point of view may discarded (Jensen et al., 2017). An example of this are passenger trains of the same type following immediately after each other. Such a sequence of trains is (usually) not desirable as it results in a bad distribution of departures for a given station or destination throughout the time period.

Note that this evaluation method may be replaced by any other approach to compute the capacity occupation rate by a set of trains on a network, $C(X, p)$. Alternative models may also target later planning phases by incorporating more input data by focussing on a more specific (fixed) timetable. However, any capacity evaluation model used to evaluate $C(X, p)$ must fulfil the two properties set out in Section 4 to be used in the framework. Many possible models do not fulfil these properties. For instance, in models where scheduling decisions are made by slowing down fast trains and/or re-routing trains, the addition of a train might lead to a lower occupation of capacity. The model from Jensen et al. (2017) that we use to evaluate the capacity occupation rate $C(X, p)$ fulfils these properties.

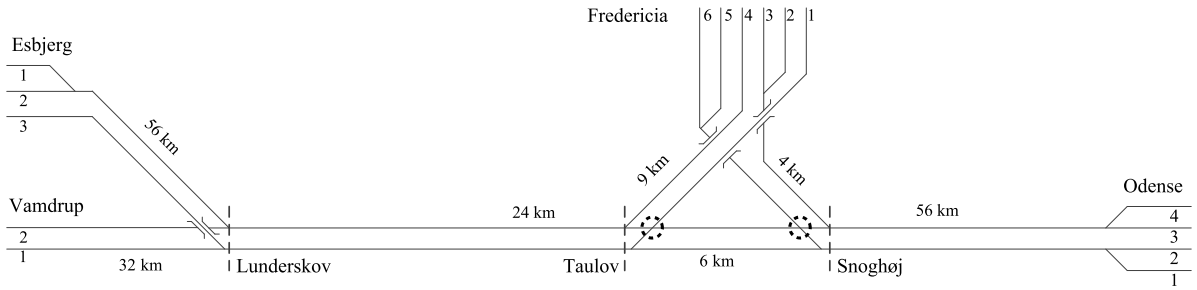


Fig. 4. Schematic plan of the network of Southwestern Denmark. Dashed circles indicate junctions that are upgraded in the upgrade scenario. In the base scenario the highlighted junctions are at-grade junctions, in the upgrade scenario the junctions are upgraded to grade separated junctions.

Table 2
Train types in the network with route used and weights assigned. Values in parenthesis denote the track used at the given station.

Train type	Route	Weight
Express	Odense (4) → Fredericia (2)	3
↪	Fredericia (6) → Odense (2)	3
Intercity-A	Odense (4) → Fredericia (2)	2
↪	Fredericia (6) → Odense (2)	2
Intercity-B	Odense (3) → Esbjerg (1)	2
↪	Esbjerg (3) → Odense (1)	2
Regional-A	Odense (4) → Fredericia (3)	1
↪	Fredericia (5) → Odense (1)	1
Regional-B	Fredericia (4) → Esbjerg (2)	1
↪	Esbjerg (3) → Fredericia (1)	1
Freight	Odense (3) → Vamdrup (2)	4
↪	Vamdrup (1) → Odense (2)	4

5. Case study and computational experiments

In this section, we apply the proposed capacity determination framework to the network used in Jensen et al. (2017). The network consists of 161 kilometres of double track, at-grade and grade separated junctions and four line end stations in Southwestern Denmark. The network is depicted schematically in Fig. 4 with tracks used in normal operation.

Table 2 provides an overview of the train types that should traverse the network and the route used. The train types form a heterogeneous operation. E.g. between Odense and Snoghøj the slowest train type (Regional-A) is 17 min slower than the express train type that takes 31 min between Odense and Snoghøj. For the case a minimum (block) headway time of 150 s is used.

During a period of one hour, the goal is to estimate the number of trains the network can handle using the framework proposed in this paper. For the optimisation, weights have been derived by taking the following into consideration: (1) the (desired) priority of the train type, and (2) how much capacity the train type consumes, i.e. a fast or slow train (type) consumes more capacity in the set of trains than a train (type) that is average in terms of speed. The latter can be difficult to account for before running the model and weights may have to be adjusted before running the model again to obtain the desired prioritisation of train types (the sensitivity of train weights are further analysed and discussed in Section 5.3). For this case study, we slightly favour express trains over intercity and regional trains as defined in Table 2. Furthermore, freight trains have also been given some priority. However, the main reason for the high weight is that the freight trains are slower than the other trains (except regional-A) making it is necessary to assign a higher weight as described above.

This case study serves as a basis for practical implications and for theoretical experiments to explore how the capacity determination framework performs. The framework and module for evaluation of capacity occupation rate $C(X, p)$ (COE) are executed on a Windows 10 desktop PC with an AMD Ryzen 7 1700 (3.7 GHz) CPU and 16 GB of RAM. Both the framework and the COE-module (developed in Jensen et al., 2017) have been implemented in C++. The model implementations do not take advantage of multi-threading.

First, we investigate the results of the base infrastructure using the branch-and-bound search, including the performance of the search (Section 5.1). Subsequently, the capacity of the network is analysed when the infrastructure is upgraded (Section 5.2). Section 5.3 contains a sensitivity analysis of train weights, while Section 5.4 explores the performance of the tabu search.

5.1. Results using branch-and-bound

Using the branch-and-bound algorithm, we have obtained the capacity results based on three different thresholds for maximum capacity occupation rate (capacity threshold), $C_{\max}$. The thresholds examined are 75%, 84% and 100% respectively. The 75% is the UIC recommendation for line sections with mixed traffic in the peak hours (UIC, 2013). 100% is the maximum that can only

Table 3
Results using branch-and-bound algorithm. COE: capacity occupation evaluation module. Nodes refer to the nodes in the enumeration tree described in Section 4.1.

C_{max}	Operational type (percentile)	Obj. value (optimal)	CPU time [s]		COE calls	Possible solutions	Theoretical number of nodes	Nodes evaluated [%]
			Total	COE				
75%	Bundled (5)	37 800	25.5	24.6	52	288	478	7.7%
	Near bundled (25)	43 200	14.4	14.0	26	4	17	41%
	Part. distributed (50)	No solutions						
	Distributed (75)	No solutions						
	95	No solutions						
84%	Bundled (5)	33 300	51.0	49.7	84	4 600	6 746	1.1%
	Near bundled (25)	37 800	28.6	27.8	51	432	478	4.8%
	Part. distributed (50)	43 200	16.5	16.1	50	6	21	23.8%
	Distributed (75)	No solutions						
	95	No solutions						
100%	Bundled (5)	27 000	343.9	337.9	403	78 400	99 745	0.4%
	Near bundled (25)	32 400	79.3	77.9	119	11 880	16 214	0.7%
	Part. distributed (50)	34 800	52.5	51.3	91	1 280	2 066	3.7%
	Distributed (75)	39 600	19.9	19.5	34	48	124	8%
	95	No solutions						

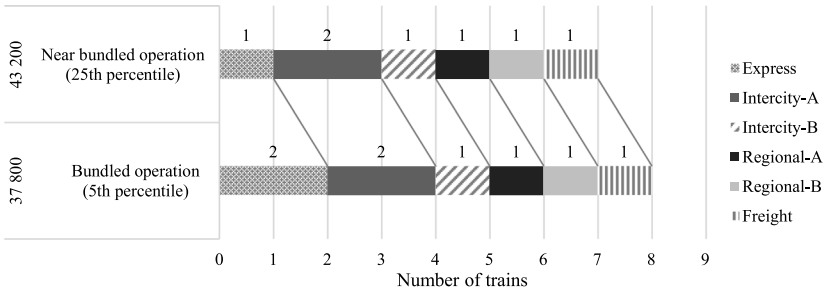
be utilised under perfect conditions, i.e. there is no buffer to absorb delay as described in Section 2. The 75% threshold given by the UIC is given for line sections which are divisions of the network into sections between larger stations and stations where overtaking and crossings take place. This 75% threshold is not suitable for routes (start to end of a train line) and networks which usually cover the equivalent of several line sections. As shown by Jensen et al. (2017), the capacity occupation on a route or in a network is higher compared to the line sections that compose the network or route. As we consider a (sub)network in our case study, we therefore use a threshold of 84%, which is found by Jensen (2015b) as suitable for the network used in the case study. The threshold of 84% is considered the most realistic threshold of the three (the practical capacity). Furthermore, the optimisation has been executed using the 5th, 25th, 50th, 75th and 95th percentile, p. Combined with the three thresholds, we have a total of 15 different $(p, C_{\max})$-instances as shown in Table 3.

First, we investigate the performance of the branch-and-bound algorithm. In Table 3, the computational results of the optimisation are shown for the three different thresholds for capacity occupation and the different percentiles. The operational type listed categorises the percentiles into different types of operation. In "bundled" operation, slow and fast trains are bundled which consumes less capacity than "distributed" operation where slow and fast trains alternates (i.e. are distributed throughout the time period). Furthermore, the table shows the size of the problem instances ranging from 4 to 78,400 possible solutions depending on values of the capacity constraint, $C_{\max}$, and the percentile used. The number of possible solutions is derived as a combination of the upper thresholds of trains that can be added of each train type from the results of the dichotomic search, i.e. $\prod_{i=1}^{n}(x_i^{\max} - x_i^{\min} + 1)$. Thus, when the capacity constraints ($C_{\max}$ and/or percentile) are relaxed, the number of possible solutions calculated increases significantly. Theoretical number of nodes is the maximum number of nodes that can be generated, i.e. if the complete enumeration tree is generated. For the instances considered, the theoretical number of nodes spans from 17 to 99,745 in the case study depending on the constraints. Nodes evaluated describes the ratio of nodes, in relation to the theoretical number of nodes, that may contain the optimal solution and thus result in branching during the branch-and-bound search. The remainder of the generated nodes are pruned by bound or infeasibility by the algorithm.

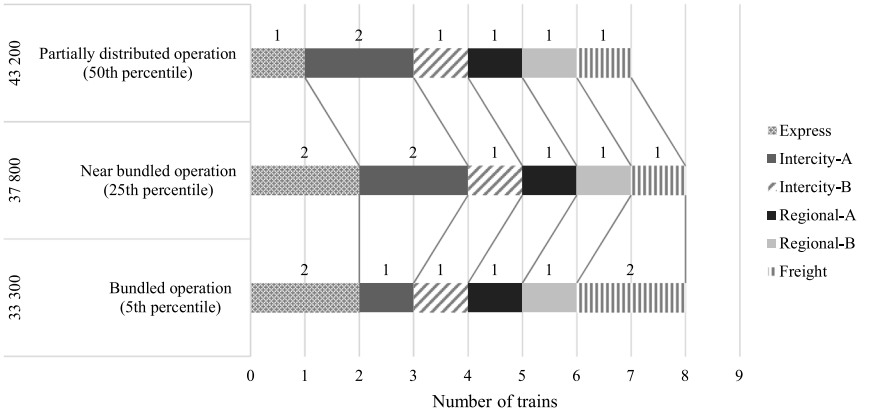
As Table 3 shows, the smallest instance is solved in less than 15 s, where the majority of the computation time is consumed by the dichotomic search described in Appendix. The largest instance is solved to optimality in less than 6 min, which is fast given the strategic planning horizon. During the execution of the branch-and-bound algorithm for the largest instance, only 0.4% of the tree nodes have to be evaluated as the remaining 99.6% can be pruned due to infeasibility or bound. Comparing these percentages with the percentages for the smaller instances it can be observed that the percentage of nodes evaluated decreases. This comes down to the fact that the more relaxed capacity constraint in the form of higher values of $C_{\max}$ and/or lower percentiles result in significantly more possible solutions as described earlier in this section. This results in a very large enumeration tree where the vast majority of solutions are found infeasible during the execution of the branch-and-bound algorithm. To avoid this, a better combined threshold on the number of trains must be produced, e.g. optimal solutions of all combinations of unique train type set of size two, three or even more. This is, however, computationally expensive due to many calls being necessary to the COE-module for evaluation of the capacity occupation rate $C(X, p)$.

The majority of computation time is spent in the COE-module with approx. 98% of the total computation time as listed in Table 3. The number of calls to the model range from 26 calls in the smallest instance to 403 calls in the largest instance.

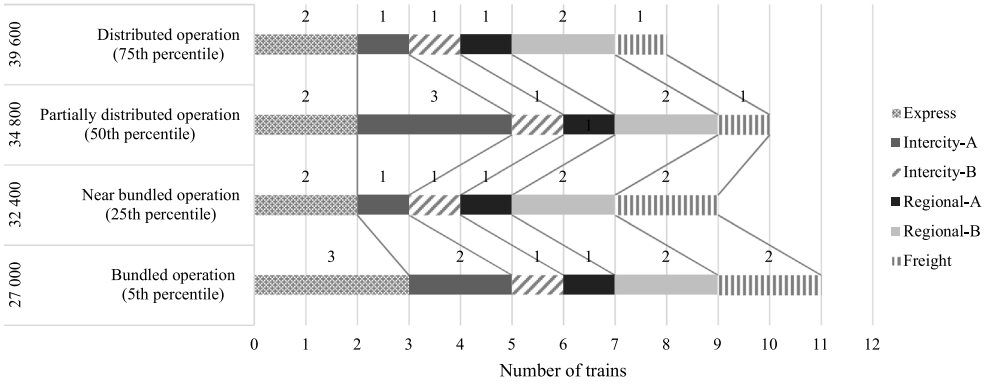
The capacity results are depicted in Fig. 5 for the three different thresholds for capacity occupation, $C_{\max}$, and the percentiles (operational types) that yield feasible solutions. Each row shows the number of trains of each train type that can be operated in the network in the defined time period. For (a) only 75% of the maximum capacity is utilised while (c) utilises all capacity. As a reference to these calculated percentiles and capacity thresholds, it can be noted that the 2013 timetable in Southwestern Denmark roughly corresponds to the 50th percentile with a capacity occupation rate threshold, $C_{\max}$, of 84%.



(a) $C_{max} = 75\%$ (UIC recommendation for line sections).



(b) $C_{max} = 84\%$ (practical maximum).



(c) $C_{max} = 100\%$ (maximum).

Fig. 5. Results for the case using branch-and-bound. Capacity figures as trains per train type per direction (specified with values immediately above the bars). Objective values for each solution to the left.

From Fig. 5(b) and (c), it can be derived that a capacity of three trains in total act as a buffer in the 2013 timetable when comparing the practical capacity (84%) with the maximum (theoretical) capacity of 100%, i.e. three potential trains are not scheduled to provide buffer time against delays. Lowering the threshold $C_{\max}$ to 75% yields a solution for the 25th percentile that is identical to the 50th percentile at 84%. From this it can be derived from that lowering the capacity threshold (to obtain better robustness against delays) requires the trains to be bundled in the timetable. Thus the 50th percentile solution at 84% is more attractive than the 25th percentile solution at 75% in terms of departure distribution even though the objective value is the same. Operational quality of a solution (capacity estimate) can therefore be perceived as the combination of the objective value, the percentile and the chosen capacity occupation rate threshold, $C_{\max}$. The objective value should be as low as possible to allow for as many departures as possible. The percentile should also be as high as possible to ensure that departures are distributed throughout

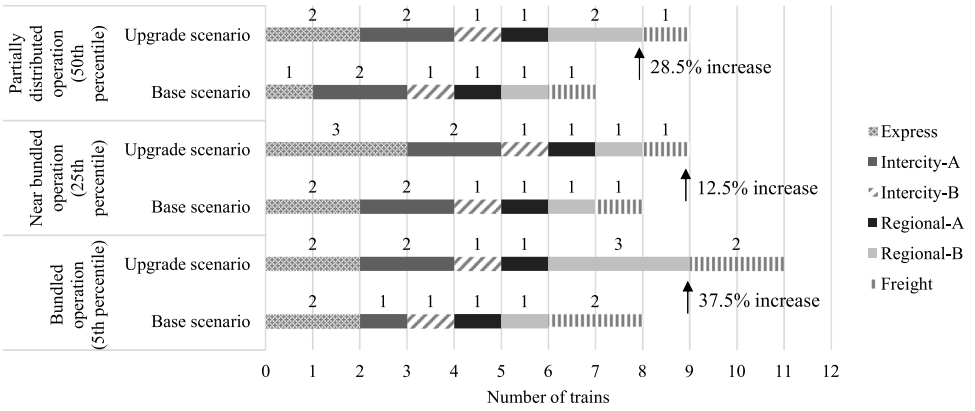


Fig. 6. Upgrade of Snoghøj and Taulov junctions (upgrade scenario). Capacity figures as trains per train type per direction (specified with values immediately above the bars). Capacity increase from base scenario (at-grade junctions) to upgrade scenario (grade-separated junctions) denoted by arrows.

Table 4
Weights used in the two sensitivity scenarios (equal weights and express priority), and the default scenario (freight priority).

Scenario	Express	Intercity-A	Intercity-B	Regional-A	Regional-B	Freight
Equal weights	1	1	1	1	1	1
Express priority	4	2	2	1	1	1
Freight priority (case default)	3	2	2	1	1	4

the time period. The capacity threshold should be high enough to maximise the number of trains, but also low enough to ensure buffer times for sufficient robustness against delays. The latter is to some extent subjective and based on preferences.

5.2. Capacity expansion

We now investigate the effect on infrastructure capacity by upgrading the infrastructure. The network as depicted in Fig. 4 has two at-grade junctions (Taulov and Snoghøj). This means that trains to/from Fredericia have to wait on trains running from Odense to Lunderskov or vice versa. These conflicts put a constraint on the capacity which can be solved by constructing an overpass or underpass (the junctions at FA are already built like this). Using the 84% capacity threshold, we use our proposed capacity determination framework to evaluate the capacity effect of upgrading the two at-grade junctions (Taulov and Snoghøj) to grade separated junctions to avoid the mentioned conflicts.

The results of this analysis are shown in Fig. 6. As labelled in the figure, the upgrade yields an increase in capacity of 1–3 trains or 12.5% to 37.5%. Comparing the 50th percentile, which resembles today's operation the most, an increase of two trains is obtained (or 28.5%) in the form of one additional express train and one additional regional-B train.

5.3. Sensitivity of train type weights

In this section, we present the results of a sensitivity analysis of the train type weights to investigate how the weights affect the capacity results. Table 4 shows the sensitivity scenarios investigated. We investigate two scenarios, one with equal weights and one with priority given to express trains. The first is chosen as it illustrates the effect of unweighted train types. The latter is chosen to significantly prioritise passenger express trains over freight trains. The freight priority weights are the default weights used throughout the case study.

Fig. 7 illustrates how the capacity results change based on weights for the 5th and 50th percentile. As the results show there is no difference for the 50th percentile under a capacity threshold of 84%. This comes down to the fact that the solution space is quite small due to the minimum threshold on trains. Thus, only one extra train in the form of an intercity-B can be added.

For the 5th percentile, the solution space is larger. Therefore, we can also observe how the different weights affect the capacity results. As observed in Fig. 7, giving priority to freight and express trains yields a lower total of trains than having equal weights. This is caused by the heterogeneity of the solution train mix. Express trains are the fastest and the freight (and regional-A) trains are the slowest. Increasing the number of trains of these types in a solution results in a more heterogeneous operation (and thus higher capacity occupation). Using equal weights results in a high amount of regional-B train being added as these consume a small amount of capacity (lowering solution heterogeneity). However, such a high amount of regional-B trains is not desirable in terms of demand. The freight priority weights (case default) therefore constitute a good basis for realistic capacity results.

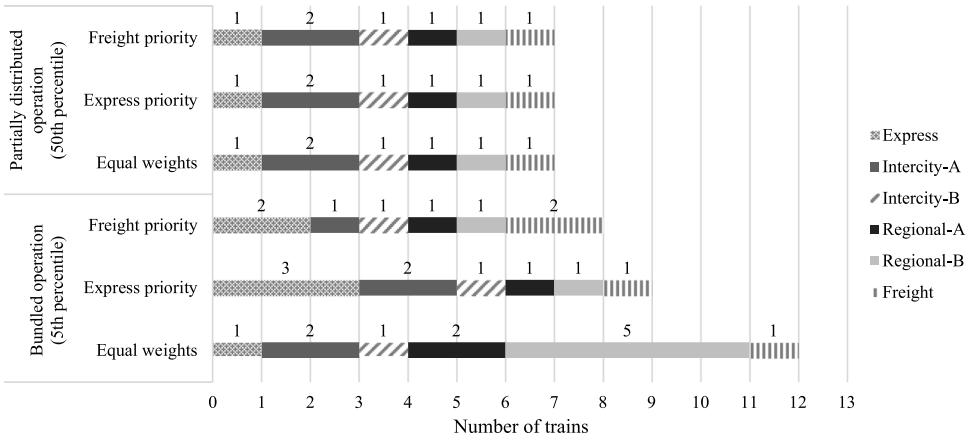


Fig. 7. Sensitivity of weights for the 5th and 50th percentile with a capacity threshold of 84%. Capacity figures as trains per train type per direction denoted with values immediately above the bars.

Table 5
Performance of tabu search algorithm. Upper bounds marked with *italic* are optimal. *Optimal solution found in dichotomic search.

C_{max}	Operational type (percentile)	Obj. value (optimal)	Tabu search		CPU time [s]		
			LB	UB	TS	B&B	B&B w/ TS
75%	Bundled (5)	37 800	21 000	*37 800*	15.3	25.5	32.7
	Near bundled (25)	43 200	40 800	*43 200*	2.3*	14.4	16.4
	Part. distributed (50)	No solutions					
	Distributed (75)	No solutions					
	95	No solutions					
84%	Bundled (5)	33 300	12 940	*33 300*	45.1	51.0	121.0
	Near bundled (25)	37 800	19 200	*37 800*	18.1	28.6	51.7
	Part. distributed (50)	43 200	41 400	*43 200*	2.6*	16.5	16.9
	Distributed (75)	No solutions					
	95	No solutions					
100%	Bundled (5)	27 000	8460	*27 000*	35.2	343.9	300.5
	Near bundled (25)	32 400	11 787	*32 400*	20.5	79.3	115.0
	Part. distributed (50)	34 800	17 190	*34 800*	31.8	52.5	81.9
	Distributed (75)	39 600	33 900	*39 600*	17.3	19.9	27.1
	95	No solutions					

5.4. Performance of tabu search algorithm

For the tabu search, we have used a restart interval of 25 iterations and a maximum tabu list size of 25 as these have provided fast convergence in our tests. In Table 5, the results are compared with the branch-and-bound algorithm. As the table shows, the tabu search is able to find an optimal solution in all cases. Except for the large instance (100% and 5th percentile) the tabu search is not faster than the branch-and-bound. The optimal solution is found faster (the CPU time defined), however, the tabu search needs to run longer than this to increase the probability that a good solution has been found. The result for the largest instance suggests that the tabu search will perform better in terms of computation time than the branch-and-bound in large problem instances. To further test this, we have scaled the case to simulate even larger problem instances by increasing the maximum capacity occupation rate, $C_{\max}$. These scaling results are shown in Fig. 8. The two main take-aways from the figure are: (1) in all but one instance, the tabu search finds the optimal solution, (2) the CPU (computation) time used by the branch-and-bound search increases exponentially (note the logarithmic scale) when the instance size increase. The latter is caused by decreasingly less tight lower bounds, as described in Table 3, causing the branch-and-bound tree to explode with increasing instance size.

In addition to providing measures for tighter lower bounds, the upper bounds may also be tightened to be able handle larger instances. One possibility is to tighten the initial upper bound for the branch-and-bound search. We have tested this by using the tabu search in sequence with the branch-and-bound algorithm. As listed in Table 5 (B&B w/ TS), this provides a reduction in computation time in the largest problem instance (5th percentile at 100%) when using branch-and-bound combined with tabu search instead of only branch-and-bound. In the smaller problem instances, the number of calls to the COE-module in the branch-and-bound search is not decreased sufficiently to offset the extra computation time used in the tabu search. Another possibility to tightened upper bounds during traversal of the branch-and-bound tree is by integrating the tabu search algorithm in the branch-and-bound algorithm as described in Puchinger and Raidl (2005). This could improve the ability of the branch-and-bound algorithm to handle larger instances faster as larger branches in the solution tree can be pruned away using the (potentially) better upper bound.

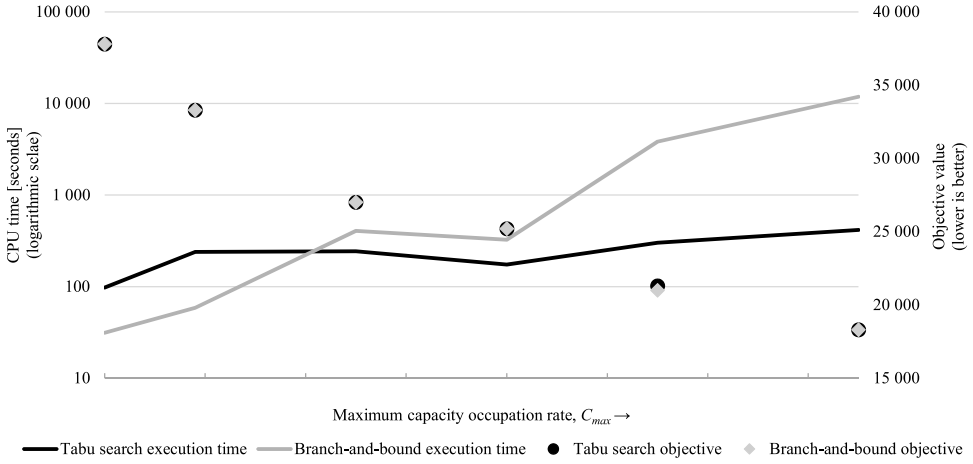


Fig. 8. Test of scalability of branch-and-bound and tabu search with increasing instance size. Maximum occupation rate (x-axis) increases from left to right.

Conclusively, the branch-and-bound search provides exact results and fast computation times in the small problem instances. The tabu search provides quality results and computation times that are much faster than the branch-and-bound search for the larger problem instances even if measures are taken to improve the branch-and-bound search.

6. Conclusions

In railway networks with heterogeneous operation the capacity depends on the sequence (order) of trains. As this is unknown in the early planning phases (strategic planning), the presented capacity determination framework provides a maximal set of trains (and a span of trains per train type) that the network can handle. The framework determines the maximal set of trains that can be scheduled within a threshold of capacity occupation while minimising weighted average waiting time. Differentiated weights make it possible to prioritise certain trains types according to demand.

The framework was tested using branch-and-bound and tabu search for solving the optimisation problem on a case network of 161 kilometres of double track lines in Southwestern Denmark. The capacity could be estimated in the order of seconds in small problem instances, and up to six minutes in the largest problem instance. The branch-and-bound and tabu search performed approximately equal in the analysed problem instances, except for the largest instances where the tabu search was faster. For early planning phases the model ran fast and should therefore be able to handle larger problem instances than used in this paper.

Some practical implications on the specific case was that upgrading the infrastructure resulted in an increase of capacity of 1–3 trains or 12.5% to 37.5%. Comparing with today's operation, an increase of two trains is obtained (or 28.5%). This result was obtained fast with a small amount of input data. Thus, with our framework it is possible to analyse a large amount of infrastructural changes, operational constraints or new types of rolling stock in a short amount of time. Consequently, this makes it possible to analyse a larger span of scenarios in early planning phases, thus reducing capital spending and optimising the utilisation of railway capacity.

CRediT authorship contribution statement

Lars Wittrup Jensen: Conceptualization, Methodology, Data curation, Software, Formal analysis, Investigation, Visualization, Writing - original draft. **Marie Schmidt:** Conceptualization, Methodology, Writing - review & editing. **Otto Anker Nielsen:** Conceptualization, Methodology, Writing - review & editing.

Acknowledgements

The research described in this paper is part of the research projects RobustRailS and IPTOP funded by the Innovation Fund Denmark. We would like to thank the council for making it possible to conduct this research. We are also very grateful to Professor Leo G. Kroon for his guidance and comments in the early work for this paper.

Appendix. Dichotomic search

To determine the maximum number of trains per train type that can be included in the capacity-representing set of trains, we use a dichotomic search algorithm which is an extension of the well-known binary search algorithm. In the following, we describe the algorithm followed by an example. The dichotomic search is described in pseudocode in Algorithm 3.

Algorithm 3: Pseudocode for dichotomic search.

Data: $i \in X$: train type for which as many trains as possible should be added;
C_{max}: threshold for capacity occupation, p: percentile;
Result: x_i^{max}: maximum number of trains that can be added
$x_i \leftarrow x_i^{min} \quad \forall i \in \{1, 2, ..., n\}$;
$x_i^{guess} \leftarrow$ guess on upper interval bound;
$x_i^{LB} \leftarrow x_i^{min}$, $x_i^{UB} \leftarrow x_i^{guess}$;
/* Part 1: Find an initial upper interval bound on how many trains of type i that can be added */
$loop \leftarrow true$;
while *loop* **do**
  $x_i \leftarrow x_i^{UB}$;
  /* If infeasible */
  if $C(X, p) > C_{max}$ **then**
    $loop \leftarrow false$; // Upper bound for interval found, terminate loop
  else
    /* If feasible */
    $x_i^{LB} \leftarrow x_i$;
    $x_i^{UB} \leftarrow x_i + x_i^{guess}$;
    if $x_i > x_i^{max}$ **then** $x_i^{max} \leftarrow x_i$; // Save as current best
/* Part 2: Use binary search to find the maximum number of trains that can be added of type i */
while $x_i^{LB} + 1 < x_i^{UB}$ **do**
  $x_i \leftarrow \lfloor (x_i^{LB} + x_i^{UB})/2 \rfloor$; // Set midpoint
  /* If infeasible */
  if $C(X, p) > C_{max}$ **then**
    $x_i^{UB} \leftarrow x_i$; // Continue search in lower search half
  else
    /* If feasible */
    $x_i^{LB} \leftarrow x_i$; // Continue search in upper search half
    if $x_i > x_i^{max}$ **then** $x_i^{max} \leftarrow x_i$; // Save as current best
return x_i^{max};

If the set of trains with a predefined minimum of trains per train type (minimum solution) is feasible, the dichotomic search is started. The dichotomic search is repeated for all train types. For our problem the search works by initially determining an upper bound on the search interval for the train type in question, x_i^{UB}. That is, an upper interval bound on the number of trains of a train type that yields an infeasible set of trains. Given this interval bound, we know that the best feasible solution is in the interval between the lower interval bound (current best feasible solution), x_i^{LB}, and the upper interval bound (an infeasible solution). To find the initial upper interval bound, a guess, x_i^{guess}, is required. A good guess will provide a good upper interval bound that is close to the best feasible solution, but still infeasible. A bad guess will result in extra iterations in the second part of the algorithm. Given that we focus on one train type at a time in the dichotomic search, a guess could be based on the minimum headway time for the train type divided by the time period. This will yield a set of trains with the maximum number of trains of the type in a total homogeneous operation. The search for the initial upper bound is described in pseudocode in the first part of Algorithm 3.

When the initial upper interval bound has been found, the search for the maximum number of trains is started (second part of Algorithm 3). For each iteration the number of trains of the train type to be tested for feasibility is the midpoint between the lower interval bound and upper interval bound. If the tested number of trains yields an infeasible solution, the upper interval bound is set to the tested number of trains. If a feasible solution is found the lower interval bound is set to the tested number of trains. The algorithm is stopped after convergence ($x_i^{LB} + 1 = x_i^{UB}$) and the best solution (maximum number of trains of a type), $x_i^{\max}$, is returned.

In Fig. A.9, an example is given to illustrate the steps in the dichotomic search. In the example, the minimum set of trains defined, $x_i^{\min} = 1$, is found to be feasible in iteration 1. The initial guess on the upper interval bound is set to $x_i^{guess} = 10$. This guess is found to be infeasible in iteration 2, and the first part of the search is complete as an upper interval bound has been found ($x_i^{UB} = 10$). In iteration 3–5, the second part of the search (the binary search) divides the search interval until the best feasible solution, $x_i^{\max} = 5$, is found.

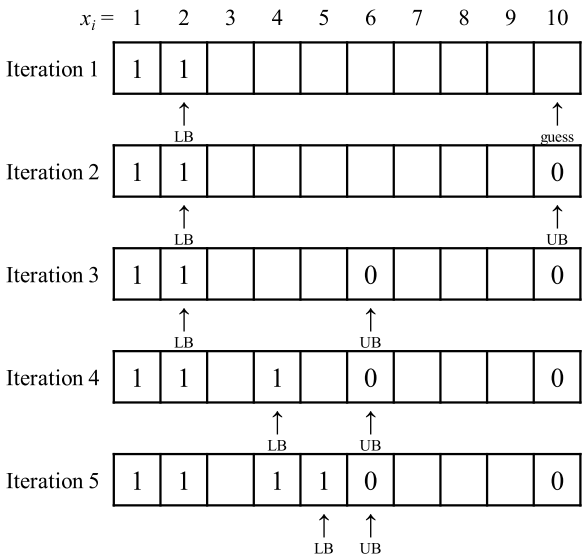


Fig. A.9. Example of iterations in the dichotomic search for a train type i. 0 denotes that the solution is infeasible while 1 denotes that the solution is feasible for a given value of x_i. Iteration 1–2 corresponds to the first part (while loop) and iteration 3–5 to the second part (while loop) in Algorithm 3. LB, UB, and guess are pointers for x_i^{LB}, x_i^{UB}, and x_i^{guess}.

References

Abril, M., Barber, F., Ingolotti, L., Salido, M.A., Tormos, P., Lova, A., 2008. An assessment of railway capacity. Transp. Res. E: Logist. Transp. Rev. 44 (5), 774–806. http://dx.doi.org/10.1016/j.tre.2007.04.001.

Bešinović, N., Goverde, R.M., 2018. Capacity assessment in railway networks. In: Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T. (Eds.), Handbook of Optimization in the Railway Industry. In: International Series in Operations Research & Management Science, vol. 268, Springer, http://dx.doi.org/10.1007/978-3-319-72153-8_2.

Bevrani, B., Burdett, R.L., Bhaskar, A., Yarlagadda, P.K., 2017a. A capacity assessment approach for multi-modal transportation systems. European J. Oper. Res. 263 (3), 864–878. http://dx.doi.org/10.1016/j.ejor.2017.05.007.

Bevrani, B., Burdett, R.L., Yarlagadda, P.K.D.V., 2017b. A case study of the Iranian national railway and its absolute capacity expansion using analytical models. Transport 32 (4), 398–414. http://dx.doi.org/10.3846/16484142.2015.1099053.

Burdett, R.L., 2015a. Incorporating complex train paths in an analysis of absolute capacity. Int. J. Railw. Technol. 4 (4), 73–83. http://dx.doi.org/10.4203/ijrt.4.4.4.

Burdett, R.L., 2015b. Multi-objective models and techniques for analysing the absolute capacity of railway networks. European J. Oper. Res. 245 (2), 489–505. http://dx.doi.org/10.1016/j.ejor.2015.03.020.

Burdett, R.L., 2016. Optimisation models for expanding a railway's theoretical capacity. European J. Oper. Res. 251 (3), 783–797. http://dx.doi.org/10.1016/j.ejor.2015.12.033.

Burdett, R.L., Kozan, E., 2006. Techniques for absolute capacity determination in railways. Transp. Res. B 40 (8), 616–632. http://dx.doi.org/10.1016/j.trb.2005.09.004.

Cacchiani, V., Toth, P., 2012. Nominal and robust train timetabling problems. European J. Oper. Res. 219 (3), 727–737. http://dx.doi.org/10.1016/j.ejor.2011.11.003.

De Kort, A.F., Heidergott, B., Ayhan, H., 2003. A probabilistic (max, +) approach for determining railway infrastructure capacity. European J. Oper. Res. 148 (3), 644–661. http://dx.doi.org/10.1016/S0377-2217(02)00467-8.

Glover, F., Laguna, M., 2013. Tabu search. In: Pardalos, P.M., Du, D.-Z., Graham, R.L. (Eds.), Handbook of Combinatorial Optimization. Springer New York, New York, NY, pp. 3261–3362. http://dx.doi.org/10.1007/978-1-4419-7997-1_17.

Harrod, S., 2009. Capacity factors of a mixed speed railway network. Transp. Res. E: Logist. Transp. Rev. 45 (5), 830–841. http://dx.doi.org/10.1016/j.tre.2009.03.004.

Ingvardson, J.B., Nielsen, O.A., Raveau, S., Nielsen, B.F., 2018. Passenger arrival and waiting time distributions dependent on train service frequency and station characteristics: A smart card data analysis. Transp. Res. C 90, 292–306. http://dx.doi.org/10.1016/j.trc.2018.03.006.

Jensen, L.W., 2015a. An optimisation framework for determination of capacity in railway networks. In: Conference on Advanced Systems in Public Transport. Rotterdam.

Jensen, L.W., 2015b. Robustness Indicators and Capacity Models for Railway Networks (Ph.D. thesis). Technical University of Denmark, p. 226.

Jensen, L.W., Landex, A., Nielsen, O.A., Kroon, L.G., Schmidt, M., 2017. Strategic assessment of capacity consumption in railway networks: Framework and model. Transp. Res. C 74, 126–149. http://dx.doi.org/10.1016/j.trc.2016.10.013.

Karoonsoontawong, A., Taptana, A., 2017. Branch-and-bound-based local search heuristics for train timetabling on single-track railway network. Netw. Spatial Econ. 17 (1), 1–39. http://dx.doi.org/10.1007/s11067-015-9316-4.

Mussone, L., Wolfler Calvo, R., 2013. An analytical approach to calculate the capacity of a railway system. European J. Oper. Res. 228 (1), 11–23. http://dx.doi.org/10.1016/j.ejor.2012.12.027.

Polinder, G.-J., Schmidt, M., Huisman, D., 2020. Timetabling for Strategic Passenger Railway Planning. ERIM Report Series Research in Management Erasmus Research Institute of Management, ERIM.

Puchinger, J., Raidl, R., 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. Artif. Intell. Knowl. Eng. Appl. Bioinspired Approach 3562, 41–53. http://dx.doi.org/10.1007/11499305_5.

Reinhardt, L.B., Pisinger, D., Lusby, R., 2017. Railway capacity and expansion analysis using time discretized paths. Flexible Serv. Manuf. J. 1–28. http://dx.doi.org/10.1007/s10696-017-9292-8.

UIC, 2004. Capacity (UIC Code 406). Technical Report, International Union of Railways (UIC), pp. 1–21.

UIC, 2013. UIC Code 406, Technical Report, second ed. International Union of Railways (UIC).

Yaghini, M., Nikoo, N., Ahadi, H.R., 2014. An integer programming model for analysing impacts of different train types on railway line capacity. Transport 29 (1), 28–35. http://dx.doi.org/10.3846/16484142.2014.894938.

Zhang, X., Nie, L., 2016. Integrating capacity analysis with high-speed railway timetabling: A minimum cycle time calculation model with flexible overtaking constraints and intelligent enumeration. Transp. Res. C 68, 509–531. http://dx.doi.org/10.1016/j.trc.2016.05.005.