Satisfiability and Optimization in Periodic Traffic Flow Problems

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)
von

Peter Großmann
geb. 28. Juni 1985 in Halle (Saale)

genehmigt von der
Fakultät Verkehrswissenschaften „Friedrich List“
TU Dresden

Eingereicht am 4. März 2016

Gutachter:
Prof. Dr. rer. nat. habil. Karl Nachtigall (Professur Verkehrsströmungslehre, TU Dresden)
Prof. Dr. Leo Kroon (Professur Quantitative Logistics, Erasmus University Rotterdam)

Tag der Verteidigung: 25. Oktober 2016

Abstract

Automatically calculating periodic timetables in public railway transport systems is an *NP*-complete problem – namely the Periodic Event Scheduling Problem (PESP). The original model is restricted to basic periodic timetabling. Extending the model by decisional transport networks with flows induces new possibilities in the timetabling and planning process. Subsequently, the given flexibility results in a generic model extension of PESP that can be applied in subsets of the timetabling process. The successful utilization of this approach is presented for distinct chain paths, duplicated chain paths and non-connected flow graphs that represent integration of routing and timetabling, planning of periodic rail freight train paths and track allocation, respectively. Furthermore, the encoding of this generic model into a binary propositional formula is introduced and the appropriate usage of several techniques like SAT solving and MaxSAT to calculate and optimize the corresponding instances will be presented accordingly. Computational results for real-world scenarios suggest the practical impact and give promising perspectives for further scientific research.

Contents

List of Figures

List of Tables

List of Abbreviations

CNF	conjunctive normal form
DPEN	decision periodic event network
DPESP	decision periodic event scheduling problem
FDPEN	flow decision periodic event network
FDPESP	flow decision periodic event scheduling problem
FG	flow graph
MaxSAT	boolean maximum satisfiability problem
MIP	mixed integer problem
PEN	periodic event network
PESP	periodic event scheduling problem
SAT	boolean satisfiability problem
WCNF	weighted conjunctive normal form

List of Symbols and Terms

$\mathcal{A}$	set of possible decision edges
c	clause, possibly indexed
χ_K	relation between set of edges and paths of flow graph K
$\mathcal{D}$	decision periodic event network
D_K	set of destination nodes of the flow graph K
δ	to be optimized slack variable of a constraint
$\mathcal{E}$	set of constraints (edges)
e	constraint or flow edge (explicitly given)
F, H	set of flow edges
$\mathcal{F}$	propositional formula in (weighted) conjunctive normal form (CNF)
f	propositional value for false (or 0)
γ	periodic events' weight function
$\mathcal{H}$	set of possible decision nodes
i,j	periodic events or integers (explicitly given)
$I(n)$	maps node n to set of incoming edges
J	interpretation (SAT, MaxSAT)
K	flow graph
L	literal, possibly indexed
$\mathcal{L}(\mathcal{R})$	set of propositional formulas (language)
l	label function between the set of flow edges and set of periodic events

n,m,o	periodic events or flow nodes (explicitly given)
$\mathbb{N}$	set of natural numbers (non-negative integers) including 0
$\mathbb{N}_+$	set of natural numbers (non-negative integers) without 0 ($\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$)
ω	constraints' weight function
$\Omega_{\mathcal{N}}$	SAT encoding of periodic events
$\Omega_{\mathcal{N}}^{max}$	MaxSAT encoding of periodic events
$O(n)$	maps node n to set of outgoing edges
p, q	propositional variable, possibly indexed
p^J	truth value of p under interpretation J
P_e	infeasible potential pairs of constraint e
Π	schedule
$\mathcal{P}(M)$	power set of M
$\Psi_{\mathcal{N}}$	SAT encoding of all constraints
R	set of flow nodes
$\mathcal{R}$	set of propositional variables
$\mathbb{R}$	set of real numbers
r_e	propositional decision variable of constraint e
ρ	path of a flow graph
$\mathcal{S}$	set of flow graphs
S_K	set of source nodes of the flow graph K
S_e	feasible potential pairs of constraint e
σ	slack of a constraint under a schedule
s_n	propositional decision variable of node n
$\mathcal{N}$	periodic event network
T	period

t	propositional value for true (or 1)
θ	maximum slack parameter
$\mathcal{V}$	set of periodic events (nodes)
W	sum of satisfied weighted clauses
$\bar{W}$	sum of minimal weights (weighted violations)
ξ	bijective function for extracting potentials by an interpretation
$\mathbb{Z}$	set of integers
ζ	maps to all rectangles of the infeasible region of a constraint
$\equiv$	binary operator for (semantical) equivalence
$\models$	models operator for propositional logic and periodic event scheduling
$\square$	empty clause
$\emptyset$	empty propositional formula (in conjunctive normal from), empty set

List of Encoding Mappings

enc	order encoding of finite ordered domains
enc_con	encoding of a PEN's constraint
enc_con_dec	encoding of a decisional constraint
enc_con_max	MaxSAT encoding for a hard constraint
$enc\_con\_max_\theta$	MaxSAT encoding of a to be optimized constraint with maximum slack parameter θ
enc_con_θ	encoding of a to be optimized constraint with maximum slack parameter θ
enc_dec	encoding of a decisional constraint with decision nodes
enc_dec_max	MaxSAT encoding of a to be optimized constraint
enc_dpen	encoding of a DPEN
enc_fdpen	encoding of an FDPEN
enc_fdpen_max	MaxSAT encoding of an to be optimized FDPEN
enc_fg	encoding of a flow graph
enc_fg_max	MaxSAT encoding of a flow graph
enc_max	MaxSAT order encoding of finite ordered domains
enc_pen	encoding of a PEN
enc_pen_max	MaxSAT encoding of a to be optimized PEN
enc_rec	excluding all infeasible pairs of a rectangle

1 Introduction

Modeling and automatic solving of large-scaled periodic event networks, like a timetable for the railway network of Germany, whose capacity network is depicted in Figure 1.1, is still an open scientific field. The model, the so called periodic event scheduling problem (PESP) [SU89], is well defined for periodic (cyclic) timetabling of public railway transport networks [Nac98]. However, since PESP is an NP-complete [Coo71] problem, there exists always pathological instances which cannot be solved in any reasonably given time frame unless $P = NP$. Besides that, recent research [Gro+12a; Nac98; Opi09] has shown that even large and complex networks can be handled efficiently in a reasonable amount of time automatically.

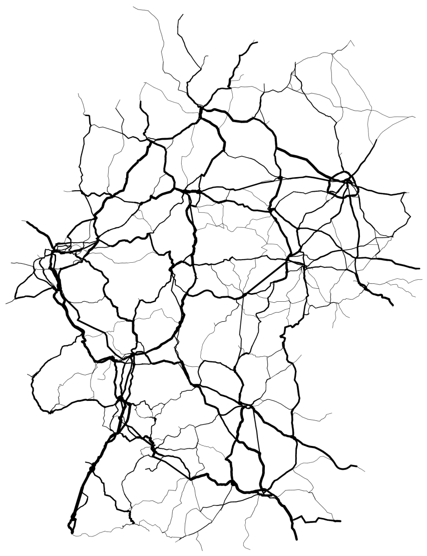


Figure 1.1: Real-world instance of Germany's long-distance train path system depicted as infrastructure network with corresponding capacity utilization (screenshot of software system TAKT).

Furthermore, the new techniques use efficient state-of-the-art SAT solvers by encoding PESP instances into a propositional formula and afterwards, let this formula be solved. This opens a whole new set of industrial scenarios that can be handled from now on for feasible timetables. Extending the model by decisional transport networks with flows – which is briefly visualized in Figure 1.2 – settles new possibilities in the timetabling and planning process in order to cover more industrial demands. Subsequently, the given flexibility results in a generic model extension of PESP that can be applied in subsets of the timetabling process. The successful utilization of this approach is presented for distinct chain paths, duplicated chain paths and non-connected flow graphs that represent integration of routing and timetabling, planning of periodic rail freight train paths and track allocation, respectively.

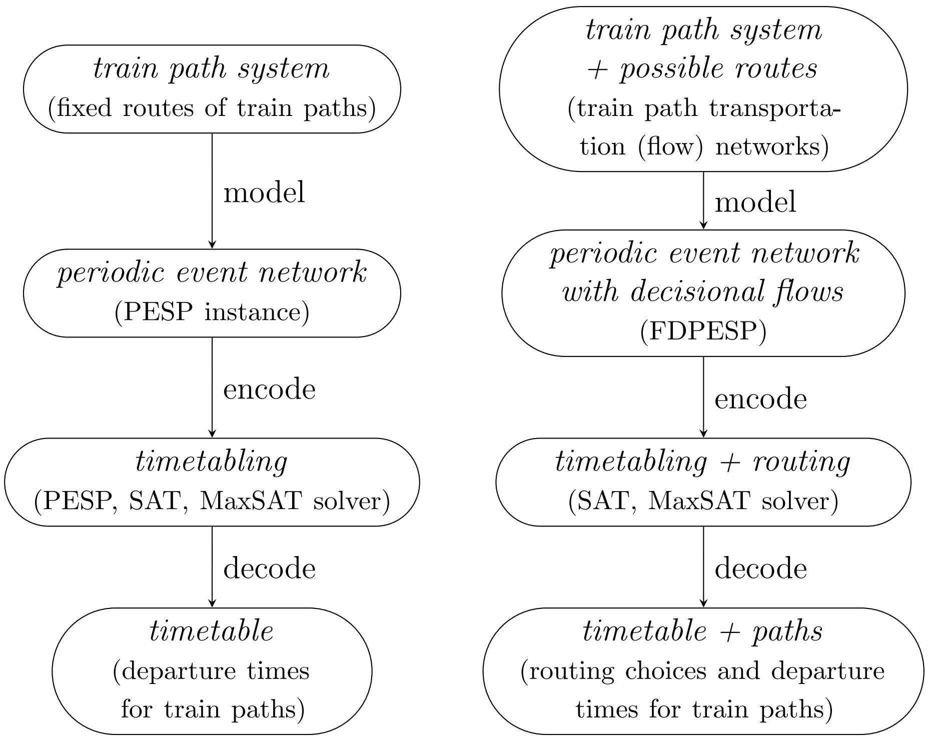

train path system
(fixed routes of train paths)
model
periodic event network
(PESP instance)
encode
timetabling
(PESP, SAT, MaxSAT solver)
decode
timetable
(departure times for train paths)
train path system + possible routes
(train path transportation (flow) networks)
model
periodic event network with decisional flows
(FDPESP)
encode
timetabling + routing
(SAT, MaxSAT solver)
decode
timetable + paths
(routing choices and departure times for train paths)

Figure 1.2: Extending periodic timetabling (left) with decisional flows by transportation networks (right).

The connection of PESP and flow graphs – alongside its complexity classifications – will be presented as well as its appropriate encodings. All of which will be validated by soundness and completeness proofs in order to ensure the correct usage of the encodings. Nevertheless, for computational evidence the new methods will be applied to real-world scenarios to ensure the possible use cases in industrial settings.

Additionally, the interest arises to optimize possible feasible timetables with respect to a compatible objective functional that suits the practical demand of railway operators.

Since the AI[1] developed powerful general purpose solvers to optimize propositional formulas[2] the work covers further encodings for the already named applications that transform the optimization instances into weighted propositional formulas. Likewise, successful computational results to real-world scenarios suggest a promising outlook for this approach.

All screenshots in this work are extracted of the software system TAKT which is developed by TU Dresden's Chair of Traffic Flow Science [Küm+13; Gro+12b; Gro+13; WON12; Küm+15] around the group of Nachtigall, Opitz, Weiß, Kümmling et al. in close collaboration with the most important German railway infrastructure provider DB Netz AG[3] [FP14; PF15; PW14; WL08] around the group of Weigand, Feil, Pöhle et al.

The outline of this work will be as follows: Firstly, Section 2 introduces the needed base problems SAT, PESP and MaxSAT as well as the base encoding that reduces PESP to SAT. Secondly, Using the already introduced SAT encoding [Gro+12a] and introducing the new encoding for flow graphs and its appropriate connection is presented in Section 3, as well as its possible applications covering both soundness and completeness theorems as well as complexity classifications. It follows in Section 4 the objective functionals for the introduced applications as MIP models as well as appropriate MaxSAT encodings. Yet again, soundness and completeness theorems show the correct possibility to use the encoded instances in state-of-the-art solvers. The experimental results in Section 5 will focus on instances for the industrial application of automated time table calculation of public railway transport networks with and without decisional flows. In the end, the work will be concluded in Section 6 with a scientific outlook for further research.

[1]artificial intelligence

[2]namely MaxSAT solvers

[3]DB Netz AG kindly provided the real-world scenario test data as input data for the software system TAKT that result in the work's screenshots and the computational results in Section 5

2 Preliminaries

This section covers the most important definitions and represents the foundation of the whole work. The two main areas propositional logic and periodic event scheduling will be introduced elementarily in Section 2.1 and Section 2.2, respectively. Furthermore, both topics are connected by the polynomial reduction from the Periodic Event Scheduling Problem (PESP) to the satisfiability problem (SAT) in Section 2.3. The latter is the basis of the subsequent encodings presented in this work. Hence, the given encoding is fundamental for the consequent understanding of the further encodings and the main point of interest in this section. The whole process is exemplified in Figure 2.1, whereas the most time consuming parts heavily depend on the given scenario.[1]

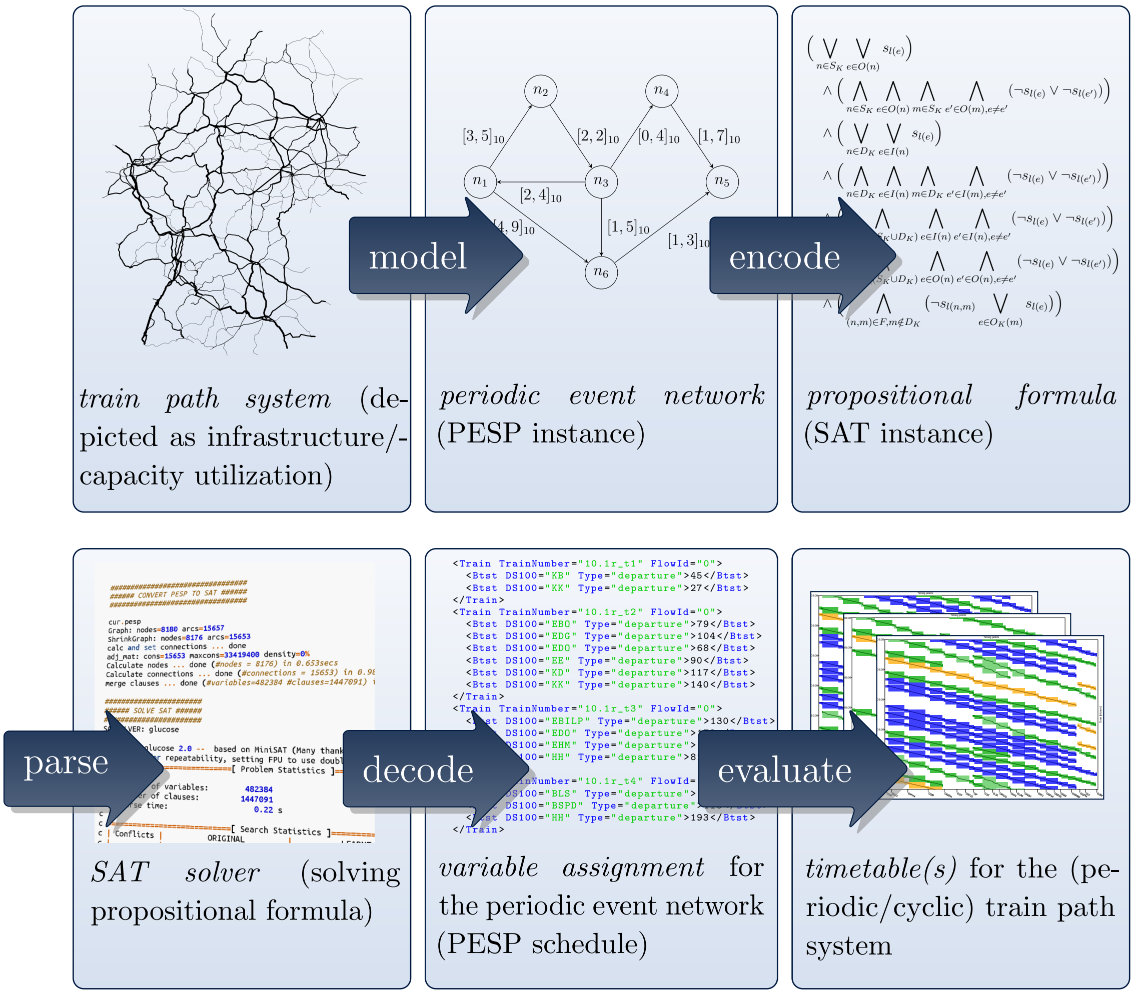
Since most parts of this section are well covered in the literature [Bie+09; SU89; Nac98; Gro11; Opi09], most lemmas and theorems will be left without proofs or just proof sketches.

2.1 Propositional Logic

Propositional logic is a fundamental topic in computer science[2] and other areas. More and more industrial problems are encoded into propositional logic [CDE08; Cla+01; MZ06]. Hence, the need of fast solvers, which can solve these problems in short time, increases. In this work, propositional logic will be introduced as binary logic, whose values are in $\{t, f\}$. An in depth overview of this topic can be found in the literature [Bie+09; DW83].

The syntax and semantics are introduced separately in Section 2.1.1 and Section 2.1.2, respectively. In the end, the satisfiability problem alongside its complexity classification will be shown followed by a short introduction of the maximum satisfiability problem in Section 2.1.4.

[1]Because the SAT solving step is NP-complete whereas the remaining parts have polynomial complexity.
[2]especially in artificial intelligence


model
encode
parse
decode
evaluate
n_1 n_2 n_3 n_4 n_5 n_6
$[3,5]_{10}$ $[2,2]_{10}$ $[0,4]_{10}$ $[1,7]_{10}$ $[2,4]_{10}$ $[4,9]_{10}$ $[1,5]_{10}$ $[1,3]_{10}$
train path system (depicted as infrastructure/-capacity utilization)
periodic event network (PESP instance)
propositional formula (SAT instance)
SAT solver (solving propositional formula)
variable assignment for the periodic event network (PESP schedule)
timetable(s) for the (periodic/cyclic) train path system

```
##################################
###### CONVERT PESP TO SAT ######
##################################

cur.pesp
Graph: nodes=8180 arcs=15657
ShrinkGraph: nodes=8176 arcs=15653
calc and set connections ... done
adj_mat: cons=15653 maxcons=33419400 density=0%
Calculate nodes ... done (#nodes = 8176) in 0.653secs
Calculate connections ... done (#connections = 15653) in 0.98
merge clauses ... done (#variables=482384 #clauses=1447091) i

#######################
###### SOLVE SAT ######
#######################
S    VER: glucose

      glucose 2.0 --  based on MiniSAT (Many thank
        r repeatability, setting FPU to use doubl
                    ===============[ Problem Statistics ]==

     of variables:          482384
     er of clauses:         1447091
     rse time:                 0.22 s
c
c ========================[ Search Statistics ]=========
c | Conflicts |            ORIGINAL            |
```

```
<Train TrainNumber="10.1r_t1" FlowId="0">
  <Btst DS100="KB" Type="departure">45</Btst>
  <Btst DS100="KK" Type="departure">27</Btst>
</Train>
<Train TrainNumber="10.1r_t2" FlowId="0">
  <Btst DS100="EBO" Type="departure">79</Btst>
  <Btst DS100="EDG" Type="departure">104</Btst>
  <Btst DS100="EDO" Type="departure">68</Btst>
  <Btst DS100="EE" Type="departure">90</Btst>
  <Btst DS100="KD" Type="departure">117</Btst>
  <Btst DS100="KK" Type="departure">140</Btst>
</Train>
<Train TrainNumber="10.1r_t3" FlowId="0">
  <Btst DS100="EBILP" Type="departure">130</Btst>
      DS100="EDO" Type="departure">
      DS100="EHM" Type="departure">
      100="HH" Type="departure">8
        inNumber="10.1r_t4" FlowId=
      DS100="BLS" Type="departure">
      DS100="BSPD" Type="departure">
  <Btst DS100="HH" Type="departure">193</Btst>
</Train>
```

Figure 2.1: Exemplifying process of timetabling in periodic (cyclic) railway timetabling.

2.1.1 Syntax

Definition 2.1 (Propositional Variable)**.** *Let* $p \in \mathcal{R}$ *be a* propositional variable *with* $\mathcal{R}$ *being the set of propositional variables.*

Definition 2.2 (Literal)**.** *A* literal L *is a propositional variable* p *or its negation* $\neg p$.

Definition 2.3 (Complement)**.** *Let* L *be a literal and* p *a propositional variable. Then* $\bar{L}$ *is the* complement *of* L *such that*

$$\bar{L} = \neg p :\Leftrightarrow L = p,$$
$$\bar{L} = p :\Leftrightarrow L = \neg p.$$

Definition 2.4 (Clause)**.** *A* clause $c = (L_1 \vee \ldots \vee L_n)$ *(*$n \in \mathbb{N}$*) is a disjunction of literals* L_i *(*$i \in \{1, \ldots, n\}$*).*

The clause $c := \square$ is called *empty*, if $n = 0$.

Definition 2.5 (Conjunctive Normal Form)**.** $\mathcal{F} = c_1 \wedge \ldots \wedge c_n \in \mathcal{L}(\mathcal{R})$ *(*$n \in \mathbb{N}$*) is a* propositional formula in conjunctive normal form (CNF) *if it is a conjunction of clauses* c_i*,* $i \in \{1, \ldots, n\}$ *with* $\mathcal{L}(\mathcal{R})$ *being the set of all propositional formulas.*

The propositional formula $\mathcal{F} := \emptyset$ is called empty, if $n = 0$. It follows firstly, that $\mathcal{F} = \square$ is the propositional formula in CNF with a single clause, the empty clause. Secondly, that $\mathcal{F} = \emptyset$ is a propositional formula in CNF without any clause.

Lemma 2.6. *Let c be a clause. Then $\mathcal{F} = c$ is a propositional formula in CNF.*

Proof. With $n = 1$ in Definition 2.5, $\mathcal{F} = c_1 = c$ is a propositional formula in CNF. $\square$

Lemma 2.7. *Let L be a literal. Then $\mathcal{F} = L$ is a propositional formula in CNF.*

Proof. Follows directly by Lemma 2.6 with $c = L_1 = L$. $\square$

Lemma 2.8. *Let $p \in \mathcal{R}$ be a propositional variable. Then $\mathcal{F} = p$ is a propositional formula in CNF.*

Proof. Follows directly by Lemma 2.7 with $L = p$. $\square$

Example 2.9 (Propositional Formulas in CNF)**.** Let $p, q, r \in \mathcal{R}$ be propositional variables. Then, with the previous definitions and lemmas it follows that

$$\begin{aligned}
\mathcal{F}_1 &= (p \vee \neg r) \wedge (\neg q \vee r),\\
\mathcal{F}_2 &= \neg p \wedge (q \vee r),\\
\mathcal{F}_3 &= (p \vee q) \wedge \square
\end{aligned}$$

are propositional formulas in CNF.

In the sequel, let all propositional formulas in CNF and clauses may be treated as finite sets. Hence, the binary set operators in $\{\cup, \cap, \setminus, \subseteq, \supseteq\}$ may be applied to them, accordingly. Thus, the empty set of both the formula and a clause will be explicitly handled in Section 2.1.2. Furthermore, in the following sections let p and q be propositional variables, L be a literal, c be a clause and $\mathcal{F}$ be a propositional formula in CNF.

2.1.2 Semantic

The mapping $J : \mathcal{L}(\mathcal{R}) \to \{t, f\}$ is called *interpretation.* It is sufficient to assign to each propositional variable $p \in \mathcal{R}$ a truth value such that $p^J \in \{t, f\}$ in order to evaluate the truth value of a propositional formula $\mathcal{F}$ in CNF [Bie+09], since all evaluations of the connectives — namely conjunction, disjunction and negation — under J are known beforehand. This will be covered by the following definitions. In the sequel, let J be an interpretation.

Definition 2.10. *A literal L is satisfied under J, denoted as $J \models L$, if and only if L is a propositional variable that is mapped to t under J, or L is a negated propositional variable that is mapped to f under J.*

Definition 2.11. *A clause c is satisfied under J, denoted as $J \models c$, if and only if it exists a literal in c that is satisfied under J.*

Definition 2.12. *A propositional formula $\mathcal{F} \in \mathcal{L}(\mathcal{R})$ in CNF is satisfied under J, denoted as $J \models \mathcal{F}$, if and only if all clauses in $\mathcal{F}$ are satisfied under J.*

If $\mathcal{F}$ is satisfied under J, then J is called *model* for $\mathcal{F}$. In case no such interpretation J for $\mathcal{F}$ exists, $\mathcal{F}$ is called *unsatisfiable.*

Example 2.13. Let be the propositional formulas

$$\begin{aligned} \mathcal{F}_1 &= (p \vee \neg r) \wedge (\neg q \vee r), \\ \mathcal{F}_2 &= \neg p \wedge (q \vee r), \\ \mathcal{F}_3 &= (p \vee q) \wedge \Box \end{aligned}$$

as in Example 2.9 and J be an interpretation such that

$$p^J = t, q^J = f, r^J = f.$$

Then, we can imply that $\mathcal{F}_1^J = t$ (or $J \models \mathcal{F}_1$), because with Definition 2.10 it follows that $(\neg q)^J = t$ and thus, with Definition 2.11 that $(p \vee \neg r)^J = t$ and $(\neg q \vee r)^J = t$. Finally, with Definition 2.12 it follows that $\mathcal{F}_1^J = t$.
$J \not\models \mathcal{F}_2$, because $(\neg p)^J = f$ and thus with Definition 2.12 $\mathcal{F}_2$ is not satisfied under J.
$J \not\models \mathcal{F}_3$, because with Definition 2.11 it follows that no literal under J is satisfied for $\Box$ and thus, $\mathcal{F}_3$ is not satisfied under J.

Definition 2.14 (Semtically Equivalent)**.** *Let $\mathcal{F}, \mathcal{G} \in \mathcal{L}(\mathcal{R})$ be propositional formulas. Then $\mathcal{F}$ and $\mathcal{G}$ are called* semantically equivalent *(or $\mathcal{F} \equiv \mathcal{G}$) if and only if for all interpretations J it holds*

$$J \models \mathcal{F} \Leftrightarrow J \models \mathcal{G}.$$

Semantical equivalence can be shown in the following famous [Bie+09] examples.

Example 2.15 (without proofs)**.** Let $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ be propositional formulas. Then

$$\begin{aligned}
\neg\neg\mathcal{F} &\equiv \mathcal{F} & \\
\neg(\mathcal{F} \wedge \mathcal{G}) &\equiv (\neg\mathcal{F} \vee \neg\mathcal{G}) & \text{(de Morgan)}\\
\neg(\mathcal{F} \vee \mathcal{G}) &\equiv (\neg\mathcal{F} \wedge \neg\mathcal{G}) & \text{(de Morgan)}\\
((\mathcal{F} \vee \mathcal{G}) \vee \mathcal{H}) &\equiv (\mathcal{F} \vee (\mathcal{G} \vee \mathcal{H})) & \text{(associativity)}\\
((\mathcal{F} \wedge \mathcal{G}) \wedge \mathcal{H}) &\equiv (\mathcal{F} \wedge (\mathcal{G} \wedge \mathcal{H})) & \text{(associativity)}\\
(\mathcal{F} \vee \mathcal{G}) &\equiv (\mathcal{G} \vee \mathcal{F}) & \text{(commutativity)}\\
(\mathcal{F} \wedge \mathcal{G}) &\equiv (\mathcal{G} \wedge \mathcal{F}) & \text{(commutativity)}\\
(\mathcal{F} \Rightarrow \mathcal{G}) &\equiv (\neg\mathcal{G} \vee \mathcal{F}) & \text{(implication).}
\end{aligned}$$

We can get the truth value of a propositional formula $\mathcal{F}$ and an interpretation J by

$$\mathcal{F}^J = \begin{cases} t & \text{if } J \models \mathcal{F} \\ f & \text{if } J \not\models \mathcal{F} \end{cases} \tag{2.1}$$

Hence, we can conduct the following corollary as described in the literature [Bie+09].

Corollary 2.16 (Semantically Equivalent)**.** *Let $\mathcal{F}, \mathcal{G} \in \mathcal{L}(\mathcal{R})$ be propositional formulas. Then with $\mathcal{F} \equiv \mathcal{G}$ it holds for all interpretations J that*

$$\mathcal{F}^J = \mathcal{G}^J.$$

Proof. 1. Let $\mathcal{F}^J = t$. Then, it follows

$$\begin{aligned}
\mathcal{F}^J = t &\overset{(2.1)}{\Leftrightarrow} J \models \mathcal{F}\\
&\overset{\text{Def 2.14}}{\Leftrightarrow} J \models \mathcal{G}\\
&\overset{(2.1)}{\Leftrightarrow} \mathcal{G}^J = t
\end{aligned}$$

2. Let $\mathcal{F}^J = f$. Then, it follows

$$\begin{aligned}\mathcal{F}^J = f &\overset{(2.1)}{\Leftrightarrow} J \not\models \mathcal{F}\\ &\overset{\text{Def 2.14}}{\Leftrightarrow} J \not\models \mathcal{G}\\ &\overset{(2.1)}{\Leftrightarrow} \mathcal{G}^J = f\end{aligned}$$

□

If we handle the special cases of an formula in CNF – namely $\mathcal{F}$ is empty or a clause c in $\mathcal{F}$ is empty – we get the following lemma.

Lemma 2.17. *Let $\mathcal{F} \in \mathcal{L}(\mathcal{R})$ in CNF. It holds for any interpretation J*

(i) $\mathcal{F} = \emptyset \Rightarrow J \models \mathcal{F}$

(ii) $\Box \in \mathcal{F} \Rightarrow J \not\models \mathcal{F}$

Proof. (i): follows by Definition 2.12, because it holds for all clauses in $\mathcal{F}$.
(ii): follows by Definition 2.11, because no literal in $\Box$ exists that becomes t under J. □

Example 2.18. Let $\mathcal{F}_3 = (p \vee q) \wedge \Box$ be as in Example 2.9. In Example 2.13 it is shown that $J \not\models \mathcal{F}_3$. With Lemma 2.17 we know that $\mathcal{F}_3$ is unsatisfied for any given interpretation and thus, it is unsatisfiable.

2.1.3 Satisfiability Problem (SAT)

There exist several classifications of the SAT problem [Bie+09]. This work will concentrate on SAT being a decision problem, which either says, that the given formula is satisfiable under a certain interpretation or that the formula is unsatisfiable. This decision can be computationally calculated by so called SAT solvers.

Definition 2.19 (SAT Problem)**.** *Let $\mathcal{F} \in \mathcal{L}(\mathcal{R})$ be a propositional formula in CNF. Then the* satisfiability problem (SAT) *is the decision problem whether an interpretation J exists with $J \models \mathcal{F}$.*

If such an interpretation exists, it is denoted as $\mathcal{F} \in \text{SAT}$.

Example 2.20. Let $\mathcal{F}_1, \mathcal{F}_3$ be the propositional formulas in CNF and J be the interpretation of Example 2.13. Then, with Example 2.18 we know that

$$\begin{aligned}\mathcal{F}_1 &\in \text{SAT},\\ \mathcal{F}_3 &\notin \text{SAT}.\end{aligned}$$

Since it is well known that the SAT problem is NP-complete [Coo71], it is not tractable to solve this decision problem. Unless it does exist an algorithm with polynomial complexity that solves an NP-complete problem, it will stay hard to solve such a SAT instance. This is a common open problem in computer science, that is known as $P \overset{?}{=} NP$.

Nearly all state-of-the-art SAT solvers [Man15; ALS13; Bie13] are not based on simple backtracking with respect to truth value assignments of the propositional variables. The new technique is called conflict driven back jumping and learning [MS96; MS99; Bie+09; SE05]. Once the solver detects a conflict it does not simply backtrack to the last decision point, but analyzes the current conflict with respect to its clauses by searching in a so called implication graph. This allows to learn a new clause (or lemma) of the current conflict and jump several decision levels back. This offers two huge improvements: On the one hand, a lot of search space is skipped, because several decision levels have been cut. On the other hand, the solver will never run into the same conflict again, since it learned the problem (clause) of the conflict it was driven beforehand.

Most state-of-the-art SAT solvers require the propositional formula to be in CNF. Hence, encoding a domain into SAT should consider this fact, since transforming afterwards a formula into CNF, for example with the rules given in Example 2.15, often ends in an exponential explosion with respect to the number of clauses.

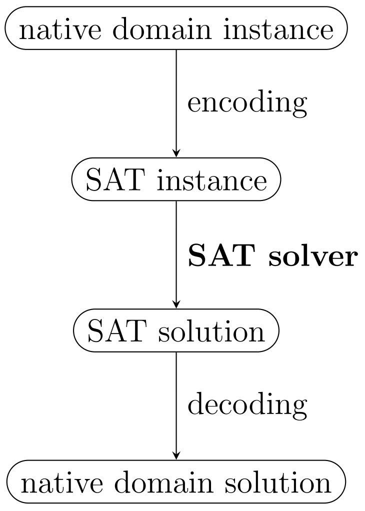


Figure 2.2: SAT solver as general purpose solver.

Several industrial problems have already been reformulated as propositional formulas and been solved by SAT solvers [CDE08; Cla+01; MZ06]. Since all those domains showed promising results, this work tries this approach as well: encoding the native domain into a propositional formula in CNF and then solve the instance by a state-of-the-art SAT solver. It can be conducted that SAT solvers can be used as general purpose solvers as visualized in Figure 2.2 with the SAT solution being either unsatisfiable or a model that satisfies the formula.

Since SAT solvers are already used in industrial domains, the SAT community came up with other powerful solvers for other SAT-based domains. One of these domains is the MaxSAT problem that maximizes the number of clauses that are satisfied under a to be determined interpretation. This technique will be introduced in the following section and further used in Section 4.

2.1.4 Maximum Satisfiability

Applying weights for clauses of a propositional formula in CNF as in Definition 2.5 results in a weighted formula [FM06; Bie+09].

Definition 2.21 (Weighted Propositional Formula)**.** *Let* $\omega_i \in \mathbb{N} \cup \{\infty\}$ *be weights for clauses* c_i *(*$i \in \{1, \ldots, n\}$*). Then,*

$$\mathcal{F} = (\omega_1, c_1) \land \ldots \land (\omega_n, c_n)$$

is called weighted propositional formula in CNF (WCNF)*.*

Each clause c with weight $\omega = \infty$ is said to be a *hard clause* [Bie+09] because it must be satisfied under any interpretation J. In the remaining work, we assume

$$\infty \cdot 0 := 0$$

Example 2.22 (WCNF)**.** Let $\mathcal{F}_1 \land \mathcal{F}_2 = (p \lor \neg r) \land (\neg q \lor r) \land \neg p \land (q \lor r)$ be a propositional formula in CNF as in Example 2.9. Then,

$$\mathcal{F}_4 = (4, (p \lor \neg r)) \land (3, (\neg q \lor r)) \land (\infty, \neg p) \land (2, (q \lor r))$$
$$\mathcal{F}_5 = (4, (p \lor \neg r)) \land (3, (\neg q \lor r)) \land (1, \neg p) \land (2, (q \lor r))$$

are propositional formulas in WCNF that only differ in the weight for the clause $(\neg p)$.

Definition 2.23 (MaxSAT Problem)**.** *Let* $\mathcal{F}$ *be a propositional formula in WCNF. Then, the* (partial weighted) MaxSAT problem *requires an interpretation* J *such that the sum of weights of clauses* c *with* $J \not\models c$ *is minimal and less than* ∞*, which is denoted as* $J \models \mathcal{F}$*.*

If the sum equals ∞ a hard clause is not satisfied and thus, no such interpretation exists, respectively for any interpretation the sum of weights that are violated equals always ∞.

Lemma 2.24. *Let $\mathcal{F} = (\infty, c_1) \wedge \ldots \wedge (\infty, c_n)$ be a propositional formula in WCNF and J an interpretation. Then it holds for a propositional formula $\mathcal{G} = c_1 \wedge \ldots \wedge c_n$ in CNF*

$$J \models \mathcal{F} \Leftrightarrow J \models \mathcal{G}.$$

Proof.

$$J \models \mathcal{F} \overset{\text{Def. 2.23}}{\Leftrightarrow} \forall c \in \mathcal{F} : J \models c \overset{\text{Def. 2.12}}{\Leftrightarrow} J \models \mathcal{G}.$$

□

This lemma shows that a propositional formula in WCNF whose weights equals ∞ can be transformed into an equivalent SAT problem.

The sum of minimal weights (i. e., violations) for a propositional formula in WCNF $\mathcal{F} = (\omega_1, c_1) \wedge \ldots \wedge (\omega_n, c_n)$ and an interpretation J can be calculated with $\bar{W}$ such that

$$\bar{W}(\mathcal{F}, J) = \sum_{i=1}^{n} \omega_i (1 - x_i)$$

with

$$x_i := \begin{cases} 1, & J \models c_i \\ 0, & J \not\models c_i. \end{cases}$$

Hence, we can equivalently formulate Definition 2.23 for a propositional formula $\mathcal{F}$ in WCNF

$$\bar{W}(\mathcal{F}, J) \rightarrow \min.$$

Equivalently, the sum of satisfied weighted clauses W can be calculated with

$$W(\mathcal{F}, J) = \sum_{i=1,\, \omega_i \neq \infty}^{n} \omega_i x_i. \tag{2.2}$$

Evaluating W for $\mathcal{F}$ and J is only useful in case of all hard clauses being satisfied under J such that

$$\forall i \in \{1, \ldots, n\}, \omega_i = \infty : J \models c_i.$$

Example 2.25 (Solution WCNF)**.** Let $\mathcal{F}_4$, $\mathcal{F}_5$ be propositional formulas in WCNF as in Example 2.22 and J_4, J_5 be interpretations such that

$$p^{J_4} = f, q^{J_4} = f, r^{J_4} = f$$
$$p^{J_5} = t, q^{J_5} = f, r^{J_5} = t.$$

Then, $J_4 \models \mathcal{F}_4$ and $J_5 \models \mathcal{F}_5$ and we can evaluate $\bar{W}$ and W with

$$\begin{aligned}
\bar{W}(\mathcal{F}_4, J_4) &= 4(1 - x_1) + 3(1 - x_2) + \infty(1 - x_3) + 2(1 - x_4) \\
&= 4 \cdot 0 + 3 \cdot 0 + \infty \cdot 0 + 2 \cdot 1 = 2, \\
\bar{W}(\mathcal{F}_5, J_5) &= 4(1 - x_1) + 3(1 - x_2) + 1(1 - x_3) + 2(1 - x_4) \\
&= 4 \cdot 0 + 3 \cdot 0 + 1 \cdot 1 + 2 \cdot 0 = 1, \\
W(\mathcal{F}_4, J_4) &= 4x_1 + 3x_2 + 2x_4 = 4 + 3 = 7, \\
W(\mathcal{F}_5, J_5) &= 4x_1 + 3x_2 + 1x_3 + 2x_4 = 4 + 3 + 2 = 9.
\end{aligned}$$

No other interpretation as a lower sum of violated weights exists. For example, $J_5 \not\models \mathcal{F}_4$ and $J_4 \not\models \mathcal{F}_5$ because

$$\begin{aligned}
\bar{W}(\mathcal{F}_4, J_5) &= 4(1 - x_1) + 3(1 - x_2) + \infty(1 - x_3) + 2(1 - x_4) \\
&= 4 \cdot 0 + 3 \cdot 0 + \infty \cdot 1 + 2 \cdot 0 = \infty > 2 = \bar{W}(\mathcal{F}_4, J_4), \\
\bar{W}(\mathcal{F}_5, J_4) &= 4(1 - x_1) + 3(1 - x_2) + 1(1 - x_3) + 2(1 - x_4) \\
&= 4 \cdot 0 + 3 \cdot 0 + 1 \cdot 0 + 2 \cdot 1 = 2 > 1 = \bar{W}(\mathcal{F}_5, J_5).
\end{aligned}$$

Furthermore, we can evaluate $\mathcal{F}_5$ for J_4:

$$W(\mathcal{F}_5, J_4) = 4x_1 + 3x_2 + 1x_3 + 2x_4 = 4 + 3 + 1 = 8 < 9 = W(\mathcal{F}_5, J_5).$$

Hence, an algorithm that solves MaxSAT as in Definition 2.23 must return an interpretation (model) with minimal violated weighted clauses. The similarities between MaxSAT and MIP problems has been discussed in the literature [FM06; MMP09]. It has been shown that it exists efficient state-of-the-art MaxSAT solvers [MML14].

2.2 Periodic Event Scheduling

Whereas SAT, respectively the corresponding automated software, serves as solvers for the problems described in this work, periodic event scheduling and its possible extensions represents the core topic. This section covers the definitions and notations of periodic event networks in Section 2.2.1 and the basic to be solved problem in Section 2.2.2. In Section 2.2.3 the current research for periodic event scheduling is discussed which covers the main point of interest: automatic railway timetabling for a lot of different applications.

2.2.1 Periodic Event Network

Several time scheduling problems have periodic properties [LW82; SU89; BDP04]. Hence, an event does not only happen once in time, but periodically often modulo a time bound. In order to restrict these events in some way, there will be introduced constraints, so called time consuming processes or activities. Connecting events with constraints will result in a graph of a so called periodic event network. If all constraints hold for a certain assignment of all the events, when they periodically happen in time, then this assignment is called valid schedule. More in depth information can be found in the literature [Nac98; Opi09; SU89; Kro+08; Kro+09; LM07b].

Definition 2.26 (Interval)**.** *Let $a, b \in \mathbb{Z}$. Then*

$$[a, b] := \{x \mid a \leq x \leq b\} \subseteq \mathbb{Z}$$

is the interval *from a to b, the lower and upper bound, respectively.*

Further Knowledge of interval arithmetic and linear subspaces can be gained in the literature [Hog06].

Definition 2.27 (Modulo Interval)**.** *Let $a, b \in \mathbb{Z}$ and $t \in \mathbb{N}_+$. Then*

$$[a, b]_T := \bigcup_{z \in \mathbb{Z}} [a + z \cdot T, b + z \cdot T] \subseteq \mathbb{Z}$$

is called interval modulo *T.*

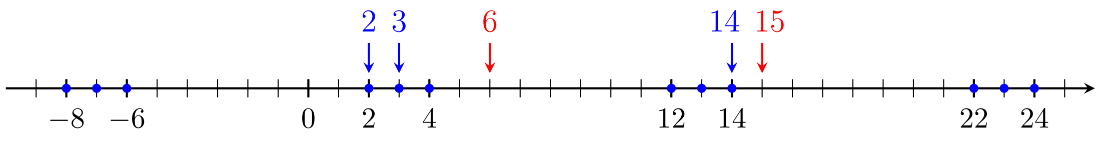


Figure 2.3: Visualization as in Example 2.28

Example 2.28. Let

$$\begin{aligned} I &= [2,4]_{10} \\ &= \ldots \cup [-8,-6] \cup [2,4] \cup [12,14] \cup [22,24] \cup \ldots \\ &= \{\ldots, -8, -7, -6, 2, 3, 4, 12, 13, 14, 22, 23, 24, \ldots\} \subset \mathbb{Z} \end{aligned}$$

be an interval modulo 10. Then

$$\begin{aligned} 2 &\in [2,4] \subset I \\ 3 &\in [2,4] \subset I \\ 6 &\notin I \\ 14 &\in [12,14] \subset I \\ 15 &\notin I. \end{aligned}$$

This can be seen visualized in Figure 2.3.

In the literature [Nac98; Opi09; SU89], z is called modulo parameter.

Definition 2.29 (Periodic Event Network)**.** *The tuple* $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ *is called* periodic event network (PEN) *with respect to period* $T \in \mathbb{N}$ *that consists of a set of periodic events* $\mathcal{V}$ *(nodes) and a set of constraints* $\mathcal{E}$ *(edges). Each constraint* $e \in \mathcal{E}$ *connects two periodic events such that*

$$e = (n, m, \mathcal{Z}_e) \in \mathcal{V} \times \mathcal{V} \times \mathcal{P}(\mathbb{Z}),$$

and $\forall i \in \mathcal{Z}_e, \forall z \in \mathbb{Z} : i + z \cdot T \in \mathcal{Z}_e$.

In the literature, the set $\mathcal{Z}_e$ is typically a modulo interval $[l_e, u_e]_T$. Thus, in many cases the edge e is denoted as $(n, m, [l_e, u_e]_T)$. Opening the intervals with a set of subsets opens a lot of varieties to define cleanly preprocessing and cutting techniques like combining edges between same events or the advanced order encoding presented in Section 2.3.3. The following lemma ensures the correctness of modulo intervals for these type of constraints.

Lemma 2.30. *Let $e = (n, m, [l_e, u_e]_T)$ be a tuple with n and m being periodic events and $\mathcal{Z}_e = [l_e, u_e]_T$ a modulo interval for period T. Then e is a constraint with respect to Definition 2.29.*

Proof. $\forall i \in \mathcal{Z}_e = [l_e, u_e]_T, \forall z \in \mathbb{Z} : i + z \cdot T \in \mathcal{Z}_e$, which follows directly by Definition 2.27. □

However, for several preprocesses and optimization techniques this is always the case. In the sequel, $\mathcal{N}$ denote PENs. It follows the introduction of the semantic part of the periodic event networks. In order to define a schedule, it is needed to assign each event a point in time, when it happens.

For convenience, we introduce the following two mappings in order to access parts of an edge. The function ϵ returns the pair of consecutive events for a constraint e such that

$$\begin{aligned} \epsilon : \mathcal{V} \times \mathcal{V} \times \mathcal{P}(\mathbb{Z}) &\to \mathcal{V} \times \mathcal{V} \\ (n, m, \mathcal{Z}_e) &\mapsto (n, m) \end{aligned}$$

and the function a maps the respective constraint to the integer set such that

$$\begin{aligned} a : \mathcal{V} \times \mathcal{V} \times \mathcal{P}(\mathbb{Z}) &\to \mathcal{P}(\mathbb{Z}) \\ (n, m, \mathcal{Z}_e) &\mapsto \mathcal{Z}_e \end{aligned}$$

Definition 2.31 (Schedule)**.** *Let $\mathcal{V}$ be a set of events and $T \in \mathbb{N}_+$ be the period. The mapping*

$$\begin{aligned} \Pi : \mathcal{V} &\to [0, T-1] \\ n &\mapsto \Pi(n) \end{aligned}$$

is called schedule *for $\mathcal{V}$.*

Each mapped event $\Pi(n)$, $n \in \mathcal{V}$, is called *potential* for event n.

Definition 2.32. *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ be a PEN, $e = (i, j, \mathcal{Z}) \in \mathcal{E}$ be an edge, and Π be a schedule. The constraint e* holds under Π, *denoted as $\Pi \models e$ if and only if*

$$\Pi(j) - \Pi(i) \in \mathcal{Z}.$$

This property is a time consuming process which describes, how much time within

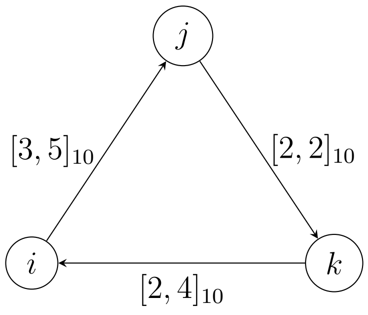


Figure 2.4: Periodic event network as in Example 2.35

$[l, u]$ modulo T is needed from an event i to the event j in order to hold with respect to the constraint e, assuming that the integer interval is a modulo interval.

Example 2.33. Let $e = (n, m, [3,5]_{10})$ be a constraint with period $T = 10$. Then this constraints holds under a schedule $\Pi(n) = 1, \Pi(m) = 5$, because $5 - 1 = 4 \in \mathcal{Z} = [3,5]_{10}$ and as well for a schedule $\Pi(n) = 9, \Pi(m) = 2$, because $2 - 9 = -7 \equiv_{10} 3 \in [3,5]_{10}$.

Definition 2.34 (Valid Schedule)**.** *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ be a PEN and Π be a schedule. Π is* valid *for $\mathcal{N}$, denoted as $\Pi \models \mathcal{N}$, if and only if each constraint $e \in \mathcal{E}$ holds under Π.*

If a valid schedule for a PEN $\mathcal{N}$ exists, then $\mathcal{N}$ is said to be *feasible*. Otherwise, $\mathcal{N}$ is said to be *infeasible*.

Example 2.35. Let $\mathcal{N} = (\{i, j, k\}, \mathcal{E}, 10)$ be a periodic event network with the set of edges $\mathcal{E} = \{(i, j, [3,5]_{10}), (j, k, [2,2]_{10}), (k, i, [2,4]_{10})\}$. This can be displayed as graph which is shown in Figure 2.4.

Additionally, let Π be a schedule with $\Pi = \{i \mapsto 1, j \mapsto 5, k \mapsto 7\}$. Π is valid for $\mathcal{N}$ ($\Pi \models \mathcal{N}$), because

$$\begin{aligned}\Pi(j) - \Pi(i) &= 5 - 1 = 4 \in [3,5]_{10}\\ \Pi(k) - \Pi(j) &= 7 - 5 = 2 \in [2,2]_{10}\\ \Pi(i) - \Pi(k) &= 1 - 7 = -6 \in [-8,-6] \subset [2,4]_{10}.\end{aligned}$$

For further discussions, we introduce the definition of equivalent PENs such that they can be semantically be compared like in Section 2.3.3 for the advanced SAT encoding.

Definition 2.36 (Equivalent PENs)**.** *Let $\mathcal{N}, \mathcal{N}'$ be PENs. Then $\mathcal{N}$ and $\mathcal{N}'$ are* equivalent *(denoted as $\mathcal{N} \equiv \mathcal{N}'$) if and only if for any schedules Π it holds*

$$\Pi \models \mathcal{N} \Leftrightarrow \Pi \models \mathcal{N}'.$$

An Example follows in Section 2.3.3.

2.2.2 Periodic Event Scheduling Problem (PESP)

The SAT problem in Section 2.1.3 is introduced as decision problem. Likewise, the Periodic Event Scheduling Problem (PESP) will be introduced as decision problem as well as in the following definition.

Definition 2.37 (Periodic Event Scheduling Problem (PESP))**.** *Let $\mathcal{N}$ be a PEN. Then, the* Periodic Event Scheduling Problem (PESP) *is the decision problem whether a valid schedule for $\mathcal{N}$ exists.*

Solving a PESP instance can be done in several ways. Firstly, it can be solved by native domain PESP solvers [Nac98; Opi09; Kro+08]. Secondly, encoding it to a linear program (LP) and be solved by a state-of-the-art LP solver [LM07b; Opi09], or thirdly, by the current state-of-the-art solver [Gro+12a] which encodes PESP to SAT and solves the encoded instance by a state-of-the-art SAT solver. This will be briefly presented in the following section. It has been shown before in several ways that PESP is *NP*-complete [SU89; Gro12b; Nac98].

2.2.3 Related Work and Applied Computational Software

This section covers both theoretical aspects and methods in the current research as well as their implementation in applied tools used for computations in real-world scenarios. Since the traditional PESP model has been introduced by Serafini and Ukovich in 1989 [SU89], a lot of researchers and indirectly practitioners have been tackled the problem and enhanced it or its respective solution approaches.

Since a decade the Chair of Traffic Flow Science of Nachtigall works in close collaboration with the DB Netz AG (German railway infrastructure company) and developed the software system TAKT [Nac98; Opi09; Gro+12b; WKO15; Küm+13]. The core module covers railway timetabling with a given train path system and includes conflict resolving techniques [Opi09; Gro12b; Gro13] as well as optimization of generated timetables. Furthermore, traffic passenger flows can be considered in the optimization step [NO08; Opi09]. As enhancement this tool can insert and optimize railway freight train paths into a given train path system [WON12; Küm+15] which will have an in depth discussion in Section 3 and Section 4 which even has recently be enhanced into automated 24 h timetabling [Gro+13].

In the Netherlands the powerful and one of the most enhanced tool DONS have been developed by Kroon, Schrijver, Peeters et al. in collaboration with the dutch railway company NS [Kro+08; Kro+09]. Basically it does an iterative, interactive process with the divided parts covering timetabling and station routing in order to generate a feasible timetable.

Furthermore, they tackled the problem of rescheduling (short-term planning, small disruptions) for rolling stock [Bud+10] as well as rescheduling under massive (large-scaled) disruptions [Vee+14] by considering rerouting, cancellation and delays of train paths. These models are again applied to real-world scenarios of the Dutch railway network.

Recently, the group around Kroon and Peeters modeled a way how to introduce flexible connections (rolling stock and passenger connections) into a cyclic railway timetable and present a solving approach of the given problem [Kro+14]. This new technique is applied in real-world scenarios as the dutch railway network which covers both passenger trains like the intercity network and freight trains.

Minimization of passenger journey times with PESP has been solved for the Danish railway network by Sels et al. [Sel+15].

Schmidt and Schöbel developed an integrated approach for timetabling and passenger routing [SS10; SS15] in order to reduce the delays for passengers which even has been combined with line planning [Sch14]. An approach for integrated (freight train) routing and aperiodic (non-periodic) timetabling has been developed by Cacchiani et al. [CCT10; CGT15].

Caimi, Laumanns et al. enhanced the PESP model by event flexibility [Cai+07] which allows not just a precise point in time for each event (node) in the PEN but a time interval. Yet, all intervals fulfill the needed criteria in order to generate feasible timetables. They transform the problem into a mixed linear program and solve the given instances with MIP solvers. The advantage they created lays in the interactive and iterative process with micro-level scheduling.

2.3 Encoding Periodic Event Networks into SAT

As depicted in Figure 2.1, this section covers the encoding of a PESP instance to a respective SAT instance [Gro11; Gro+12a; Gro12a] which is called as well polynomial reduction [DW83]. Similarly as converting constraint satisfaction problems into SAT, the finite domains of the events can be translated in very different ways. In this work, only the order encoding [TTB11] will be presented. This encoding is superior to other encodings like direct encoding.

For most lemmas only proof sketches are provided. An in-depth analysis of the presented lemmas and definitions and a encoding comparison can be found in the literature [Gro11].

2.3.1 Order Encoding for Variables of Finite Ordered Domains

Since domains in PESP are subsets of $\mathbb{Z}$, especially they are sets of intervals, it is natural to apply the order relation $\leq$ on $\mathbb{Z}$.

Example 2.38. Let $x \in \{1,2,3,4,5\} = [1,5] \subset \mathbb{Z}$ be a variable. Then we know

$$\begin{aligned} & x \geq 1 \Rightarrow x \not\leq 0 \Rightarrow \neg(x \leq 0), && (2.3)\\ & x \leq 5, && (2.4)\\ & \forall i \in \{1,\dots,5\} : (x \leq i-1) \rightarrow (x \leq i) \\ \Leftrightarrow\ & \forall i \in \{1,\dots,5\} : \neg(x \leq i-1) \vee (x \leq i). && (2.5) \end{aligned}$$

With facts (2.3) and (2.4), we can conclude

$$(2.5) \overset{(2.3),(2.4)}{\Rightarrow} \forall i \in \{2,\dots,4\} : \neg(x \leq i-1) \vee (x \leq i).$$

The previous example shows the intension of the order encoding and results in the following definition.

Let $q_{n,i}$ be a propositional variable with $i \in [-1, T-1]$ and $n \in \mathcal{V}$ an periodic event. Then, the variable $q_{n,i}$ is interpreted as $\Pi(n) \leq i$ and $\neg q_{n,i}$ as $\neg(\Pi(n) \leq i)$ which is equivalent[3] to $\Pi(n) \geq i+1$. The function *enc* maps the set of events $\mathcal{V}$ to a propositional formula in CNF, such that this ordering holds:

$$enc(n) = (\neg q_{n,-1} \wedge q_{n,T-1}) \bigwedge_{i \in [0,T-1]} (\neg q_{n,i-1} \vee q_{n,i}) \tag{2.6}$$

This encoding for variables of finite ordered domains is discussed in details by Tanjo et al. [TTB11]. Obviously, the unit clauses[4] could be easily propagated and will directly decrease the amount of clauses and should be considered for possible implementations. However for better a illustration, these clauses are maintained.

In order to encode all events' potentials, which will be the decoded schedule afterwards,

[3]because Π maps to a natural number in $[0, T-1]$
[4]clauses containing only one literal like p or $\neg p$

we define

$$\Omega_{\mathcal{N}} := \bigwedge_{n \in \mathcal{V}} enc(n). \tag{2.7}$$

If a model J has been found for this formula, *extracting the value* of the event n by J is done by the bijective function $\xi_n(J)$ with $\xi_n(J) = k$ such that $J \not\models q_{n,k-1}$, $J \models q_{n,k}$ and $k \in [0, T-1]$.

Example 2.39. Let $x \in [1,5] \subset \mathbb{Z}$ be the variable in Example 2.38. Applying *enc* yields

$$enc(x) = \neg q_{x,0} \wedge q_{x,5} \wedge (\neg q_{x,1} \vee q_{x,2}) \wedge (\neg q_{x,2} \vee q_{x,3}) \wedge (\neg q_{x,3} \vee q_{x,4})$$

Let J be an interpretation with

$$\left.\begin{aligned} q_{x,0}^J &= f,\\ q_{x,1}^J &= f,\\ q_{x,2}^J &= f,\\ q_{x,3}^J &= f, \end{aligned}\right\} i < 4 \to f$$
$$\left.\begin{aligned} q_{x,4}^J &= t,\\ q_{x,5}^J &= t. \end{aligned}\right\} i \geq 4 \to t$$

Then we can extract the value for x by applying ξ, which yields

$$x = 4,$$

because $x \not\leq 3$ ($q_{x,3}^J = f$) and $x \leq 4$ ($q_{x,4}^J = t$).

The mapping ξ_n is well-defined and ensures extracting exactly one value due to the following lemma.

Lemma 2.40. *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ be a PEN, $n \in \mathcal{V}$ be an event and J an interpretation. Then*

(i) $J \models enc(n) \Leftrightarrow \exists! k \in [0, T-1] : \forall i \in [-1, k-1] : J \not\models q_{n,i}$,

(ii) $\forall j \in [k, T-1] : J \models q_{n,j}$.

Proof (sketch). "$\Rightarrow$": Let $J \models enc(n)$. To show:

1. $\exists k \in [l,u] : \forall i \in [l, k-1] : J \not\models q_{n,i}, \forall j \in [k, u-1] : J \models q_{n,j} \quad \wedge$

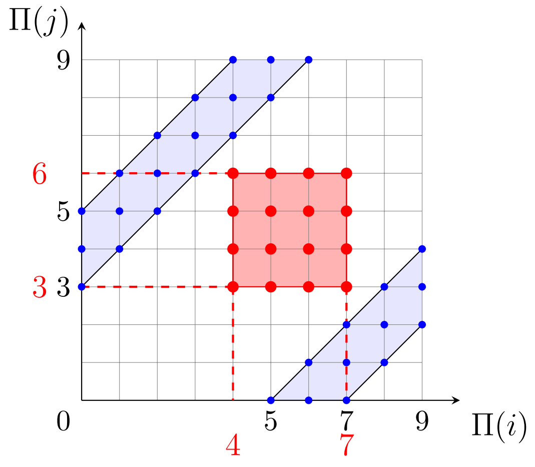


Figure 2.5: Feasible (blue) and infeasible (white) regions for constraint $a(i, j) = [3, 5]_{10}$. The red square shows an infeasible rectangle.

2. $\nexists h \in [l, u], h \neq k : \forall i \in [l, h-1] : J \not\models q_{n,i}, \forall j \in [h, u-1] : I \models q_{n,j}$

1. is simply shown with mathematical induction.
2. is simply shown without loss of generality $h > k$ implies $q^J_{n,k} = f$, which is in contradiction to 1.
"$\Leftarrow$": can be simply shown with mathematical induction as in "$\Rightarrow$". □

Extracting the schedule Π can be done on a per-element basis of a model J with

$$\forall n \in \mathcal{V} : \Pi(n) = \xi_n(J), \Pi = \xi(J) = \{n \mapsto \xi_n(J) \mid n \in \mathcal{V}\}. \tag{2.8}$$

Without proof, we can use an artificial inverse function

$$\xi^{-1}(\Pi) = J \tag{2.9}$$

that returns for the schedule Π the interpretation J that suffices the conditions in (2.8) and Lemma 2.40.

2.3.2 Polynomial Reduction from PESP to SAT

In order to encode a constraint $e = (i, j, [l, u]_T) \in \mathcal{E}$, we take a deeper look at all feasible pairs $(\Pi(i), \Pi(j))$ that hold under e. Uniting all these pairs is called the *feasible region* $S_e := \{(\Pi(i), \Pi(j)) \mid \Pi(j) - \Pi(i) \in [l, u]_T\}$ respectively the union of every other pair the *infeasible region*[5]

$$P_e := [0, T-1] \times [0, T-1] \setminus S_e. \tag{2.10}$$

The intension of the encoding is presented in the following example.

Example 2.41. Let i, j be two events and $e = (i, j, [3,5]_{10})$ be a constraint in Example 2.33 with $\Pi(i)$ and $\Pi(j)$ the potential of i and j, respectively.

A part of the infeasible region would be for instance

$$r = ([4,7] \times [3,6])$$

with $(\Pi(i), \Pi(j)) \notin r$. The to be excluded rectangle r is visualized in Figure 2.5.

Thus, all pairs which are not feasible represented as the set

$$\{(i,j) \mid i \in [4,7], j \in [3,6]\} = r.$$

Then, we know that for each pair $(\Pi(i), \Pi(j)) \notin r$, if e holds for $\Pi(i)$, $\Pi(j)$. With $q_{i,k}$, $k \in [-1, 9]$ and $q_{j,l}$, $l \in [-1, 9]$ being the propositional variables in (2.6), it follows

$$\begin{aligned}
&\nexists(\Pi(i), \Pi(j)) : \Pi(i) \leq 7, \Pi(i) \geq 4, \Pi(j) \leq 6, \Pi(j) \geq 3 \\
\Leftrightarrow\ &\neg((\Pi(i) \leq 7) \wedge (\Pi(i) \geq 4) \wedge (\Pi(j) \leq 6) \wedge (\Pi(j) \geq 3)) \\
\Leftrightarrow\ &\neg((\Pi(i) \leq 7) \wedge \neg(\Pi(i) \leq 3) \wedge (\Pi(j) \leq 6) \wedge \neg(\Pi(j) \leq 2)) \\
\Leftrightarrow\ &\neg(q_{i,7} \wedge \neg q_{i,3} \wedge q_{j,6} \wedge \neg q_{j,2}) \\
\Leftrightarrow\ &(\neg q_{i,7} \vee q_{i,3} \vee \neg q_{j,6} \vee q_{j,2}) =: c
\end{aligned}$$

Since c is a clause, we can directly connect this conjunctively to the resulting formula.

In general, let $e = (i, j, [l, u]_T) \in \mathcal{E}$ be a constraint. Then we can exclude all infeasible pairs of a rectangle $[i_1, i_2] \times [j_1, j_2]$, that is a subset of the infeasible region P_e, by a single clause:

$$\mathit{enc\_rec}([i_1, i_2] \times [j_1, j_2]) = \neg q_{i,i_2} \vee q_{i,i_1-1} \vee \neg q_{j,j_2} \vee q_{j,j_1-1} \tag{2.11}$$

With $\lfloor\cdot\rfloor$ being the round down function, $\lceil\cdot\rceil$ being the round up function and the integers $u, l \in \mathbb{Z}$ with $u < l$, we can define

$$\delta(l, u) := l - u - 1$$

[5]which could be equivalently formulated as $P_e := \{(\Pi(i), \Pi(j)) \mid \Pi(j) - \Pi(i) \notin [l, u]_T\}$

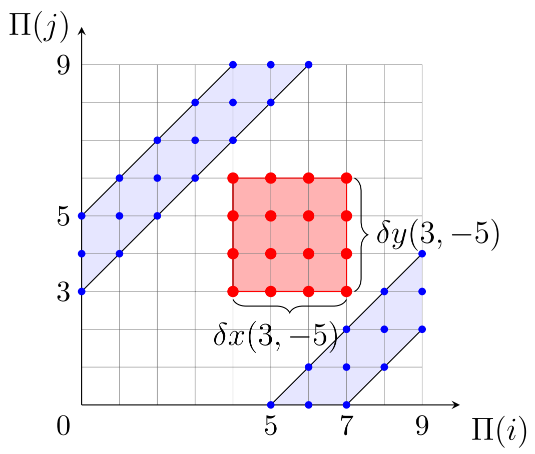


Figure 2.6: Evaluated function values $\delta y(3, -5)$, $\delta x(3, -5)$ as in Example 2.42.

and thus, the height of a rectangle by

$$\delta y(l,u) := \left\lfloor \frac{\delta(l,u)}{2} \right\rfloor$$

and likewise, the width of a rectangle by

$$\delta x(l,u) := \left\lceil \frac{\delta(l,u)}{2} \right\rceil - 1.$$

These definitions allow us to determine a rectangle between u and l, such that it has maximum area having minimum perimeter. Basically each rectangle has a width of $\delta x(l,u)$ and a height of $\delta y(l,u)$.

Example 2.42. Let i, j be two events and $e = (i, j, [3,5]_{10})$ be a constraint in Example 2.33 with $\Pi(i)$ and $\Pi(j)$ the potential of i and j, respectively.
For example, we choose $l = 3$, $u = -5$. Then we can evaluate

$$\begin{aligned}
\delta(3,-5) &= 3 - (-5) - 1 = 7,\\
\delta y(3,-5) &= \left\lfloor \frac{\delta(3,-5)}{2} \right\rfloor = \left\lfloor \frac{7}{2} \right\rfloor = 3,\\
\delta x(3,-5) &= \left\lceil \frac{\delta(3,-5)}{2} \right\rceil - 1 = \left\lceil \frac{7}{2} \right\rceil - 1 = 3.
\end{aligned}$$

The evaluated values are displayed in Figure 2.6.

In order to cover each infeasible pair in the area between $u - T$ and l, we need

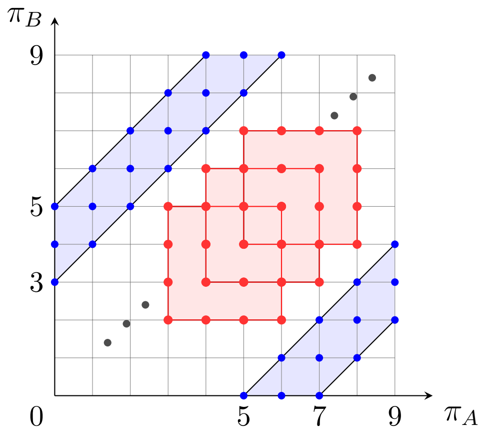


Figure 2.7: Excluded rectangles for constraint $[3, 5]_{10}$.

approximately T rectangles. The function $\zeta : \mathcal{P}(\mathbb{Z}) \to \mathcal{P}(\mathcal{P}(\mathbb{Z}) \times \mathcal{P}(\mathbb{Z}))$,

$$\zeta([l, u]_T) = \{H \times G \mid |H| = \delta x(l, u - T), |G| = \delta y(l, u - T), (H \times G) \cap S_e = \emptyset\}, \quad (2.12)$$

with $H, G \in \mathcal{P}(\mathbb{Z})$ being intervals, maps to the set of all infeasible rectangles of constraint e. The application of ζ to constraint $[3, 5]_{10}$ as in Example 2.42 is depicted in Figure 2.7 which exemplifies the exclusion in the center linear search space. The sufficiency that all infeasible pairs of a constraint are excluded, is given by the following lemma.

Lemma 2.43. *Let $e = (i, j, [l, u]_T)$ be a constraint. Then the following holds*

$$\text{(i)}\quad P_e \subseteq \bigcup_{A \in \zeta(a(e))} A$$

$$\text{(ii)}\quad S_e \cap \bigcup_{A \in \zeta(a(e))} A = \emptyset.$$

The lemma states that in (i) ξ covers the whole infeasible space and in (ii) that ξ is disjoint with the feasible region. The proof is provided in the literature [Gro11].

Definition 2.44 (Encode Constraint)**.** *Let $e = (i, j, [l, u]_T)$ be a constraint. Then*

$$enc\_con : \mathcal{E} \to \mathcal{L}(\mathcal{R})$$
$$e \mapsto \bigwedge_{A \in \zeta([l,u]_T)} enc\_rec(A)$$

is the order encoding mapping *of the edge e.*

Let the encoding of all edges $e \in \mathcal{E}$ be

$$\Psi_{\mathcal{N}} := \bigwedge_{e \in \mathcal{E}} enc\_con(e). \tag{2.13}$$

This results in the encoding of a PESP instance such that

Definition 2.45 (Encoding PESP). *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ be a PEN. Then the function*

$$enc\_pen(\mathcal{V}, \mathcal{E}, T) := \Omega_{\mathcal{N}} \wedge \Psi_{\mathcal{N}} \tag{2.14}$$

is the order encoding function of a PESP *with respect to $\mathcal{N}$.*

It has been shown previously [Gro11] that *enc_pen* maps to propositional formula in CNF and thus, can be easily given to a state-of-the-art SAT solvers.

As proposed and proved in the literature [Gro11] the cardinality of the set of encoded clauses and variables is linear to the period T such that

$$|enc\_pen(\mathcal{N})| \in \mathcal{O}(T \cdot (|\mathcal{V}| + |\mathcal{E}|))$$

for a given PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$.

In the following, the proof for soundness and completeness of the polynomial reduction will be provided. The subsequent lemma ensures that each pair of the encoded rectangle is not satisfiable with respect to the encoded formula and thus, not part of a feasible schedule.

Lemma 2.46. *Let $r = ([i_1, i_2] \times [j_1, j_2]) \subseteq P_e$ be a rectangle in the infeasible region of the constraint $e = (i, j, [l, u]_T)$. Then*

$$J \models enc\_rec(r) \Leftrightarrow (\xi_i(J), \xi_j(J)) \notin r$$

with J being an interpretation.

Proof. "⇒":

$$J \models enc\_rec(r) \Rightarrow (\neg q_{i,i_2} \vee q_{i,i_1-1} \vee \neg q_{j,j_2} \vee q_{j,j_1-1})^J = t \tag{2.15}$$

Proof by contradiction: assume $(\xi_i(J), \xi_j(J)) \in r = ([i_1, i_2] \times [j_1, j_2])$. Then

$$i_1 \leq \xi_i(J) \wedge i_2 \geq \xi_i(J) \wedge j_1 \leq \xi_j(J) \wedge j_2 \geq \xi_j(J)$$

which results in

$$\begin{aligned} & q^J_{i,i_2} = t \wedge q^J_{i,i_1-1} = f \wedge \neg q^J_{j,j_2} = t \wedge q^J_{j,j_1-1} = f \\ \Rightarrow\ & (\neg q_{i,i_2} \vee q_{i,i_1-1} \vee \neg q_{j,j_2} \vee q_{j,j_1-1})^J = f \end{aligned}$$

This is a contradiction to (2.15). Hence,

$$(\xi_i(J), \xi_j(J)) \notin r$$

"$\Leftarrow$": analogous to "$\Rightarrow$". □

It has been shown that the Periodic Event Scheduling Problem can be reduced to satisfiability testing efficiently [Gro+12a] and that the reduction is sound and complete as conducted in the next theorem.

Theorem 2.47 (Soundness Completeness Encoding)**.** *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ be a PEN and*

$$\mathcal{F} := enc\_pen(\mathcal{N}) \in \mathcal{L}(\mathcal{R})$$

be the order encoded propositional formula of $\mathcal{N}$. Then

$$\exists J : J \models \mathcal{F} \Leftrightarrow \exists \Pi : \Pi \models \mathcal{N}$$

with J being an interpretation and Π a schedule of $\mathcal{V}$.

Proof.

$$\begin{aligned} & \exists J : J \models \mathcal{F} \overset{\text{Def. 2.45}}{\Leftrightarrow} \exists J : J \models (\Omega_\mathcal{N} \wedge \Psi_\mathcal{N}) \\ \Leftrightarrow\quad & \exists J : J \models \Omega_\mathcal{N}, J \models \Psi_\mathcal{N} \\ \overset{(2.7)}{\Leftrightarrow}\quad & \exists J : J \models \textstyle\bigwedge_{n \in \mathcal{V}} enc(n), J \models \Psi_\mathcal{N} \\ \overset{\text{Def. } \xi_n\text{,Lem 2.40}}{\Leftrightarrow}\quad & \forall n \in \mathcal{V} : \Pi(n) := \xi_n(J), J \models \Psi_\mathcal{N} \\ \overset{(2.13)}{\Leftrightarrow}\quad & \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_n(J), J \models \textstyle\bigwedge_{e \in \mathcal{E}} \bigwedge_{r \in \zeta(a(e))} enc\_rec(r) \\ \Leftrightarrow\quad & \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_n(J), \forall e \in \mathcal{E}\ \forall r \in \zeta(a(e)) : J \models enc\_rec(r) \\ \overset{\text{Lem. 2.46,2.43}}{\Leftrightarrow}\quad & \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_n(J), \forall e \in \mathcal{E} : e \text{ holds under } \Pi \\ \Leftrightarrow\quad & \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_n(J), \Pi \models \mathcal{N} \\ \overset{\text{Def. } \xi_n\text{,Lem 2.40}}{\Leftrightarrow}\quad & \exists \Pi : \Pi \models \mathcal{N} \end{aligned}$$

□

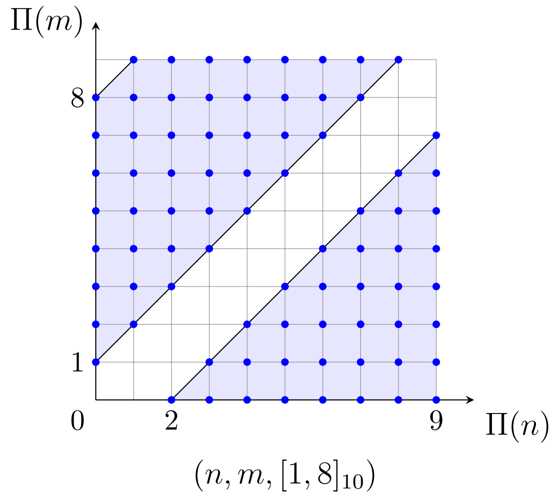


Figure 2.8: Solution space for PEN $\mathcal{N}$ as in Example 2.48 constraint $[1, 8]_{10}$ (left) and $[7, 12]_{10}$ (right).

Solving the SAT instance with a state-of-the-art SAT solver requires a very short time frame compared to a state-of-the-art PESP solver[6] [Gro11; Gro+12a]. The reduction to SAT allows to solve a whole set of larger, more complex instances, which could not be handled before. Consequently, SAT-based solvers should be taken into consideration for possible extensions of PESP, as presented in Section 3 and Section 4.

Generalizing this encoding to all constraints and not just the encoding of constraints with modulo intervals follows in the consecutive section.

2.3.3 Advanced Constraint Encoding

In the previous section, the reduction from PESP to SAT has been introduced for constraints that are based on modulo intervals like $[l, u]_T$. We will extend this by the general constraints given as in Definition 2.29. The advanced encoding will not be proved for soundness and completeness, but follows the same schema as in the proofs in the literature [Gro11].

Example 2.48 (Constraint Comparison)**.** Let

$$\mathcal{N} = (\{n, m\}, \{(n, m, [7, 12]_{10}), (n, m, [1, 8]_{10})\}, 10)$$

be a PEN which is depicted in Figure 2.9. Then an equivalent PEN, with respect to

[6]see Section 5.2

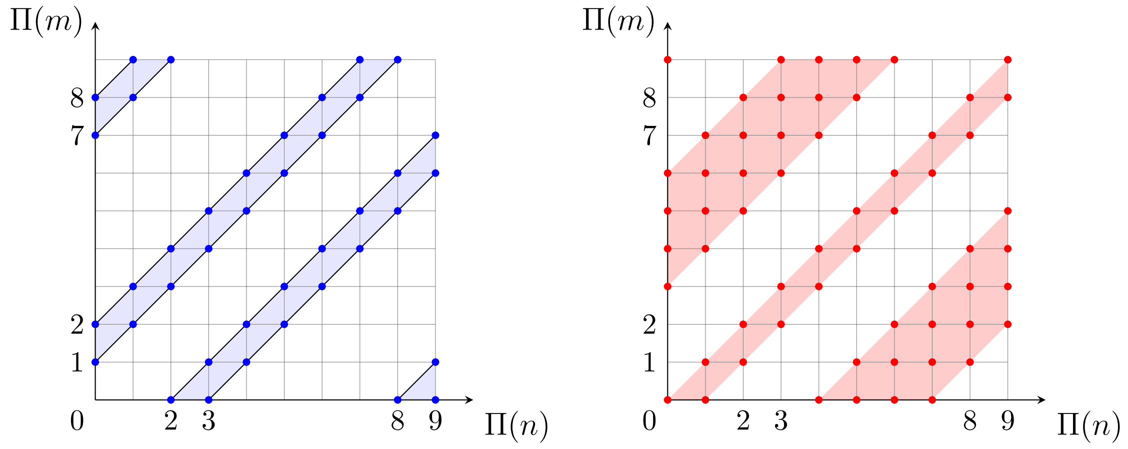


Figure 2.10: Solution space (left) and infeasible space (right) for PEN $\mathcal{N}'$ as in Example 2.48.

solution and search space, would be $\mathcal{N}'$ with $\mathcal{N} \equiv \mathcal{N}'$ such that

$$\mathcal{N}' = (\{n, m\}, \{(n, m, [1, 2]_{10} \cup [7, 8]_{10})\}, 10),$$

because $[1, 8]_{10} \cap [7, 12]_{10} = [1, 2]_{10} \cup [7, 8]_{10}$ which is depicted in Figure 2.9 as well.

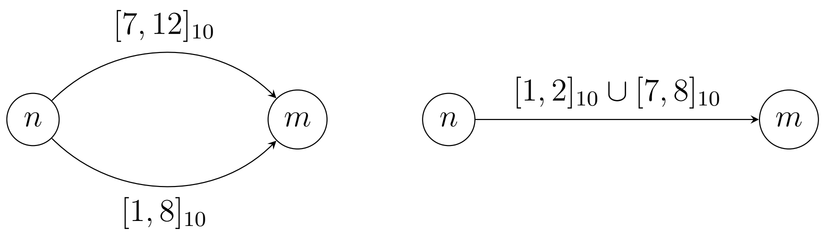


Figure 2.9: PEN $\mathcal{N}$ (left) and $\mathcal{N}'$ (right) as in Example 2.48 with 2 events.

The equivalence of disjunctive constraints is given in the literature [SU89; Opi09]. The solution space of both constraints in $\mathcal{N}$ and $\mathcal{N}'$ is visualized for any given schedule Π for $\mathcal{V}$ in Figure 2.8 and Figure 2.10 (left), respectively. Consequently, as in the previous section, we exclude all rectangles that are part of the infeasible space.

Firstly, we need all intervals of the infeasible space which is the complement set of the disjunctive constraint. With $\mathcal{Z}$ being a constraint for period T as in Definition 2.29 such that $\forall i \in \mathcal{Z}, \forall z \in \mathbb{Z} : i + z \cdot T \in \mathcal{Z}$ we can define

$$\overline{\mathcal{Z}} := \{i \in \mathbb{Z} \mid i \notin \mathcal{Z}\}. \tag{2.16}$$

For instance, the complement of the constraint in $\mathcal{N}'$ of Example 2.48 is depicted in Figure 2.10 (right). Evaluating the complement set for $\mathcal{Z} = [1,2]_{10} \cup [7,8]_{10}$ we get

$$\overline{\mathcal{Z}} = [3,6]_{10} \cup [9,10]_{10}.$$

Subsequently, the size of each rectangle stream has to be evaluated for each linear infeasible area in the constraint. Since we can use the same rectangle encoding, we have to extend the function ζ that maps to all rectangles that shall be excluded in (2.12) by sectioning it to the subintervals as in $\overline{\mathcal{Z}}$. First of all, with (2.16) we can write a disjunctive set of modulo intervals as

$$\mathcal{Z} = [l_0, u_0]_T \cup \ldots \cup [l_{k-1}, u_{k-1}]_T, k \in \mathbb{N}.$$

Furthermore, we extend ζ for a constraint $e = (n, m, \mathcal{Z})$ with the set of modulo intervals $\mathcal{Z} = [l_0, u_0]_T \cup \ldots \cup [l_{k-1}, u_{k-1}]_T, k \in \mathbb{N}$ such that

$$\begin{aligned}\zeta(e) = \{H \times G \mid i \in \{1, \ldots, k\}, u = u_i, l = l_{i+1 \bmod k} :\\ |H| = \delta x(l,u), |G| = \delta y(l,u), |(H \times G) \cap P_{(n,m,\overline{[l,u]_T})}| = |H| \cdot |G|\}\end{aligned} \tag{2.17}$$

with P_e as in (2.10), which maps to all rectangles of the infeasible region.

Finally, we can simply apply *enc_pen* as in Definition 2.45 with the new defined mapping ζ. The soundness and completeness follows directly by the sound definition of ζ that excludes the whole infeasible search space of each constraint. However, this proof will be omitted in this work. The advantage in this encoding is not the reduced amount of rectangles, which in fact are equal, but the additional encoded information into the propositional formula by excluding additional already known infeasible parts of constraints.

Regarding the constraints in Example 2.48 it can be easily concluded that

$$\zeta(n, m, [7,12]_{10}) \cup \zeta(n, m, [3,8]_{10}) = \zeta(n, m, [1,2]_{10} \cup [7,8]_{10})$$

which means that the set of excluded rectangles are exactly the same. Thus, in this example, the encoding of both PENs $\mathcal{N}$ and $\mathcal{N}'$ would be equivalent such that

$$\mathit{enc\_pen}(\mathcal{N}) = \mathit{enc\_pen}(\mathcal{N}')$$

which implies that they are not just semantically equivalent ($\equiv$) yet even syntactically equivalent. Subsequently, to show the difference between the advanced encoding and the encoding in Section 2.3.2 we need an extended example as in the following.

Example 2.49. Let $\mathcal{N}, \mathcal{N}'$ be PENs such that

$$\mathcal{N} = (\{n, m\}, \{(n, m, [0, 6]_{10}), (n, m, [1, 8]_{10}, (n, m, [5, 12]_{10})\}, 10),$$
$$\mathcal{N}' = (\{n, m\}, \{(n, m, [1, 2]_{10} \cup [5, 6]_{10})\}, 10)$$

which implies that $\mathcal{N} \equiv \mathcal{N}'$, since

$$[0, 6]_{10} \cap [1, 8]_{10} \cap [5, 12]_{10} = [1, 2]_{10} \cup [5, 6]_{10}.$$

which implies that

$$\zeta(n, m, [0, 6]_{10}) \cup \zeta(n, m, [1, 8]_{10}) \cup \zeta(n, m, [5, 12]_{10}) \neq \zeta(n, m, [1, 2]_{10} \cup [5, 6]_{10}).$$

We can even conclude that

$$|\zeta(n, m, [0, 6]_{10}) \cup \zeta(n, m, [1, 8]_{10}) \cup \zeta(n, m, [5, 12]_{10})| > |\zeta(n, m, [1, 2]_{10} \cup [5, 6]_{10})|$$

which results in more clauses with respect to the encoding for $\mathcal{N}$ with Definition 2.44.

The previous example shows that the number of clauses are in general less compared to the base encoding. The diverse computational results can be found in Section 5.2.

This encoding offers a lot of application with respect to preprocessing techniques which will not be covered in this work and have already been heavily discussed in the literature [Nac98; Opi09] maintaining equivalence for all applied methods.

3 Flow and Decision Periodic Event Scheduling Problem

The modelling power of PESP has its drawbacks with respect to conditionally ensuring constraints. However, this is highly needed in automatic insertion of rail freight transport paths and other applications alongside the PESP model [Opi09]. These enhanced requirements contain a flow graph whose (flow) edges are connected to nodes of a PEN. Subsequently, we have the power to flood PENs with much more information and possibilities for low granularity of real-world problems. These additional information must be tackled by newly introduced encodings and solution approaches.

The decision and optimization variants as in this section and Section 4, respectively, are treated separately, because even the decision version is *NP*-complete, which will be shown in the following. Thus, it is important to develop an efficient approach that evaluates an initial solution – or schedule – of the optimization beforehand.

Firstly, we introduce elementarily the possibility of conditional constraints and nodes under several given flow graphs in Section 3.1. Secondly, in Section 3.2 we develop the encoding of these enhanced models by propositional formulas and finally, in Section 3.3 the possible applications with respect to public rail transport systems alongside the new models are discussed.

3.1 Periodic Event Scheduling Problem with Flows

This section covers an in-depth analysis of the connection of PENs and flow graphs. Firstly, we enhance the PEN with a decisional structure of constraints and nodes in Section 3.1.1. This means, given a set of decision nodes and edges, these respective nodes and edges may not be regarded with respect to valid schedules and are combined as a so called Decisional Periodic Event Network. Furthermore, the soundness to the existing models will be shown. This offers not solely new possibilities for optimization, yet we need this newly gained model in Section 3.1.2 by connecting decisional nodes to

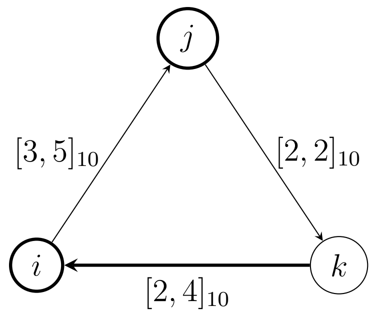


Figure 3.1: Periodic event network as in Example 3.2

flow edges. These flow edges form a flow graph. Consequently, the newly defined valid schedule includes a valid path for each given flow graph and are combined as a so called Flow Decision Periodic Event Network. Again, the soundness will be shown and for both topics a complexity classification will be given.

3.1.1 Decision Periodic Event Scheduling

Before connecting flow graphs with a periodic event network, we introduce a modified version of PESP with respect to PENs by extending the tuple with a set of decision nodes $\mathcal{H}$ and a set of decision edges $\mathcal{A}$.

Definition 3.1 (Decision Periodic Event Network). *Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ be a PEN. Then the tuple $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ is called* decision periodic event network (DPEN) *if $\mathcal{H} \subseteq \mathcal{V}$ and $\mathcal{A} \subseteq \mathcal{E}$.*

This definition allows us to decide, whether certain nodes in $\mathcal{H}$ or certain edges in $\mathcal{A}$ shall be active or not. If they are not active, they shall not be regarded for a valid schedule. The sets $\mathcal{H}$ and $\mathcal{A}$ are called *decision nodes* and *decision constraints* (or edges), respectively.

Example 3.2. Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, 10)$ be a PEN with the set of nodes $\mathcal{V} = \{i, j, k\}$ and the set of edges $\mathcal{E} = \{e_1, e_2, e_3\}$ as in Example 2.35 such that

$$\begin{aligned} e_1 &= (i, j, [3,5]_{10}), \\ e_2 &= (j, k, [2,2]_{10}), \\ e_3 &= (k, i, [2,4]_{10}). \end{aligned}$$

Then with $\mathcal{H} = \{i, j\} \subseteq \mathcal{V}$ and $\mathcal{A} = \{e_3\} \subseteq \mathcal{E}$ we can construct a DPEN $\mathcal{D}$ such that

$$\begin{aligned}\mathcal{D} &= (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A}) \\ &= (\{i, j, k\}, \{e_1, e_2, e_3\}, 10, \{i, j\}, \{e_3\}).\end{aligned}$$

The decision nodes and edges are highlighted in Figure 3.1 alongside the DPEN.

Definition 3.3 (Valid Schedule (extended))**.** *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN, $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes and Π be a schedule for $\mathcal{V}'$. Further let $\mathcal{E}' \subseteq \mathcal{E}$ be a set of constraints of $\mathcal{D}$. Then Π is* valid *for $\mathcal{D}$ and $\mathcal{E}'$ if and only if*

$$\forall n \in \mathcal{V} \setminus \mathcal{H} : n \in \mathcal{V}', \tag{3.1}$$

$$\forall e \in \mathcal{E} \setminus \mathcal{A} : e \in \mathcal{E}' \tag{3.2}$$

and for all $e = (i, j, \mathcal{Z}_e) \in \mathcal{E}'$ it holds

$$i, j \in \mathcal{V}' \Rightarrow \Pi \models e. \tag{3.3}$$

Given the set of active nodes $\mathcal{V}'$ and the set of active edges $\mathcal{E}'$ the previous definition states the following: all constraints must hold under Π if both the two nodes and the constraint are part of the active nodes $\mathcal{V}'$ and active edges $\mathcal{E}'$, respectively. Equivalently, we could require for $\mathcal{V}'$ and $\mathcal{E}'$ with respect to (3.1) and (3.2)

$$(\mathcal{V} \setminus \mathcal{H}) \subseteq \mathcal{V}', \ (\mathcal{E} \setminus \mathcal{A}) \subseteq \mathcal{E}'.$$

In order to set only nodes in $\mathcal{H}$ and edges in $\mathcal{A}$ inactive, both (3.1) and (3.2) must hold, respectively. Hence, only the remaining nodes and edges must hold under Π, which is given in (3.3).

Example 3.4. Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be the DPEN as in Example 3.2 and further let Π_1 be a schedule for $\mathcal{V}_1 = \{i, j, k\}$ and Π_2 be a schedule for $\mathcal{V}_2 = \{i, k\}$ with

$$\begin{aligned}\Pi_1 &= \{i \mapsto 1, j \mapsto 5, k \mapsto 7\}, \\ \Pi_2 &= \{i \mapsto 7, k \mapsto 2\}.\end{aligned}$$

Schedule Π_1 is valid for $\{e_1, e_2, e_3\}$ because both (3.1) and (3.2) holds and Example 2.35 already showed that this schedule is valid.

Schedule Π_2 is valid for $\{e_1, e_2\}$, since

$$\begin{aligned}\mathcal{V} \setminus \mathcal{H} = \{i, j, k\} \setminus \{i, j\} = \{k\} \subseteq \mathcal{V}_2 = \{i, k\} \Rightarrow (3.1)\\ \mathcal{E} \setminus \mathcal{A} = \{e_1, e_2, e_3\} \setminus \{e_3\} = \{e_1, e_2\} \Rightarrow (3.2)\end{aligned}$$

and further:

$$\forall e = (n, m, \mathcal{Z}) \in \{e_1, e_2\} : n \notin \mathcal{V}_2 \text{ or } m \notin \mathcal{V}_2$$

since $n = j$ or $m = j$ and $j \notin \mathcal{V}_2$.
Schedule Π_2 is *not* valid for $\mathcal{E}' = \{e_1, e_2, e_3\}$ because

$$e_3 = (k, i, [2, 4]_{10}) \in \mathcal{E}' \ (k, i \in \mathcal{V}_2)$$

does not hold for Π_2 ($\Pi_2 \not\models e_3$), since for $\Pi_2(i) = 7, \Pi_2(k) = 2$

$$7 - 2 = 5 \notin [2, 4]_T$$

which violates (3.3).

Definition 3.3 is sound with Definition 2.34 with respect to a given, fixed set of nodes $\mathcal{V}'$ and set of edges $\mathcal{E}'$. This will be ensured by the following lemma.

Lemma 3.5 (Soundness Valid Schedule)**.** *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN, $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes, $\mathcal{E}' \subseteq \mathcal{E}$ be a set of constraints and Π be a schedule for $\mathcal{V}'$. Then Π is valid for the PEN $\mathcal{N} = (\mathcal{V}', \mathcal{E}' \setminus \{e \mid e \in \mathcal{E}', \epsilon(e) = (n, m), n \notin \mathcal{V}' \text{ or } m \notin \mathcal{V}'\}, T)$, if Π is valid for $\mathcal{D}$ and $\mathcal{E}'$.*

Proof. Let Π be valid for $\mathcal{D}$ and $\mathcal{E}'$. Then it follows with Definition 3.3 that (3.1), (3.2) and (3.3) holds.
Let $e = (i, j, \mathcal{Z}_e) \in \mathcal{E}' \setminus \{e \mid e \in \mathcal{E}', \epsilon(e) = (n, m), n \notin \mathcal{V}' \text{ or } m \notin \mathcal{V}'\}$ be an arbitrary, but fixed constraint of $\mathcal{N}$.

$$\Rightarrow i, j \in \mathcal{V}' \overset{(3.3)}{\Rightarrow} \Pi(j) - \Pi(i) \in \mathcal{Z}_e$$

and thus, $\Pi \models e$. Since e is arbitrary, it follows with Definition 2.34 that Π is valid for $\mathcal{N}$. □

The set of constraints, whose nodes – respectively one of the nodes – are not in the set of active nodes $\mathcal{V}'$, are inactive with respect to (3.3). Hence, all these edges can be removed from the resulting PEN $\mathcal{N}$, denoted as $\{e \mid e \in \mathcal{E}', \epsilon(e) = (n, m), n \notin \mathcal{V}' \text{ or } m \notin \mathcal{V}'\}$. All remaining edges must hold, as stated in the lemma.

Consequently, without the inactive nodes and inactive edges, the schedule Π is valid for the remaining PEN $\mathcal{N}$. This allows us the successful application of all evaluations and analysis with respect to the valid schedule on the remaining PEN.

Obviously, the easiest way to gain such a valid schedule Π would be, if all decision nodes and decision edges are inactive, such that $\mathcal{V}' = \mathcal{V} \setminus \mathcal{H}$ and $\mathcal{E}' = \mathcal{E} \setminus \mathcal{A}$. However, given a to be minimized objective functional f alongside the DPEN, the empty solution for these sets, is obviously not minimal with respect to the domain-dependent f. A possible application for such an objective functional is given in Section 4.1.2.

Definition 3.6 (Decision Periodic Event Scheduling Problem)**.** *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN and $\mathcal{V}'$, $\mathcal{E}'$ be given as in Definition 3.3. Then the* Decision Periodic Event Scheduling Problem (DPESP) *is the question whether a schedule* Π *for* $\mathcal{V}'$ *exists such that* Π *is valid for* $\mathcal{D}$ *and* $\mathcal{E}'$*.*

A short outlook on the complexity classification can be found in the end of Section 3.1.2 which states that DPESP is *NP*-complete.

3.1.2 Flow Decision Periodic Event Scheduling

In this section, we combine both PENs and flow graphs such that each flow edge will be connected to a periodic event. A priori, we define flow graphs and valid paths and show certain properties in order to simplify the encoding in Section 3.2.2.

Definition 3.7 (Flow Graph)**.** *The graph $K = (R, F)$ with $F \subseteq R \times R$ is called* flow graph (FG) *if it is directed and acyclic.*

In graph theory, this graph is often called flow network or transportation network [Bon08; Hoc12] with each edge's weight set to 1.[1]

Example 3.8 (Flow Graph)**.** Let $K = (R, F)$ be an FG such that

$$\begin{aligned} R &= \{1, \dots, 6\}, \\ F &= \{e_1, \dots, e_7\} \\ &= \{(1,3), (1,4), (2,5), \dots, (5,6)\} \end{aligned}$$

that is depicted in Figure 3.2 which has only directed edges and is obviously acyclic.

[1]Which results in the keyword DAG (directed acyclic graph.

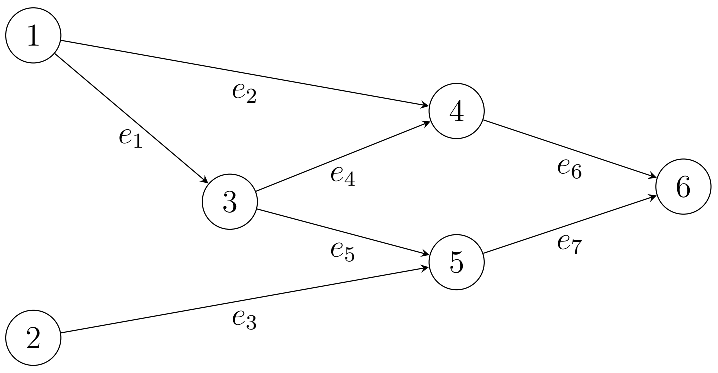


Figure 3.2: Exemplifying flow graph.

Furthermore, we introduce the following two mappings as described in the literature [WON12]. Firstly, let I be the function that maps each node to its set of incoming edges such that

$$\begin{aligned} I : R &\to \mathcal{P}(F) \\ n &\mapsto \{(m,n) \in F \mid \forall n \in R\} \end{aligned}$$

and let O be the function that maps each node to its set of outgoing edges such that

$$\begin{aligned} O : R &\to \mathcal{P}(F) \\ n &\mapsto \{(n,m) \in F \mid \forall n \in R\}. \end{aligned}$$

Given an FG $K = (R, F)$, we can implicitly extract the set of sources

$$S_K := \{n \in R \mid I(n) = \emptyset\} \tag{3.4}$$

and the set of destinations

$$D_K := \{n \in R \mid O(n) = \emptyset\}. \tag{3.5}$$

Often, both functions are indexed with the respective FG K.

Example 3.9. Let $K = (R, F)$ be an FG as in Example 3.8. Then, the set of sources and set of destinations can be evaluated such that $S_K = \{1, 2\}$ and $D_K = \{6\}$, respectively. Evaluating both the functions I and O we can state for the node $4 \in R$ that

$$I_K(4) = \{e_2, e_4\} \text{ and } O_K(4) = \{e_6\}.$$

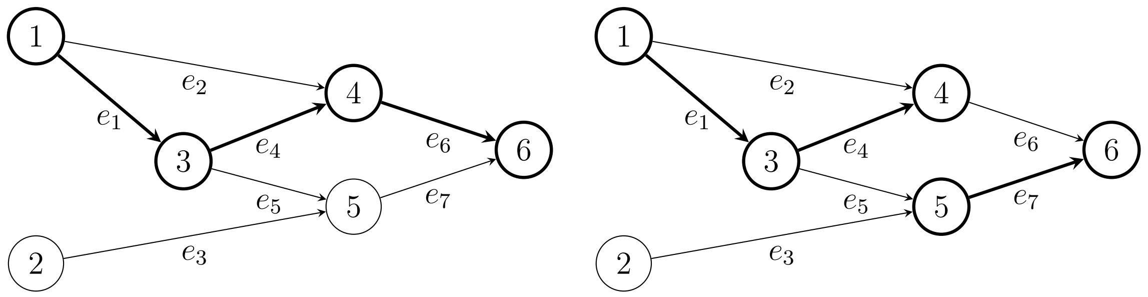


Figure 3.3: Sequences of flow edges ρ_3 (left) and ρ_2 (right) as in Example 3.11.

Definition 3.10 (Path)**.** *Let $K = (R, F)$ be an FG and $\rho = (e_1, \dots, e_k) \in F^k$, $k \in \mathbb{N}$ be a sequence of edges with $e_i = (n_i, n_{i+1})$ $(i \in \{1, \dots, k\})$. Then ρ is called* path *for K if and only if*

$$n_1 \in S_K \tag{3.6}$$

$$n_{k+1} \in D_K \tag{3.7}$$

$$\forall i \in \{2, \dots, k+1\} : (n_{i-1}, n_i) \in F \tag{3.8}$$

with the set of sources S_K and set of destinations D_K as in (3.4) *and* (3.5)*, respectively.*

In other words, a tuple (or sequence) of edges is only a path [Hoc12], if the first node is a source node (Equation (3.6)), the last node is a destination node (Equation (3.7)) and for all nodes in between it holds that it exists a previous and succeeding edge in the set of edges F, which is stated in (3.8) that ensures the flow conservation. Since an FG is acyclic, there is no possibility of a cycle within a path. Hence, we can implicitly assume (without proof) that the number of edges of a path is finite or $k < \infty$ and that each edge only occurs once.

Example 3.11 (Sequences)**.** Let $K = (R, F)$ be an FG as in Example 3.8. Further, let $\rho_1 = (e_5, e_7)$, $\rho_2 = (e_1, e_4, e_7)$ and $\rho_3 = (e_1, e_4, e_6)$ be sequences of flow edges. We can imply that ρ_1 is not a path, because $e_5 = (3, 5)$ and $3 \notin S_K$ and thus, (3.6) is violated. Likewise, ρ_2 is not a path, because $e_4 = (3, 4) \in F$, but the next edge in the sequence is $e_7 = (5, 6) \in F$ and $4 \neq 5$. Hence, (3.8) is violated.
Furthermore, ρ_3 is a path, because

$$e_1 = (1, 3),\ 1 \in S_K \Rightarrow (3.6)$$

$$e_6 = (4, 6),\ 6 \in D_K \Rightarrow (3.7)$$

$$e_1 = (1, 3) \in F, e_4 = (3, 4) \in F, e_6 = (4, 6) \in F \Rightarrow (3.8)$$

and thus, all conditions in Definition 3.10 are fulfilled. Both sequences ρ_2 and ρ_3 can be seen in Figure 3.3.

It is sufficient to have the respective set of edges given, in order to extract the corresponding path. These two structures can be combined by introducing the binary relation χ_K such that

$$\chi_K := \{(H, \rho) \mid \rho = (e_1, \ldots, e_k) \in F^k \text{ path of } K : H = \{e_1, \ldots, e_k\}\} \tag{3.9}$$

Example 3.12. Let $K = (R, F)$ be an FG as in Example 3.8 and let $\rho = (e_1, e_4, e_6) \in F^k$ be a path of K. Then the respective set of edges is $H = \{e_1, e_4, e_6\}$ such that $(H, \rho) \in \chi_K$.

The sufficiency of the set of edges with respect to the relation χ_K is ensured by the following lemma.

Lemma 3.13 (Path Extraction). *Let $K = (R, F)$ be an FG and $H \subseteq F$ be a set of edges and let $k := |H|$. Then it holds with $\rho = ((n_1, n_2), \ldots, (n_k, n_{k+1}))$*

$$\left.\begin{array}{l} \exists!(n, m) \in H : n \in S_K, n_1 := n, \\ \exists!(n, m) \in H : m \in D_K, n_{k+1} := m, \\ \forall i \in \{2, \ldots, k\} \exists!(n_{i-1}, n) \in H : (n_{i-1}, n) \in F, n_i := n \end{array}\right\} \Leftrightarrow (H, \rho) \in \chi_K.$$

Proof. "$\Leftarrow$": $(H, (n_1, n_2), \ldots, (n_k, n_{k+1})) \in \chi_K$. Hence, with (3.9) it follows

$$H = \{(n_1, n_2), \ldots, (n_k, n_{k+1})\}.$$

To show: it must hold

$$\exists!(n, m) \in H : n \in S_K, n_1 := n, \tag{3.10}$$

$$\exists!(n, m) \in H : m \in D_K, n_{k+1} := m, \tag{3.11}$$

$$\forall i \in \{2, \ldots, k\} \exists!(n_{i-1}, n) \in H : (n_{i-1}, n) \in F, n_i := n \tag{3.12}$$

Let be $\rho = ((n_1, n_2), \ldots, (n_k, n_{k+1}))$. With (3.6) it follows that $n_1 \in S_K$ is a source node. All following edges on the path are successor edges (by (3.8)). Since K is directed and acyclic (Definition 3.7), n_1 is the only source node. Hence, it exists exactly one source node. Likewise, this can be stated for (3.11) and n_{k+1}.
(3.12) follows with (3.8) and $n = n_i$.

"⇒": Let H be a set of edges with $k = |H|$ and

$$\exists!(n,m) \in H : n \in S_K, n_1 := n, \tag{3.13}$$

$$\exists!(n,m) \in H : m \in D_K, n_{k+1} := m, \tag{3.14}$$

$$\forall i \in \{2,\dots,k\} \exists!(n_{i-1},n) \in H : (n_{i-1},n) \in F, n_i := n \tag{3.15}$$

To show: $\rho = ((n_1,n_2),\dots,(n_k,n_{k+1})) \in F^k$ is a path of K as in Definition 3.10.
(3.6) and (3.7) follow directly by (3.13) and (3.14), respectively. Equation (3.8) follows with (3.15). □

Lemma 3.13 iteratively constructs the path in the last condition of the equivalence. Given the source node and destination node in the first two conditions, ensures a valid path such that $(H,\rho) \in \chi_K$. The relation χ_K ensures the correct usage of paths and their respective sets. Hence, for convenience the set H belonging to a path ρ with $(H,\rho) \in \chi_K$ will be called path as well.

The multiplicity of an FG is implicitly *one*, because it is not explicitly given.[2] This results in the fact that it must be exactly one path inserted, given that the instance is a decision problem. In general, there may be not just one FG for a DPEN, but a set of FGs.

In order to connect paths and their respective edges with DPENs and their respective events, we introduce a *bijective label function* with $\mathcal{H}$ being a set of events and F being an set of flow edges and $|\mathcal{H}| = |F|$ such that

$$\begin{aligned} l : F &\to \mathcal{H} \\ e &\mapsto n \end{aligned} \tag{3.16}$$

with $range(l) = \mathcal{H}$ being the range of l. Since l is bijective, we can easily get the flow edge of an event by the invert function l^{-1}. Subsequently, a set of edges H that represents a path may be equivalently given as set of periodic events and will be used accordingly.

Example 3.14 (Label Function). Let $K = (R,F)$ be an FG as in Example 3.8 and $\mathcal{H}$ be a set of nodes such that $\mathcal{H} = \{n_1,\dots,n_7\}$. Then, for example, we label each flow edge such that $l(e_i) := n_i$ with $e_i \in F$ and $n_i \in \mathcal{H}$. For example, we can use the invert function for node n_3 such that $l^{-1}(n_3) = e_3$ as depicted in Figure 3.4.

In the sequel, in many cases an application specific label function will be implicitly

[2]and not needed, either, because we can achieve this goal via the approach in Section 3.3.2

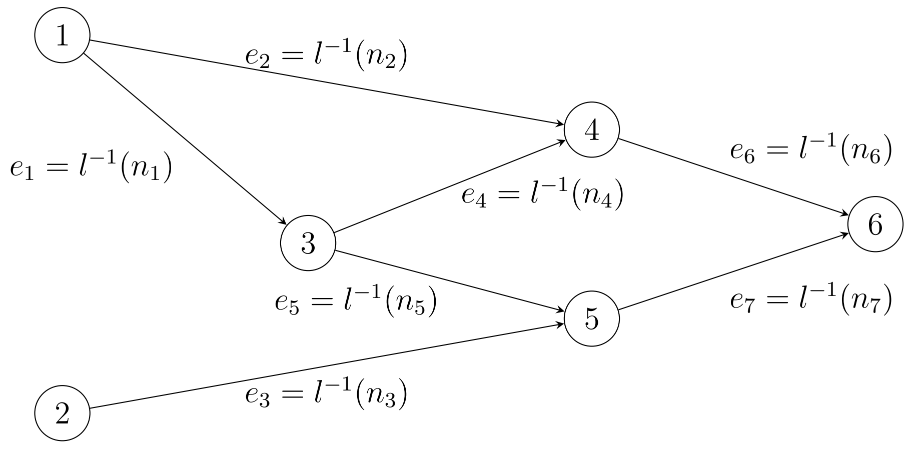


Figure 3.4: Labeled flow graph.

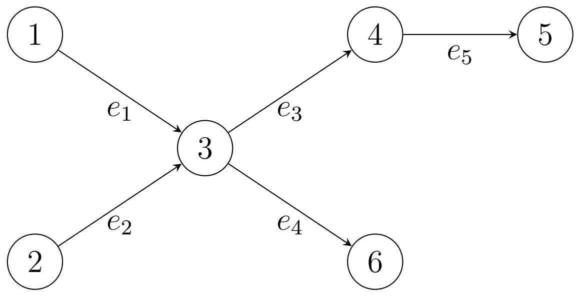


Figure 3.5: Flow graph as in Example 3.16.

presupposed and will not be explicitly required in the following definitions, lemmas and theorems.

Definition 3.15 (Flow Decision Periodic Event Network)**.** *Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN. Then $\mathcal{D}$ is a* flow decision periodic event network (FDPEN) *for $\mathcal{S}$ if and only if*

$$\bigcup_{K=(R,F)\in\mathcal{S}} \bigcup_{e\in F} l(e) \subseteq \mathcal{H}.$$

The definition basically requires that all the labeled events of their respective flow edges are contained[3] in the decision node set $\mathcal{H}$. Please note that the FGs may not be disjoint and thus, some set of nodes may intersect. This opens a lot of opportunities but never the less, will not be part of this work's applications. However, the following definitions are well-defined and sound for the case of intersecting FGs.

[3]which can be equivalently defined as $\forall K = (R, F) \forall e \in F : l(e) \in \mathcal{H}$

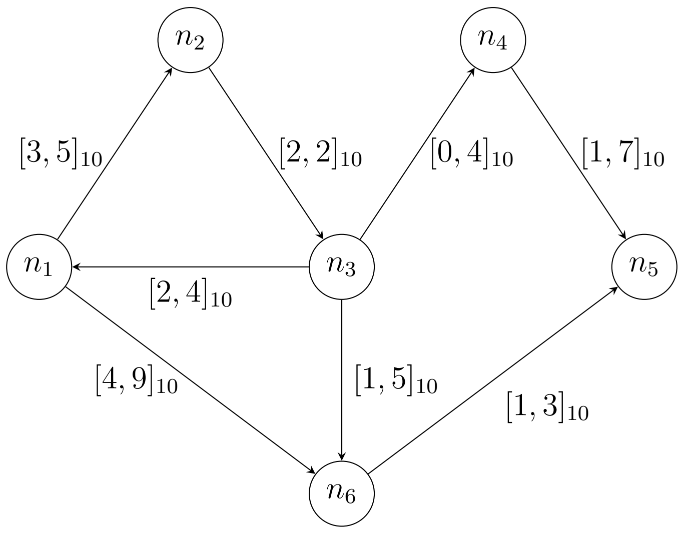


Figure 3.6: PEN as in Example 3.16.

Example 3.16 (FDPEN)**.** Let $K = (R, F)$ be an FG with

$$R = \{1, \ldots, 6\},$$
$$F = \{e_1, \ldots, e_5\}$$

that is depicted in Figure 3.5 and let l be the labeling function for the given set of nodes $\mathcal{H} = \{n_1, \ldots, n_5\}$ of Example 3.14 such that $l(e_i) = n_i$ $(i \in \{1, \ldots, 5\})$.
Further, let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, 10, \mathcal{H}, \emptyset)$ be a DPEN with $\mathcal{V} = \{n_1, \ldots, n_6\}$ and $\mathcal{E}$ as depicted in Figure 3.6. Then, $\mathcal{D}$ is an FDPEN for the set of FGs {K}. Please note, that node n_6 is not labeled and even not in the set of decision nodes.

Definition 3.17 (Valid Schedule (extended))**.** *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for the set of FGs $\mathcal{S}$. Further let $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes, $\mathcal{E}' \subseteq \mathcal{E}$ be a set of edges and* Π *be a schedule of $\mathcal{V}'$. Then* Π *is* valid *for $\mathcal{D}$, $\mathcal{E}'$ under $\mathcal{S}$ if and only if* Π *is valid*[4] *for $\mathcal{D}$ and $\mathcal{E}'$ and it holds for all $K = (R, F) \in \mathcal{S}$ that*

$$\exists! n \in \mathcal{V}' \cap range(l), (m, o) = l^{-1}(n) \in F : m \in S_K, \tag{3.17}$$

$$\exists! n \in \mathcal{V}' \cap range(l), (m, o) = l^{-1}(n) \in F : o \in D_K, \tag{3.18}$$

$$\forall n \in R, n \notin D_K \cup S_K : \left| \bigcup_{(m,n) \in I_K(n)} l(m, n) \cap \mathcal{V}' \right| = \left| \bigcup_{(n,o) \in O_K(n)} l(n, o) \cap \mathcal{V}' \right| \in \{0, 1\}. \tag{3.19}$$

[4]as in Definition 3.3

Hence, a schedule is only valid, if it suffices the general flow constraints for each FG: On the one hand, (3.17) and (3.18) states that it must be exactly one source node and destination node in $\mathcal{V}'$ (their respective labels), respectively. More precisely, exactly one outgoing edge of the source nodes and one ingoing edge of the destination nodes must be in the labeled set $\mathcal{V}'$. On the other hand, (3.19) ensures the flow conservation, which means that the number of incoming edges for a flow node must be the number of outgoing edges alongside the resulting path[5].

Example 3.18 (Valid Schedule)**.** Let $K = (R, F)$ be an FG and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, 10, \mathcal{H}, \emptyset)$ be an FDPEN for $\{K\}$ as in Example 3.16 for the set of decision nodes $\mathcal{H} = \{n_1, \ldots, n_5\}$ and the given label function l. Then, with the set of nodes $\mathcal{V}' = \{n_2, n_3, n_5, n_6\}$, the set of edges $\mathcal{E}' = \mathcal{E}$ and the schedule Π for $\mathcal{V}'$ with

$$\begin{aligned}\Pi(n_2) &= 0,\\ \Pi(n_3) &= 2,\\ \Pi(n_5) &= 6,\\ \Pi(n_6) &= 5,\end{aligned}$$

we know that Π is valid for $\mathcal{D}$ and $\mathcal{E}'$, because (3.1)[6], (3.2) hold and

$$\begin{aligned}\Pi(n_3) - \Pi(n_2) = 2 \in [2,2]_{10} &\Rightarrow \Pi \models (n_2, n_3, [2,2]_{10})\\ \Pi(n_6) - \Pi(n_3) = 4 \in [1,5]_{10} &\Rightarrow \Pi \models (n_3, n_6, [1,5]_{10})\\ \Pi(n_5) - \Pi(n_6) = 1 \in [1,3]_{10} &\Rightarrow \Pi \models (n_6, n_5, [1,3]_{10})\end{aligned}$$

as in Definition 3.3. Furthermore, all equations in Definition 3.17 are fulfilled, because

$$\begin{aligned} n_2 \in \mathcal{V}' \cap \mathit{range}(l) = \{n_2, n_3, n_5\},\ l^{-1}(n_2) = (2,3) \in F,\ 2 \in S_K &\Rightarrow (3.17),\\ n_5 \in \mathcal{V}' \cap \mathit{range}(l),\ l^{-1}(n_5) = (4,5) \in F,\ 5 \in D_K &\Rightarrow (3.18),\\ \bigcup_{(m,3) \in I_K(3)} l(m,3) = \{n_1, n_2\}, \quad \bigcup_{(3,o) \in O_K(3)} l(3,o) = \{n_3, n_4\} &\\ \Rightarrow |\{n_1, n_2\} \cap \mathcal{V}'| = |\{n_3, n_4\} \cap \mathcal{V}'| = 1 \in \{0,1\} & \end{aligned} \tag{3.20}$$

[5]its labeled events
[6]$n_6 \in \mathcal{V}'$

$$\bigcup_{(m,4)\in I_K(4)} l(m,4) = \{n_3\}, \quad \bigcup_{(4,o)\in O_K(4)} l(4,o) = \{n_5\}$$
$$\Rightarrow |\{n_3\} \cap \mathcal{V}'| = |\{n_5\} \cap \mathcal{V}'| = 1 \in \{0,1\} \tag{3.21}$$

with *range*$(l) = \mathcal{H}$ being the range of the label function l. (3.20) and (3.21) imply (3.19), because $\{3,4\}$ are the flow nodes of the FG that are not source or destination nodes. Hence, Π is valid for $\mathcal{D}$, $\mathcal{E}'$ under $\{K\}$.
Finally, we can extract a path ρ from $\mathcal{V}'$ such that $\rho = (e_2, e_3, e_5)$, because

$$\forall n \in \mathcal{V}' \cap \mathit{range}(l) = \{n_2, n_3, n_5\} : l(n) \in F.$$

The following lemma ensures that a valid schedule guarantees exactly one path with respect to each flow graph in the FDPEN.

Lemma 3.19 (Valid Path). *Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and let $\mathcal{E}'$ be given as in Definition 3.3. Further let $\mathcal{V}'$ be a set of nodes and Π be a schedule for $\mathcal{V}'$ such that Π is valid for $\mathcal{D}$, $\mathcal{E}'$ under $\mathcal{S}$. Then it holds for all $K = (R,F) \in \mathcal{S}$ that*

$$\exists! H \subseteq F, k = |H| : H = \Big(\bigcup_{n\in\mathcal{V}'\cap \mathit{range}(l)} l^{-1}(n)\Big) \cap F = \{e_1, \ldots, e_k\},$$
$$\rho = (e_1, \ldots, e_k), (H, \rho) \in \chi_K.$$

Proof. Let $K = (R,F) \in \mathcal{S}$ be an arbitrary but fixed FG. Firstly, we have to show that at least one such H exists and with Lemma 3.13 we know that it is sufficient to show that H must suffice

$$\exists!(n,m) \in H : n \in S_K, n_1 := n, \tag{3.22}$$
$$\exists!(n,m) \in H : m \in D_K, n_{k+1} := m, \tag{3.23}$$
$$\forall i \in \{2, \ldots, k\} \exists!(n_{i-1}, n) \in H : (n_{i-1}, n) \in F, n_i := n, \tag{3.24}$$

in order to have a path ρ with $(H, \rho) \in \chi_K$.

Secondly, we have to show that it exists not more than one H such that

$$\begin{aligned} M := \{H \mid H \subseteq \Big(\bigcup_{n\in\mathcal{V}'\cap \mathit{range}(l)} l^{-1}(n)\Big) \cap F, H = \{e_1, \ldots, e_k\}, k = |H|,\\ (H, (e_1, \ldots, e_k)) \in \chi_K\}, |M| \le 1. \end{aligned} \tag{3.25}$$

with M being the set of extracted paths.
Let be $H = \mathcal{V}' \cap \bigcup_{e \in F} l(e)$. Both (3.22) and (3.23) follow directly by (3.17) and (3.18), respectively.

With mathematical induction we can show that (3.24) holds: For the basis ($i = 2$), we know with (3.17) that it exists exactly one $(n_1, n_2) \in F$ with $n_1 \in S_K$ and hence, with (3.19) that

$$\left| \bigcup_{(m,n_2) \in I_K(n_2)} l(m, n_2) \cap \mathcal{V}' \right| = \left| \bigcup_{(n_2,o) \in O_K(n_2)} l(n_2, o) \cap \mathcal{V}' \right| = 1.$$

Thus, with the previous deduction we know that $m = n_1$ and since the cardinality of the sets equals 1 we know that exactly one such $(n_1, n_2) \in F$ exists that suffices (3.24) for $i = 2$. For the inductive step, we can easily show with the same technique that (3.24) holds: let the inductive hypothesis hold such that (3.24) holds for step i. Thus, we know that it exists exactly one $(n_{i-1}, n_i) \in H$. With (3.19) we know that

$$\left| \bigcup_{(m,n_i) \in I_K(n_i)} l(m, n_i) \cap \mathcal{V}' \right| = \left| \bigcup_{(n_i,o) \in O_K(n_i)} l(n_i, o) \cap \mathcal{V}' \right| = 1,$$

since with the inductive hypothesis for $m = n_{i-1}$ $|\bigcup_{(m,n_i) \in I_K(n_i)} l(m, n_i) \cap \mathcal{V}'| = 1$ (it exists exactly one such flow edge) is ensured. Subsequently, we can set $n_{i+1} = o$ which ensures step $i + 1$

We can indirectly show (3.25) by assuming that

$$M \geq 2.$$

Thus, M has at least 2 paths. Subsequently, we have possibly two different source nodes or destination nodes, which is a contradiction to (3.17) and (3.18), respectively. If the source nodes and destination nodes are the same, then there must exist at least one edge $(n, m) \in F$ in between with: $|\{e \mid e \in O_K(m) \cap \bigcup_{n \in \mathcal{V}' \cap range(l)} l^{-1}(n)\}| \geq 2$ which is a contradiction to (3.19). Likewise, this can be stated for the incoming edges. Thus, $M \leq 1$. Even parts of paths (sub-paths) cannot be in the resulting labeled node set, because it would contradict the given equations.

Finally, since at least one such H exists, we know that $M \geq 1$ and with (3.25) it follows directly that $M = 1$. Hence, it exists exactly one such H.

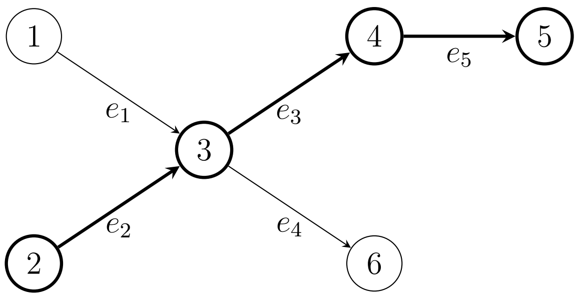


Figure 3.7: Path as in Example 3.20.

Since K is arbitrary, it holds for all $K \in \mathcal{S}$. □

The proof gives us an constructive instrument on extracting the path H for each FG (R, F) with

$$H := \mathcal{V}' \cap \bigcup_{e \in F} l(e) \tag{3.26}$$

given the set of nodes $\mathcal{V}'$. Please note as stated above that the path maybe represented as set of periodic events as in here, since the label function l is bijective. Thus, we could equivalently evaluate H such that

$$H = \Big(\bigcup_{n \in \mathcal{V}' \cap range(l)} l^{-1}(n) \Big) \cap F. \tag{3.27}$$

Example 3.20. As in Example 3.18 let Π be the schedule with $\mathcal{V}' = \{n_2, n_3, n_5, n_6\}$ for the FG K. Then with (3.27) we get

$$H = \{e_2, e_3, e_5, e_6\} \cap \{e_1, \ldots, e_5\} = \{e_2, e_3, e_5\}.$$

Then with Lemma 3.19 we can safely extract the respective path $\rho = (e_2, e_3, e_5)$ as depicted in Figure 3.7 such that $(H, \rho) \in \chi_K$.

With respect to the further encodings we will use an alternative definition for valid schedules under an FG set. In order to have the same results as in Lemma 3.19 the following theorem is given.

Theorem 3.21 (Alternative Valid Schedule)**.** *Let $\mathcal{S}$ be a set of FGs and $\mathcal{D}$ be an FDPEN with $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ for $\mathcal{S}$ and let $\mathcal{E}'$ be given as in Definition 3.3. Further let $\mathcal{V}'$ be a set of nodes and Π be a schedule for $\mathcal{V}'$ such that Π is valid for $\mathcal{D}$, $\mathcal{E}'$. Further, it* *holds* (3.17), (3.18). *Then the following statements are equivalent for all* $K = (R, F) \in \mathcal{S}$*:*

$$(i):$$

$$\forall n \in R, n \notin D_K \cup S_K : \left| \bigcup_{(m,n)\in I_K(n)} l(m,n) \cap \mathcal{V}' \right| = \left| \bigcup_{(n,o)\in O_K(n)} l(n,o) \cap \mathcal{V}' \right| \in \{0,1\}. \tag{3.28}$$

$$(ii):$$

$$\forall n \in R \setminus (S_K \cup D_K) : \left| \bigcup_{e\in I_K(n)} l(e) \cap \mathcal{V}' \right| \leq 1, \tag{3.29}$$

$$\forall n \in R \setminus (S_K \cup D_K) : \left| \bigcup_{e\in O_K(n)} l(e) \cap \mathcal{V}' \right| \leq 1, \tag{3.30}$$

$$\forall (n,m) \in F, m \notin D_K : \Big(l(n,m) \in \mathcal{V}' \Rightarrow \exists o \in \mathcal{V}' \cap \mathit{range}(l) : l^{-1}(o) \in O_K(m) \Big) \tag{3.31}$$

with (3.28) *as in* (3.19).

Proof. Let $K = (R, F) \in \mathcal{S}$ be an arbitrary but fixed FG and let

$$A(n) := \bigcup_{(m,n)\in I_K(n)} l(m,n) \cap \mathcal{V}',$$
$$B(n) := \bigcup_{(n,m)\in O_K(n)} l(n,m) \cap \mathcal{V}'.$$

"$(i) \Rightarrow (ii)$": Let (3.28) hold. Then, (3.29) and (3.30) directly holds, because we know $A(n), B(n) \in \{0,1\}$. If it exists a flow edge $(n,m) \in F, m \notin D_K$ with $l(n,m) \in \mathcal{V}'$, then $A(m) = 1$. With (3.28) we know that $B(m) = 1$ which results that (3.31) holds.

"$(ii) \Rightarrow (i)$": Let (3.29), (3.30), (3.31) hold. Then we know for all flow edges with (3.29) and (3.30) that

$$A(n) \in \{0,1\} \text{ and } B(n) \in \{0,1\}, \tag{3.32}$$

respectively, for an arbitrary but fixed flow node $n \in R, n \notin D_K \cup S_K$. It remains to show that $|A(n)| = |B(n)|$.
We have to show two cases

$$(1): \ |A(n)| = 1 \Rightarrow |B(n)| = 1$$

$$(2): \ |B(n)| = 1 \Rightarrow |A(n)| = 1$$

(1): If it exists a flow edge $(m, n) \in F, n \notin D_K$ with $l(m, n) \in \mathcal{V}'$, then we know firstly, that $|A(n)| = 1$ and secondly, with (3.31) that $|B(n)| \geq 1$. Since we already stated in (3.32) that $B(n) \in \{0, 1\}$ we know that $B(n) = 1$.
(2): With (3.18) we know that exactly one destination is allowed and with (3.29) that $A(n) \leq 1$ for any node m in $R \setminus (S_K \cup D_K)$. Since $|B(n)| = 1$ there must be a series of edges (see (3.31)) from a source node (see (3.17)) such that it exists a flow edge $(o, n) \in F$ with $l(o, n) \in \mathcal{V}'$ and thus, $|A(n)| = 1$.

Because (1) and (2) holds, we can conclude that

$$A(n) = 1 \Leftrightarrow B(n) = 1. \tag{3.33}$$

For the second case ($|A(n)| = |B(n)| = 0$) we do the proof by contradiction divided again into two parts:

$$(1): \ |A(n)| = 0 \Rightarrow |B(n)| = 0 \tag{3.34}$$
$$(2): \ |B(n)| = 0 \Rightarrow |A(n)| = 0 \tag{3.35}$$

(1): Let be $|A(n)| = 0$. We assume that $|B(n)| = 1$. Then, with (3.33) it follows that $|A(n)| = 1$ which is a contradiction to the assumption. Thus, $|B(n)| \neq 1$ and with (3.32) ($|B(n)| \in \{0, 1\}$) we can conclude that $|B(n)| = 0$.
(2): We assume that $|A(n)| = 1$. Then with (3.31) it follows that $|B(n)| = 1$ which is a contradiction to the assumption in (3.35). Hence, $|A(n)| \neq 0$ and with (3.32) ($|A(n)| \in \{0, 1\}$) we can conclude that $|A(n)| = 0$.
Because (1) and (2) hold, we can conclude that

$$A(n) = 0 \Leftrightarrow B(n) = 0. \tag{3.36}$$

Finally, with (3.36) and (3.33) we can deduct (3.28). Since n is arbitrary, it holds for all nodes. Since K is arbitrary, it holds for all $K \in \mathcal{S}$. □

Equations (3.29) and (3.30) are stating for all flow nodes it must hold that *at most one* incoming and outgoing flow edge (its corresponding labeled events) in the resulting path exist, respectively. Having *at least one* successor flow edge in the path if an incoming flow edge of a flow node is in the path is stated in (3.31).

As described before, Theorem 3.21 does not require (3.19) to hold. Rather Equations (3.29), (3.30) and (3.31) are introduced which serve the same purpose such that it exists exactly one path for each FG for a valid schedule. This is given because the theorem ensures the equivalence of both schedule definitions for FGs and thus, Lemma 3.19 can be applied.

Knowing that all constraints corresponding to their paths hold with Lemma 3.5, we can conclude the following soundness theorem.

Theorem 3.22 (Soundness Valid Schedule). *Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and let $\mathcal{E}'$ be given as in Definition 3.3. Further let $\mathcal{V}'$ be a set of nodes and Π be a schedule for $\mathcal{V}'$ such that Π is valid for $\mathcal{D}$, $\mathcal{E}'$ under $\mathcal{S}$. Then it holds for all $K = (R, F) \in \mathcal{S}$ with $H = \mathcal{V}' \cap \bigcup_{e \in F} l(e)$ being the set of extracted labeled path nodes*[7] *such that $(H, \rho) \in \chi_K$:*

$$\forall e \in \{(n, m, \mathcal{Z}) \in \mathcal{E}' \mid n \in H, m \in \mathcal{V}' \text{ or } m \in H, n \in \mathcal{V}'\} : \Pi \models e$$

as in (3.3).

Proof. Let $K \in \mathcal{S}$ be an FG. We know with Lemma 3.19 that a path H exists. Hence, for all path nodes $n \in H$ it holds: $n \in \mathcal{V}'$. Subsequently, all edges $e = (n, m, \mathcal{Z}) \in \mathcal{E}'$ corresponding to node $n \in H$ and to node $m \in \mathcal{V}'$ are active and must hold under Π such that $\Pi \models e$. This is ensured by Lemma 3.5, since Π is valid for $\mathcal{D}$ and $\mathcal{E}'$. $\square$

Theorem 3.22 states that all constraints belonging to a path H ($\{(n, m, \mathcal{Z}) \in \mathcal{E} \mid n \in H, m \in \mathcal{V}'$ or $m \in H, n \in \mathcal{V}'\}$) hold under the schedule Π. This is *the* important conclusion for valid schedules of FDPENs. Please note, that both nodes (n, m) in the given set may be contained in H as well, since $H \subseteq \mathcal{V}'$.

Example 3.23. Let $K = (R, F)$ be an FG and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, 10, \mathcal{H}, \emptyset)$ be an FDPEN for $\{K\}$ as in Example 3.16 for the set of decision nodes $\mathcal{H} = \{n_1, \ldots, n_5\}$ and let Π be a valid schedule for $\mathcal{V}' = \{n_2, n_3, n_5, n_6\}$, $\mathcal{E}' = \mathcal{E}$ under $\{K\}$. Then we can evaluate the set of respective edges of Theorem 3.22

$$\begin{aligned}\{(n, m, \mathcal{Z}) \in \mathcal{E}' \mid n \in H, m \in \mathcal{V}' \text{ or } m \in H, n \in \mathcal{V}'\} = \\ \{(n_2, n_3, [2, 2]_{10}), (n_3, n_6, [1, 5]_{10}), (n_6, n_5, [1, 3]_{10})\}\end{aligned}$$

and with Example 3.18 we know that for all constraints $e \in M$ it holds $\Pi \models e$

[7] Which can be equivalently formulated as set of flow edges.

Definition 3.24 (Flow Decision Periodic Event Scheduling Problem)**.** *Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and let $\mathcal{E}'$ be given as in Definition 3.3. Then the* Flow Decision Periodic Event Scheduling Problem (FDPESP) *is the question whether we find a set of nodes $\mathcal{V}'$ and a schedule Π for $\mathcal{V}'$ such that Π is valid for $\mathcal{D}$, $\mathcal{E}'$ under $\mathcal{S}$.*

In the following, we will show that FDPESP is *NP*-complete by showing first its hardness and then the corresponding containment in *NP*.

Lemma 3.25. *FDPESP is NP-hard.*

Proof. It has already been shown that PESP is *NP*-hard [SU89; Nac98]. One of the newer proofs is by reducing SAT to PESP [Gro12b]. Let the set of FGs $\mathcal{S} = \emptyset$ be empty and let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \emptyset, \emptyset)$ be an FDPEN for $\mathcal{S}$. The we can easily construct a PEN $\mathcal{N}$ for $\mathcal{D}$ as in Lemma 3.5. Now, the reduction from SAT to PESP can be done, since $\mathcal{D}$, and consequently $\mathcal{N}$, is arbitrary. □

Lemma 3.26. *FDPESP $\in$ NP.*

Proof. Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and let $\mathcal{E}'$ be given as in Definition 3.3. Further let $\mathcal{V}'$ be a set of nodes and Π be a schedule for $\mathcal{V}'$ such that Π is valid for $\mathcal{D}$, $\mathcal{E}'$ under $\mathcal{S}$. Then, we can easily construct a deterministic Turing machine such that it checks all conditions in polynomial time in Definition 3.17, since all Equations (3.17), (3.18), (3.19), (3.1), (3.2), (3.3) with respect to its used sets and computational effort are finite and linear, respectively. □

Theorem 3.27. *FDPESP is NP-complete.*

Proof. Follows directly by Lemma 3.25 and Lemma 3.26. □

As proposed in Section 3.1.1, the previous results can easily be applied to DPENs with the following corollary.

Corollary 3.28. *DPESP is NP-complete.*

Proof. Follows directly by Theorem 3.27 with an empty FG set such that $\mathcal{S} = \emptyset$. □

Even the given decision problem whether a valid schedule exists as in Definition 3.24 is difficult[8] to solve[9] and will be presented in the respective results section.

Nevertheless, it is not "more" difficult than PESP, which is *NP*-complete as well. This results in the question, if an efficient encoding for FDPESP exists, which will be answered in the sequel.

[8]not tractable
[9]as shown it is *NP*-complete

3.2 SAT Encoding

Section 2.3 introduced the possibility of encoding a PEN[10] into a propositional formula in CNF. This section extends this encoding by the flow graphs as in Section 3.1.2 and thus, implicitly the decision PEN as in Section 3.1.1.

Firstly, the DPESP will be encoded into SAT in Section 3.2.1. Subsequently, the structure of the flow graphs will be added in Section 3.2.2. Hence, the FDPESP will be encoded into SAT in this section.

3.2.1 Encoding Decision Periodic Event Scheduling

In Section 2.3.2 the encoding from PESP to SAT has been introduced. This encoding considered that all constraints must hold of a given PEN. However, a DPEN offers the new structure of having nodes and constraints optional with respect to hold under a valid schedule. Thus, the implying conditions of holding with respect to nodes and constraints must be encoded into the propositional formula as well.

We divide the procedure into two parts: Firstly, we introduce the encoding of decision and constraints secondly, the encoding of decision nodes. After ensuring the valid application of the encoding by the appropriate lemmas, we conclude both in a soundness and completeness theorem ensuring the successful application of the presented method.

Example 3.29 (Decision Constraint)**.** Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \emptyset, \mathcal{A})$ be a DPEN and $e \in \mathcal{E}$ be a constraint of $\mathcal{D}$. If e is not in the set of decision edges such that $e \notin \mathcal{A}$ then it must hold under any circumstance in order to get a valid schedule Π. However, if it is in $\mathcal{A}$, then it must only hold under Π if it is in the resulting set of constraints $\mathcal{E}'$ as in Definition 3.3. Thus, we can formulate this for a schedule Π as

$$e \in \mathcal{E}' \Rightarrow e \text{ holds under } \Pi.$$

Example 3.29 shows the important implication for the decision edges: if e is in the set of edges $\mathcal{E}'$, then it must hold under Π. Since it is an implication, we can semantically equivalently reform this term as

$$\neg(e \in \mathcal{E}') \vee (e \text{ holds under } \Pi). \tag{3.37}$$

In order to encode this statement as propositional formula, we introduce the new

[10]respectively the PESP

propositional variables $r_e \in \mathcal{R}$ with the semantics

$$r_e :\Leftrightarrow e \in \mathcal{E}'. \tag{3.38}$$

Hence, we can conclude the two cases for an interpretation J

1. $r_e^J = t \Rightarrow e \in \mathcal{E}'$ and

2. $r_e^J = f \Rightarrow e \notin \mathcal{E}'$.

The term "e holds under Π" in (3.37) can be reformed with the introduced function *enc_con* as in Definition 2.44. Thus, we can reformulate (3.37) with (3.38) as

$$\neg r_e \vee \mathit{enc\_con}(e). \tag{3.39}$$

Since $\mathit{enc\_con}(e)$ is a conjunction of clauses[11], the disjunction in (3.39) is not a propositional formula in CNF. Transforming this formula into CNF results in an encoding for decision constraints such that

$$\begin{aligned} \mathit{enc\_con\_dec} &: \mathcal{E} \to \mathcal{L}(\mathcal{R}) \\ e &\mapsto \begin{cases} \mathit{enc\_con}(e) & e \notin \mathcal{A} \\ \bigwedge_{A \in \zeta(c)} (\neg r_e \vee \mathit{enc\_rec}(A)) & e \in \mathcal{A} \end{cases} \end{aligned} \tag{3.40}$$

This mapping simply extends Definition 2.44 by adding disjunctively to each encoded rectangle the negated propositional variable r_e if it is a decision constraint.

Because in general the set of decision nodes $\mathcal{H}$ is not the empty set, we have to consider those nodes as well in order to get a valid schedule as in Definition 3.3. This will be shown in the following example.

Example 3.30 (Decision Node)**.** Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN and $n, m \in \mathcal{V}$ be nodes in $\mathcal{D}$ and let $e = (n, m, \mathcal{Z}_e) \in \mathcal{E}$ be a constraint. As described in Example 3.29, we have to consider Definition 3.3. If $n \notin \mathcal{H}$ and $m \notin \mathcal{H}$, then e must hold for any schedule Π in order to be valid. If $n \in \mathcal{H}$ and $m \notin \mathcal{H}$, it must hold

$$n \in \mathcal{V}' \Rightarrow \Pi(m) - \Pi(n) \in \mathcal{Z}_e$$

[11] it is in CNF

as in (3.3). The general case can be concluded as

$$n \in \mathcal{V}', m \in \mathcal{V}' \Rightarrow \Pi(m) - \Pi(n) \in \mathcal{Z}_e.$$

As shown in (3.37), we can reformulate (3.3) as

$$\begin{aligned} &\neg(n \in \mathcal{V}' \wedge m \in \mathcal{V}') \vee (\Pi(m) - \Pi(n) \in \mathcal{Z}_e) \\ \equiv\ &\neg(n \in \mathcal{V}') \vee \neg(m \in \mathcal{V}') \vee (\Pi(m) - \Pi(n) \in \mathcal{Z}_e) \end{aligned} \tag{3.41}$$

Again, we introduce the following propositional variable $s_n \in \mathcal{R}$ with the semantically meaning

$$s_n :\Leftrightarrow n \in \mathcal{V}'. \tag{3.42}$$

Subsequently, we again handle the two cases with an interpretation J such that

1. $s_n^J = t \Rightarrow n \in \mathcal{V}'$
2. $s_n^J = f \Rightarrow n \notin \mathcal{V}'$

Inserting the introduced propositional variables in (3.41) and replacing the right term with the function *enc_con_dec* results in

$$\neg s_n \vee \neg s_m \vee enc\_con\_dec(e).$$

Hence, we are able to encode both the decision nodes and decision edges with the following definition with the same intention as in (3.40).

Definition 3.31 (Encoding Decision Node and Constraint)**.** *Let $e = (i, j, \mathcal{Z}_e) \in \mathcal{E}$ be a constraint, $\mathcal{A} \subseteq \mathcal{E}$ be a set of constraints and $\mathcal{H} \subseteq \mathcal{V}$ be a set of nodes. Further, let $S = \{r_e \in \mathcal{R} \mid e \in \mathcal{A}\} \cup \{s_n \in \mathcal{R} \mid n \in \{i, j\} : n \in \mathcal{H}\}$ be the set of propositional decision variables. Then*

$$\begin{aligned} enc\_dec : \mathcal{E} &\to \mathcal{L}(\mathcal{R}) \\ e &\mapsto \bigwedge_{A \in \zeta(c)} \Big(\bigvee_{p \in S} (\neg p) \vee enc\_rec(A) \Big) \end{aligned}$$

is the decision order encoding mapping *of the constraint e with respect to nodes i, j.*

Without a proof, it can be easily shown that *enc_dec* maps to a propositional formula in CNF, since *enc_rec* is a disjunction of literals.

The order encoding of the nodes will be as in (2.6), which adds up to the encoding of all nodes $\Omega_{\mathcal{N}}$ as in (2.7). Here, no additional decision variables have to be added, because the encoding of the potentials – respectively the events – have to hold under any circumstances. Combining this information with Definition 3.31 results in the whole encoding of DPENs with $\mathcal{P}(\mathcal{D}) = \{\mathcal{D} \mid \mathcal{D} \text{ is a DPEN}\}$ being the set of all possible DPENs.

Definition 3.32 (SAT Encoding DPEN). *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN and let be $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ the respective PEN of $\mathcal{D}$. Then*

$$\begin{aligned} \mathit{enc\_dpen} : \mathcal{P}(\mathcal{D}) &\to \mathcal{L}(\mathcal{R}) \\ \mathcal{D} &\mapsto \Omega_{\mathcal{N}} \wedge \bigwedge_{e \in \mathcal{E}} \mathit{enc\_dec}(e) \end{aligned}$$

is the DPESP order encoding (SAT) *of $\mathcal{D}$ with $\Omega_{\mathcal{N}} = \bigwedge_{n \in \mathcal{V}} enc(n)$ as in* (2.7).

The mapping in Definition 3.32 is quite similar to the encoding as in (2.14). However, each constraint is encoded by the function *enc_dec*. Thus, we have to show the soundness and completeness separately.

Before showing in Theorem 3.34 the soundness and completeness of the encoding, we show in the following lemma that an encoded constraint is valid with respect to its set of decision constraints and set of decision nodes.

Lemma 3.33 (Valid Constraint). *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN, $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes, $\mathcal{E}' \subseteq \mathcal{E}$ be a set of constraints and $e = (i, j, \mathcal{Z}_e) \in \mathcal{E}$ be a constraint. Further let $\mathcal{F} = enc(i) \wedge enc(j) \wedge \mathit{enc\_dec}(e)$ be an encoded constraint with its respective node encoding, $S = \{r_e \in \mathcal{R} \mid e \in \mathcal{A}\} \cup \{s_n \in \mathcal{R} \mid n \in \{i, j\} : n \in \mathcal{H}\}$ as in Definition 3.31 and J an interpretation. Then it holds*

$$\begin{aligned} &(i) \;\; (\exists p \in S : p^J = f) \Rightarrow J \models \mathit{enc\_dec}(e), \\ &(ii) \;\; (\forall p \in S : p^J = t) \Rightarrow (J \models \mathcal{F} \Leftrightarrow \exists \Pi : \Pi \models \mathcal{Z}_e) \end{aligned}$$

with Π being a schedule for $\{i, j\}$.

Proof. Firstly, we know that (i):

$$\mathit{enc\_dec}(e) \overset{\text{Def 3.31}}{=} \bigwedge_{A \in \zeta(c)} \Big(\bigvee_{p \in S} (\neg p) \vee \mathit{enc\_rec}(A) \Big)$$

$$\overset{\text{De Morgan}}{\equiv} \big(\bigvee_{p\in S} \neg p\big) \vee \bigwedge_{A\in\zeta(c)} enc\_rec(A) \tag{3.43}$$

If it holds

$$\exists p \in S : p^J = f,$$

then it follows

$$\exists p \in S : (\neg p)^J = t$$

and with Definition 2.11

$$J \models \bigvee_{p\in S} \neg p.$$

Since (3.43) is a disjunction and the left term $(\bigvee_{p\in S} \neg p)$ is true under J, it follows

$$J \models \big(\bigvee_{p\in S} \neg p\big) \vee \bigwedge_{A\in\zeta(c)} enc\_rec(A).$$

Hence, we can conclude with semantic equivalence in (3.43)

$$J \models enc\_dec(e).$$

(ii):

$$\begin{aligned}
\mathcal{F} &= enc(i) \wedge enc(j) \wedge enc\_dec(e)\\
&= enc(i) \wedge enc(j) \wedge \bigwedge_{A\in\zeta(c)} \Big(\bigvee_{p\in S} (\neg p) \vee enc\_rec(A)\Big)\\
&\overset{\text{De Morgan}}{\equiv} enc(i) \wedge enc(j) \wedge \Big(\big(\bigvee_{p\in S} \neg p\big) \vee \bigwedge_{A\in\zeta(c)} enc\_rec(A)\Big)
\end{aligned} \tag{3.44}$$

If it holds

$$\forall p \in S : p^J = t,$$

then it follows

$$\forall p \in S : (\neg p)^J = f. \tag{3.45}$$

With Corollary 2.16 we can conclude for the disjunction in (3.44) and (3.45) that

$$\Big(\big(\bigvee_{p\in S} \neg p\big) \vee \bigwedge_{A\in\zeta(c)} enc\_rec(A)\Big)^J = \Big(\bigwedge_{A\in\zeta(c)} enc\_rec(A)\Big)^J \tag{3.46}$$

and thus,

$$\mathcal{F}^J = \Big(enc(i) \wedge enc(j) \wedge \bigwedge_{A \in \zeta(c)} enc\_rec(A)\Big)^J$$

Finally, we can conclude for (3.46) with Definition 2.44

$$\mathcal{F}^J = \Big(enc(i) \wedge enc(j) \wedge enc\_con(e)\Big)^J$$

and thus, with Theorem 2.47 for the single edge e

$$J \models \mathcal{F} \Leftrightarrow \exists \Pi : \Pi \models \mathcal{Z}_e.$$

□

The previous lemma is the equivalence to Lemma 3.5 and it ensures in (i) that a constraint holds in the SAT encoding if any decision variable is false, and in (ii) that if all decision variables are true, the constraint holds only for a valid schedule which corresponds to Definition 3.3. This results in the following soundness and completeness theorem.

Theorem 3.34 (Soundness Completeness DPESP Encoding). *Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be a DPEN, $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes and let $\mathcal{E}' \subseteq \mathcal{E}$ be a set of constraints of $\mathcal{D}$. Further let*

$$\mathcal{F} := enc\_dpen(\mathcal{D}) \in \mathcal{L}(\mathcal{R})$$

be the encoded propositional formula of $\mathcal{D}$. Then

$$\exists J : J \models \mathcal{F} \Leftrightarrow \exists \Pi : \Pi \textit{ is valid for } \mathcal{D} \textit{ and } \mathcal{E}'$$

with J being an interpretation and Π a schedule of $\mathcal{V}'$.

Proof. As in proof of Theorem 2.47 by extending the constraint encoding equivalence with Lemma 3.33, since this lemma can be applied to all constraints of $\mathcal{D}$ and their respective SAT encoding. □

3.2.2 Encoding Flow Decision Periodic Event Scheduling

The previous section introduced the encoding of DPENs into SAT. Likewise, this section covers the corresponding encoding by extending it for FGs. The core encoding of DPENs will stay equivalently and will soundly connected to the FG structure.

The additional encoding must suffice the three additional requests in Definition 3.17. The appropriate encoding will be done by at-least-one and at-most-one constraints (clauses and clause sets). After presenting the whole encoding, the section will be concluded in a soundness and completeness theorem.

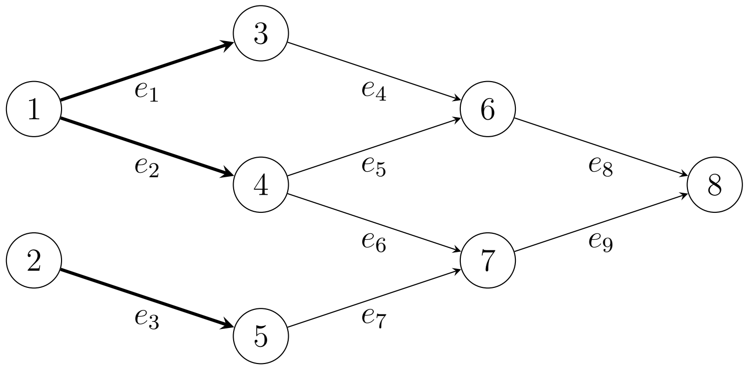


Figure 3.8: Exemplifying flow network with highlighted set of source edges.

Example 3.35 (Source Nodes)**.** Let $K = (R, F)$ be the FG as in Figure 3.8 with the set of source nodes $S_K = \{1, 2\}$ for K and l be a label function as in (3.16) such that

$$l(e_1) = n,\ l(e_2) = m,\ l(e_3) = o$$

with $\{n, m, o\} \subseteq \mathcal{V}$ being periodic events. Then we know with Definition 3.31 that $A := \{s_n, s_m, s_o\}$ are the corresponding propositional decision variables for all labeled flow nodes. In order to get a valid schedule, the encoding must suffice (3.17), which states that it must be exactly one source edge for K in order to have a valid path. In propositional logic this can be encoded by stating

1. at least one variable in A must be true and

2. at most one variable in A must be true.

The first statement can simply be achieved by a single clause with all variables such that

$$c_1 = s_n \vee s_m \vee s_o.$$

Pairwise excluding that two variables are true results in the second statement (at most one) such that

$$\begin{aligned} c_2 &= \neg s_n \lor \neg s_m \\ c_3 &= \neg s_n \lor \neg s_o \\ c_4 &= \neg s_m \lor \neg s_o. \end{aligned}$$

Connecting all clauses conjunctively in $\mathcal{F}$ results in the exactly-one encoding such that

$$\mathcal{F} = c_1 \land c_2 \land c_3 \land c_4.$$

One of the possible interpretations that satisfies $\mathcal{F}$ is J with

$$s_n^J = f, s_m^J = t, s_o^J = f,$$

since this satisfies all clauses $c_1, \ldots, c_4$.
A counter example would be that non of the decision variables are true such that

$$s_n^J = f, s_m^J = f, s_o^J = f,$$

Then, $\mathcal{F}$ is not satisfied since c_1 is not satisfied, because it is not at least one variable true under J.
Another counter example would be if more than one variable is true, for example

$$s_n^J = t, s_m^J = t, s_o^J = f,$$

Now, c_1 is satisfied. However, c_2 is not satisfied, because both $J \models s_n$ and $J \models s_m$.

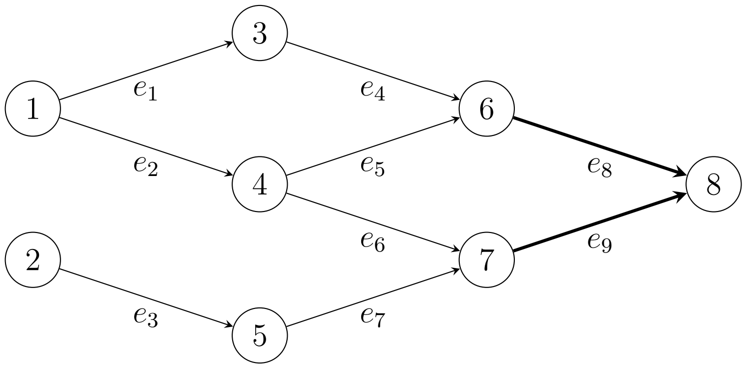


Figure 3.9: Exemplifying flow network with highlighted set of destination edges.

The previous example shows the basic idea of encoding that exactly one source node for an FG must be in the resulting node set $\mathcal{V}'$. Likewise, this can be applied to the set of destination edges with the same technique as in Example 3.35 which is visualized in Figure 3.9.

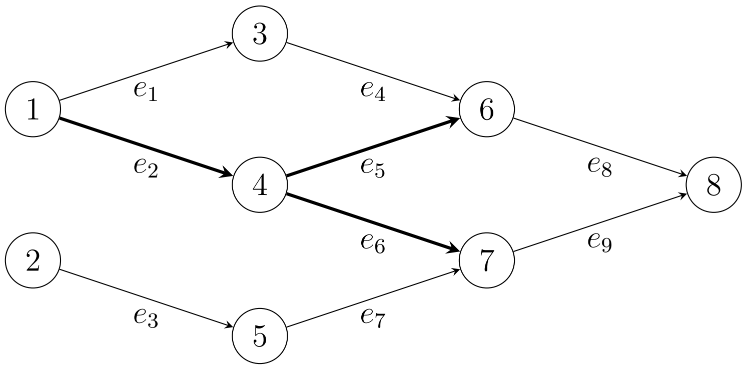


Figure 3.10: Exemplifying flow network as in Example 3.36 which highlights the needed edges for the flow conservation of node flow 4.

Example 3.36 (Flow Conservation)**.** Let $K = (R, F)$ be an FG which can be visualized be seen in Figure 3.10. Further let l be a label function as in (3.16) such that

$$l(e_2) = m,\ l(e_5) = r,\ l(e_6) = s$$

with $\{n, r, s\} \subseteq \mathcal{V}$ being periodic events.

In Definition 3.17 the flow conservation is stated in (3.19) which can be translated as: If exactly one incoming flow edge (labeled node) is active, then exactly one outgoing flow edge (labeled node) must be active. Here we only regard the flow node 4. We know that it must hold for this node, since $4 \in R = \{1, \dots, 8\}$ and $4 \notin S_K \cup D_K = \{1, 2, 8\}$. Since we have proved in Theorem 3.21 that we can define an alternative schedule but have the same results with respect to flow conservation, it is sufficient that (3.29), (3.30) and (3.31) hold for the flow node 4 such that

$$|\bigcup_{(i,4)\in I_K(4)} l(i,4) \cap \mathcal{V}'| \leq 1, \tag{3.47}$$

$$|\bigcup_{(4,j)\in O_K(4)} l(4,j) \cap \mathcal{V}'| \leq 1, \tag{3.48}$$

$$\forall (i,j) \in F, j \notin D_K : \Big(l(i,j) \in \mathcal{V}' \Rightarrow \exists k \in \mathcal{V}' \cap \mathit{range}(l) : l^{-1}(k) \in O_K(j)\Big) \tag{3.49}$$

holds. Substituting $I_K(4) = \{e_2\}$ and $O_K(4) = \{e_5, e_6\}$ we can conclude that (3.47) and (3.48) holds if and only if

$$\begin{aligned}
&|\bigcup_{(i,4)\in\{e_2\}} l(i,4)\cap\mathcal{V}'| \leq 1,\\
&|\bigcup_{(4,j)\in\{e_5,e_6\}} l(4,j)\cap\mathcal{V}'| \leq 1\\
\Leftrightarrow\ &|\{l(e_2)\}\cap\mathcal{V}'| \leq 1,\\
&|\{l(e_5),l(e_6)\}\cap\mathcal{V}'| \leq 1\\
\Leftrightarrow\ &|\{m\}\cap\mathcal{V}'| \leq 1, \qquad (3.50)\\
&|\{r,s\}\cap\mathcal{V}'| \leq 1 \qquad (3.51)
\end{aligned}$$

As introduced in (3.42) we have the propositional decision variables $\{s_m, s_r, s_s\}$ which are only true under an interpretation if and only if the corresponding nodes are in the set of nodes $\mathcal{V}'$. The Equations (3.50) and (3.51) are at-most-one constraints which can be achieved by pairwise excluding that two decision variables are true such that for an interpretation J it holds

$$\begin{aligned}
(3.50) &\Leftrightarrow J \models \emptyset, &(3.52)\\
(3.51) &\Leftrightarrow J \models (\neg s_r \vee \neg s_s). &(3.53)
\end{aligned}$$

In (3.52) we see that it holds under any interpretation since only one flow edge goes into the flow node 4 and thus, it is always less or equal than one and in (3.53) we can deduce that at most one of the decision variables in $\{s_r, s_s\}$ can be true under the interpretation J.
Regarding the last condition and substituting the node j for node 4 in (3.49) we get

$$\forall(i,4)\in F, 4\notin D_K : \Big(l(i,4)\in\mathcal{V}' \Rightarrow \exists k\in\mathcal{V}'\cap \mathit{range}(l) : l^{-1}(k)\in O_K(4)\Big)$$

and thus we only have to regard the flow edge e_2 such that

$$\begin{aligned}
&l(e_2)\in\mathcal{V}' \Rightarrow \exists k\in\mathcal{V}'\cap \mathit{range}(l) : l^{-1}(k)\in O_K(4)\\
\Leftrightarrow\ &m\in\mathcal{V}' \Rightarrow \exists k\in\mathcal{V}'\cap \mathit{range}(l) : l^{-1}(k)\in \{e_5,e_6\}\\
\overset{l \text{ is bijective}}{\Leftrightarrow}\ &m\in\mathcal{V}' \Rightarrow \exists k\in\mathcal{V}'\cap \mathit{range}(l) : k\in \{l(e_5),l(e_6)\}\\
\Leftrightarrow\ &m\in\mathcal{V}' \Rightarrow \exists k\in\mathcal{V}'\cap \mathit{range}(l) : k\in \{r,s\}\\
\Leftrightarrow\ &m\in\mathcal{V}' \Rightarrow |\mathcal{V}'\cap \mathit{range}(l)\cap\{r,s\}| \geq 1 \qquad (3.54)
\end{aligned}$$

Hence, at least one labeled node must be in the resulting set of decision nodes $\mathcal{V}'$, if a preceding labeled node is in $\mathcal{V}'$. In other words, if a flow edge goes into a flow node, then at least one flow edge must go out of the flow node. Encoding this circumstance for this example as propositional formula, we get for an interpretation J.

$$\begin{aligned}(3.54) &\Leftrightarrow (J \models s_m) \Rightarrow (J \models s_r) \text{ or } (J \models s_s)\\ &\Leftrightarrow J \models (s_m \Rightarrow (s_r \vee s_s))\\ &\Leftrightarrow J \models (\neg s_m \vee s_r \vee s_s) \end{aligned} \tag{3.55}$$

Finally, we can conclude that all conditions for a valid schedule (3.29), (3.30) and (3.31) hold if and only if all propositional formulas (3.52), (3.53) and (3.55) hold such that we have the formula

$$\mathcal{F} = (\neg s_r \vee \neg s_s) \wedge (\neg s_m \vee s_r \vee s_s)$$

For example, if $s_m^J = t$, $s_r^J = f$ and $s_s^J = t$, then we know that $\mathcal{F}$ is satisfied under J, which ensures that only e_2 and e_5 are in the resulting (sub) path.

The previous rather long, extended example shows the idea of encoding the alternative conditions for a valid schedule as at-least-one and at-most-one propositional formulas. Both Example 3.35 and Example 3.36 result in the encoding for FGs in order to extract paths as in the following definition.

Definition 3.37 (Encoding FG)**.** *Let $K = (R, F)$ be an FG, and l be a label function into the set of nodes $\mathcal{H}$. Further, let $U = \{s_n \in \mathcal{R} \mid n \in \mathcal{H} \cap range(l)\}$ be the set of propositional decision variables for K. Then*

$$\begin{aligned} enc\_fg_l : \mathcal{S} &\to \mathcal{L}(\mathcal{R})\\ K \mapsto\ & \Big(\bigvee_{n \in S_K} \bigvee_{e \in O(n)} s_{l(e)} \Big) && (3.56)\\ &\wedge \Big(\bigwedge_{n \in S_K} \bigwedge_{e \in O(n)} \bigwedge_{m \in S_K} \bigwedge_{e' \in O(m), e \neq e'} (\neg s_{l(e)} \vee \neg s_{l(e')}) \Big) && (3.57)\\ &\wedge \Big(\bigvee_{n \in D_K} \bigvee_{e \in I(n)} s_{l(e)} \Big) && (3.58)\\ &\wedge \Big(\bigwedge_{n \in D_K} \bigwedge_{e \in I(n)} \bigwedge_{m \in D_K} \bigwedge_{e' \in I(m), e \neq e'} (\neg s_{l(e)} \vee \neg s_{l(e')}) \Big) && (3.59)\\ &\wedge \Big(\bigwedge_{n \in R \setminus (S_K \cup D_K)} \bigwedge_{e \in I(n)} \bigwedge_{e' \in I(n), e \neq e'} (\neg s_{l(e)} \vee \neg s_{l(e')}) \Big) && (3.60) \end{aligned}$$

$$\wedge \Big(\bigwedge_{n \in R \setminus (S_K \cup D_K)} \bigwedge_{e \in O(n)} \bigwedge_{e' \in O(n), e \neq e'} (\neg s_{l(e)} \vee \neg s_{l(e')}) \Big) \tag{3.61}$$

$$\wedge \Big(\bigwedge_{(n,m) \in F, m \notin D_K} \big(\neg s_{l(n,m)} \bigvee_{e \in O_K(m)} s_{l(e)}\big) \Big) \tag{3.62}$$

is the FG encoding *for K with respect to U and $\mathcal{S}$ being the set of FGs.*

The function *enc_fg$_l$* maps to a propositional in CNF, since all Equations (3.56)–(3.62) are connected conjunctively and all equations are either disjunctions of literals or conjunctions of disjunctions or literals, respectively. Hence, each mapped FG under *enc_fg$_l$* can be connected conjunctively and results in a propositional formula in CNF, which is needed for state-of-the-art SAT solvers [Bie+09].

Comprehensively describing Definition 3.37 we could align the respective equations as follows

$$\left.\begin{array}{l}(3.56) \rightarrow \text{at-least-one source} \\ (3.57) \rightarrow \text{at-most-one source}\end{array}\right\} \text{exactly one source,}$$

$$\left.\begin{array}{l}(3.58) \rightarrow \text{at-least-one destination} \\ (3.59) \rightarrow \text{at-most-one destination}\end{array}\right\} \text{exactly one destination,}$$

$$(3.60) \rightarrow \text{at-most-one incoming flow edge per flow node,}$$

$$(3.61) \rightarrow \text{at-most-one outgoing flow edge per flow node,}$$

$$(3.62) \rightarrow \text{successor flow edge active if incoming active.}$$

In other words, (3.56) and (3.57) ensure that exactly one flow edge (its labeled node) is active that is an outgoing edge of the set of source nodes S_K. Likewise, (3.58) and (3.59) state that exactly one destination flow edge is active, which means that its respective labeled node is in $\mathcal{V}'$. Equation (3.60) and (3.61) suffice the condition that at-most-one incoming and outgoing flow edge of a flow node is active, respectively. Finally, (3.62) ensures the condition in (3.31) such that if a preceding flow edge (its labeled node) is active, then at-least-one succeeding flow edge must be active. Please note, that here we present only the most basic at-most-one encoding[12] but it could be simply switched out by more advanced at-most-one[13] encodings like the binary encoding [Fri+05], the product encoding [Che10], the commander encoding [KK07], etc.

[12] i. e., the pairwise encoding
[13] or in general at-most-k

Example 3.38 (Encode FG)**.** Let $K = (R, F)$ be an FG as in Example 3.36 and l be a label function for the set of periodic nodes $\mathcal{V} = \{n_1, \ldots, n_9\}$ such that $l(e_i) = n_i$ $(i \in \{1, \ldots, 9\})$. Then, we can encode K with *enc_fg*$_l$ such that

$$\begin{aligned}
enc\_fg_l(K) = &(s_{n_1} \vee s_{n_2} \vee s_{n_3})\\
&\wedge (\neg s_{n_1} \vee \neg s_{n_2}) \wedge (\neg s_{n_1} \vee \neg s_{n_3}) \wedge (\neg s_{n_2} \vee \neg s_{n_3})\\
&\wedge (s_{n_8} \vee s_{n_9})\\
&\wedge (\neg s_{n_8} \vee \neg s_{n_9})\\
&\wedge (\neg s_{n_4} \vee \neg s_{n_5}) \wedge (\neg s_{n_6} \vee \neg s_{n_7}) \wedge (\neg s_{n_8} \vee \neg s_{n_9})\\
&\wedge (\neg s_{n_5} \vee \neg s_{n_6})\\
&\wedge \big((\neg s_{n_1} \vee s_{n_4}) \wedge (\neg s_{n_2} \vee s_{n_5} \vee s_{n_6}) \wedge (\neg s_{n_3} \vee s_{n_7})\\
&\wedge (\neg s_{n_4} \vee s_{n_8}) \wedge (\neg s_{n_5} \vee s_{n_8}) \wedge (\neg s_{n_6} \vee s_{n_9}) \wedge (\neg s_{n_7} \vee s_{n_9})\big).
\end{aligned}$$

Encoding further FGs will result in connecting them conjunctively to the resulting formula.

Inducing that each interpretation J for an encoded FG results in a valid path as in Definition 3.10 if and only if J is a model[14] is given by the following lemma.

Lemma 3.39 (Extract Path from Encoding)**.** *Let $K = (R, F)$ be an FG, and l be a label function into the set of nodes $\mathcal{H}$. Further, let $U = \{s_n \in \mathcal{R} \mid n \in \mathcal{H} \cap range(l)\}$ be the set of propositional decision variables for K. Then it holds with $\mathcal{F} := enc\_fg_l(K)$ for an interpretation J and $H^J = \{l^{-1}(n) \in F \mid n \in \mathcal{H} : s_n^J = t\}$ being the set of satisfied labeled flow edges:*

$$J \models \mathcal{F} \Leftrightarrow (H^J, \rho) \in \chi_K$$

for a path ρ and χ_K being the relation as in (3.9)*.*

Proof. "$\Leftarrow$": Let ρ be a path for K such that $(H^J, \rho) \in \chi_K$. Then, we can construct the interpretation J as in the preconditions such that

$$J = \{s_n \mapsto s_n^J \mid n \in \mathcal{H}\} \text{ with } s_n^J = \begin{cases} t, & l^{-1}(n) \in H^J \\ f, & l^{-1}(n) \notin H^J \end{cases} \tag{3.63}$$

We need to show that all Equations (3.56)–(3.62) hold, because they are connected conjunctively.

[14]Which means that it satisfies the propositional formula.

With Lemma 3.13 we know that it holds

$$\exists!(n,m) \in H^J : n \in S_K, n_1 := n, \tag{3.64}$$

$$\exists!(n,m) \in H^J : m \in D_K, n_{k+1} := m, \tag{3.65}$$

$$\forall i \in \{2,\dots,k\} \exists!(n_{i-1},n) \in H^J : (n_{i-1},n) \in F, n_i := n \tag{3.66}$$

for the path $\rho = ((n_1,n_2),\dots,(n_k,n_{k+1}))$ such that $(H^J,\rho) \in \chi_K$. Consequently,

$$\begin{aligned}(3.64) &\overset{(3.63)}{\Rightarrow} J \models s_{l(n,m)} \overset{\text{Disjunction}}{\Rightarrow} (3.56) \text{ holds,}\\ (3.64) &\overset{(3.63)}{\Rightarrow} \exists! s_{l(n,m)} \in \mathcal{R} : J \models s_{l(n,m)} \Rightarrow (3.57) \text{ holds,}\\ (3.65) &\overset{(3.63)}{\Rightarrow} J \models s_{l(n,m)} \overset{\text{Disjunction}}{\Rightarrow} (3.58) \text{ holds,}\\ (3.65) &\overset{(3.63)}{\Rightarrow} \exists! s_{l(n,m)} \in \mathcal{R} : J \models s_{l(n,m)} \Rightarrow (3.59) \text{ holds.}\end{aligned}$$

With (3.66) (it exists *exactly one* flow edge) we know that for all $n \in R \setminus (S_K \cup D_K)$

$$|\{e \in F \mid e \in H^J, e \in I_K(n)\}| = |\{e \in F \mid e \in H^J, e \in O_K(n)\}| \leq 1. \tag{3.67}$$

Subsequently, for all $n \in R \setminus (S_K \cup D_K)$

$$\begin{aligned}(3.67) &\overset{(3.63)}{\Rightarrow} |\{s_{l(e)} \in \mathcal{R} \mid e \in I_K(n), J \models s_{l(e)}\}| \leq 1 \Rightarrow (3.60) \text{ holds,}\\ (3.67) &\overset{(3.63)}{\Rightarrow} |\{s_{l(e)} \in \mathcal{R} \mid e \in O_K(n), J \models s_{l(e)}\}| \leq 1 \Rightarrow (3.61) \text{ holds.}\end{aligned}$$

With (3.66) we can conduct for all $m \in R, m \notin D_K$ and all edges $(n,m) \in F$

$$(n,m) \in H^J \Rightarrow \exists (m,o) \in F : (m,o) \in H^J \tag{3.68}$$

and thus,

$$(3.68) \overset{(3.63)}{\Rightarrow} \exists (m,o) \in F : J \models s_{l(m,o)}, \text{ if } J \models s_{l(n,m)} \overset{\text{Ex 2.15}}{\Rightarrow} (3.62) \text{ holds.}$$

Finally, since all Equations (3.56)–(3.62) hold for the constructed interpretation J and they are connected conjunctively:

$$J \models \mathcal{F}.$$

"$\Rightarrow$": Let $J \models \mathcal{F}$ and thus, we can extract $H^J = \{l^{-1}(n) \in F \mid n \in \mathcal{H} : s_n^J = t\}$ as in the precondition. Since $\mathcal{F}$ is in CNF, J models all Equations (3.56)–(3.62). Now, we construct an FDPEN $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ for the set of FGs $\{K\}$ with $\mathcal{E} = \mathcal{A} = \emptyset$ and $\mathcal{V} = \mathcal{H}$ as in the precondition. Then, we can construct a schedule Π into the set of nodes $\mathcal{V}'$ such that

$$\mathcal{V}' = \{n \mid n \in \mathcal{H}, l^{-1}(n) \in H^J\} \tag{3.69}$$

with $\Pi(n) = 0$ for all $n \in \mathcal{V}'$. Then, since $\mathcal{E}'$ is empty ($\mathcal{A} = \emptyset$), we know with Lemma 3.5 that Π is valid for $\mathcal{D}$ and $\mathcal{E}' = \emptyset$. Subsequently, with Theorem 3.21 we know that it is sufficient to show that $\mathcal{V}'$ suffices the path conditions for valid schedules in (3.17), (3.18) (source and destination conditions) and (3.29), (3.30), (3.31) (alternative flow conditions) with $\mathcal{V}'$ being the extracted labeled nodes given as in the interpretation J in (3.69).
For (3.17) we need to show

$$A = \{j \in R \mid n \in \mathcal{V}', (j,k) = l^{-1}(n) \in F, j \in S_K\},\ |A| = 1.$$

Hence, it is sufficient to show

$$(i) : |A| \geq 1,$$
$$(ii) : |A| \leq 1.$$

With (3.69) and the definition of H^J we can conclude

$$|A| = |\{s_n \mid n \in \mathcal{H}, l^{-1}(n) = (m,o) \in F, m \in S_K : s_n^J = t\}| \tag{3.70}$$

(i) holds, since (3.56) is satisfied under J and thus, at least one propositional decision variable as in (3.70) is true under J Consequently, $|A| \geq 1$.
(ii) holds, since (3.57) is satisfied under J and thus, at most one propositional decision variable is true, since all variables are pairwise excluded to be true, which could be indirectly be proved. Thus, $|A| \leq 1$ and hence, (3.17) holds.
Likewise, we can do this for the destination condition in (3.18) with (3.58) and (3.59).
It remains to show the alternative flow conditions in (3.29), (3.30) and (3.31). First, we show that the cardinality of the set of incoming and outgoing flow edges for each flow node is less or equal than one as in (3.29) and (3.30), respectively. Again, we show that both sets are isomorphic and thus, have the same cardinality.
Since (3.60) is satisfied under J it can easily be indirectly proved, that at most one propositional decision variable is true in the corresponding clauses, since all variables are

pairwise excluded to be true such that for all $n \in R \setminus (S_K \cup D_K)$ we can conclude that

$$\begin{aligned}
& |\{s_{l(e)} \in \mathcal{R} \mid e \in I_K(n) : J \models s_{l(e)}\}| \leq 1 \\
\Leftrightarrow \quad & |\{l(e) \in \mathcal{H} \mid e \in I_K(n) : J \models s_{l(e)}\}| \leq 1 \\
\Leftrightarrow \quad & |\{l(e) \in \mathcal{H} \mid e \in I_K(n)\} \cap \{n \mid n \in \mathcal{H}, J \models s_n\}| \leq 1 \\
\Leftrightarrow \quad & |\textstyle\bigcup_{e \in I_K(n)} l(e) \cap \{n \mid n \in \mathcal{H}, J \models s_n\}| \leq 1
\end{aligned}$$

which is equivalent to $|\bigcup_{e \in I_K(n)} l(e) \cap \mathcal{V}'| \leq 1$ as in (3.29) if and only if

$$\begin{aligned}
& \{n \mid n \in \mathcal{H}, J \models s_n\} = \mathcal{V}' \\
\overset{(3.69)}{\Leftrightarrow} \quad & \{n \mid n \in \mathcal{H}, J \models s_n\} = \{n \mid n \in \mathcal{H}, l^{-1}(n) \in H^J\} \\
\Leftrightarrow \quad & \forall n \in \mathcal{H} : (J \models s_n \Leftrightarrow l^{-1}(n) \in H^J) \\
\overset{\text{Def } H^J}{\Leftrightarrow} \quad & \forall n \in \mathcal{H} : (J \models s_n \Leftrightarrow l^{-1}(n) \in \{l^{-1}(n) \in F \mid n \in \mathcal{H} : s_n^J = t\}) \\
\Leftrightarrow \quad & \forall n \in \mathcal{H} : (J \models s_n \Leftrightarrow l^{-1}(n) \in \{l^{-1}(n) \in F \mid n \in \mathcal{H} : J \models s_n\}) \\
\overset{l \text{ is bijective}}{\Leftrightarrow} \quad & \forall n \in \mathcal{H} : (J \models s_n \Leftrightarrow n \in \{n \in R \mid n \in \mathcal{H} : J \models s_n\}) \\
\overset{\forall n \in \mathcal{H}}{\Leftrightarrow} \quad & \forall n \in \mathcal{H} : (J \models s_n \Leftrightarrow J \models s_n),
\end{aligned}$$

which is obviously a tautology and thus, (3.29) holds. Likewise, this can be shown for the outgoing flow edges in (3.30) with (3.61).
It remains to show that the predecessor-successor relation in (3.31) holds. We know that (3.62) is satisfied under J and therefore, it holds

$$\begin{aligned}
& J \models \Big(\textstyle\bigwedge_{(n,m) \in F, m \notin D_K} \big(\neg s_{l(n,m)} \bigvee_{e \in O_K(m)} s_{l(e)} \big) \Big) \\
\overset{\text{Conjunction}}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : J \models \big(\neg s_{l(n,m)} \textstyle\bigvee_{e \in O_K(m)} s_{l(e)} \big) \\
\overset{\text{Ex 2.15}}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : J \models \big(s_{l(n,m)} \Rightarrow (\textstyle\bigvee_{e \in O_K(m)} s_{l(e)}) \big) \\
\overset{\text{Def } H^J}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : \big((n,m) \in H^J \Rightarrow J \models (\textstyle\bigvee_{e \in O_K(m)} s_{l(e)}) \big) \\
\overset{\text{Disjunction}}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : \big((n,m) \in H^J \Rightarrow \exists e \in O_K(m) : J \models s_{l(e)} \big) \\
\overset{\text{Def } H^J}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : \big((n,m) \in H^J \Rightarrow \exists e \in O_K(m) : e \in H^J \big) \\
\overset{l \text{ is bijective, } (3.69)}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : \big(l(n,m) \in \mathcal{V}' \Rightarrow \exists e \in O_K(m) : l(e) \in \mathcal{V}' \big) \\
\overset{\mathcal{V}' \subseteq \mathcal{H}, K \text{ is finite}}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : \big(l(n,m) \in \mathcal{V}' \Rightarrow \exists l(e) \in \mathcal{V}' \cap \mathcal{H} : e \in O_K(m) \big) \\
\overset{l \text{ is bijective,}}{\Rightarrow} \quad & \forall (n,m) \in F, m \notin D_K : \\
& \qquad \big(l(n,m) \in \mathcal{V}' \Rightarrow \exists o \in \mathcal{V}' \cap \mathit{range}(l) : l^{-1}(o) \in O_K(m) \big),
\end{aligned}$$

because $\mathcal{H} = \mathit{range}(l)$. This is actually the last condition such that (3.31) holds.
Since all Equations (3.17), (3.18), (3.29), (3.30), and (3.31) hold, we can finally conclude

with Lemma 3.19 that $(H^J, \rho) \in \chi_K$. □

As stated before, we can simply connect conjunctively each encoded FG to the final propositional formula. Furthermore, since we use the same propositional decision variables, we can use the encoding of DPENs in Definition 3.32 to encode the corresponding constraints of a given FDPEN, which results in the following Definition.

Definition 3.40 (Encoding FDPEN)**.** *Let $\mathcal{S}$ be a set of FGs. Further let $\mathcal{D}$ be an FDPEN with $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ for $\mathcal{S}$ and l be a label function into the set of nodes $\mathcal{H}$. Then*

$$\begin{aligned} \mathit{enc\_fdpen}_l : \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{S}) &\to \mathcal{L}(\mathcal{R}) \\ (\mathcal{D}, \mathcal{S}) &\mapsto \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K) \wedge \mathit{enc\_dpen}(\mathcal{D}) \end{aligned}$$

is the FDPESP SAT encoding *of $\mathcal{D}$ for $\mathcal{S}$.*

Subsequently, we added to the mapping *enc_dpen* simply all encoded FGs via $\mathit{enc\_fg}_l$ conjunctively which results in the function $\mathit{enc\_fdpen}_l$. Thus, the resulting formula is in CNF as well.

The following theorem ensures the soundness and completeness of the given encoding for valid schedules as in Definition 3.17.

Theorem 3.41 (Soundness Completeness FDPESP Encoding)**.** *Let $\mathcal{S}$ be a set of FGs. Further let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and l be a label function into the set of nodes $\mathcal{H}$, $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes and let $\mathcal{E}' \subseteq \mathcal{E}$ be a set of constraints of $\mathcal{D}$. Further let*

$$\mathcal{F} := \mathit{enc\_fdpen}_l(\mathcal{D}, \mathcal{S}) \in \mathcal{L}(\mathcal{R})$$

be the encoded propositional formula of $\mathcal{D}$ for $\mathcal{S}$. Then

$$\exists J : J \models \mathcal{F} \Leftrightarrow \exists \Pi : \Pi \text{ is valid for } \mathcal{D}, \mathcal{E}' \text{ under } \mathcal{S}$$

with J being an interpretation and Π a schedule of $\mathcal{V}'$.

Proof.

$$
\begin{aligned}
& \exists J : J \models \mathcal{F} \\
\overset{\text{Def 3.40}}{\Leftrightarrow}\quad & \exists J : J \models \textstyle\bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K) \wedge \mathit{enc\_dpen}(\mathcal{D}) \\
\overset{\text{Lem 3.39, Conjunction}}{\Leftrightarrow}\quad & \exists J : J \models \mathit{enc\_dpen}(\mathcal{D}),\ \forall K \in \mathcal{S} : (H^J, \rho) \in \chi_K \\
\overset{\text{Thm 3.34}}{\Leftrightarrow}\quad & \exists \Pi : \Pi \text{ is valid for } \mathcal{D} \text{ and } \mathcal{E}',\ \forall K \in \mathcal{S} : (H^J, \rho) \in \chi_K \\
\overset{\text{Def 3.17,Lem 3.19}}{\Leftrightarrow}\quad & \exists \Pi : \Pi \text{ is valid for } \mathcal{D} \text{ and } \mathcal{E}' \text{ under } \mathcal{S}
\end{aligned}
$$

with H^J as in Lemma 3.39. □

Theorem 3.41 ensures the successful application of SAT solvers to the encoded formula such that we can extract the resulting paths for each FG and the accompanying valid schedule. Computational results will be shown in Section 5. However, firstly we will display several possible applications in Section 3.3 to show the usage of the introduced FDPESP.

3.3 Applications

This section covers some of the main applications that are currently used on real-world scenarios. However, with a general model as given in Section 3.1, it can easily be applied to wide variety of new problems.

The problems covered in this work are distinct chain paths, duplicated chain paths and non-connected flow graphs in Section 3.3.1, Section 3.3.2 and Section 3.3.3, respectively [Gro+15a]. The first one manages timetabling for periodic train paths whose driving (or vehicle) dynamics and paths are not precisely given, yet, discretized. Furthermore, the second handles scheduling of rail freight trains into given public transport networks. Finally, the last deals with the station track allocation of the corresponding periodic train paths.

In the following examples, all respective times[15] are made up and need to be calculated automatically in real-world scenarios with a software like TAKT [Opi09; Küm+15].

3.3.1 Distinct Chain Paths

The integration of the trains' routing and timetabling into a singe planning step for railway networks has a high necessity in real-world applications. This is reasoned in the fact that often the train path system is known beforehand, however, the microscopic

[15] i. e. headways, driving, change

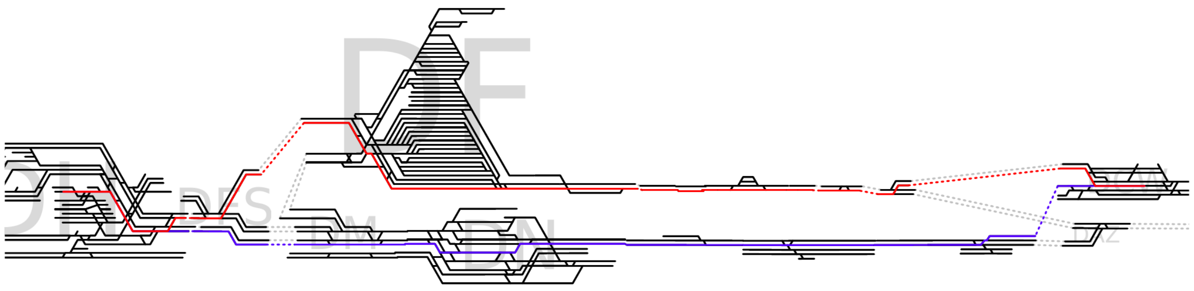


Figure 3.11: Different possible routes (red, blue) for a periodic train path between DH (Dresden Hauptbahnhof, Germany) and DCW (Coswig, Germany).

routed train paths are yet open for the planners, since only the stopping stations of the train path are already fixed. Especially in highly crowded sections that reach the capacity limit, the possibilities of alternative routes and driving (or vehicle) dynamics yield more efficient timetables, since the network capacity can be better utilized, maintaining conflict-freeness.

Consequently, the possible routes and the corresponding driving dynamics can be passed to the mathematical model. In our case, we use the FDPEN approach to achieve this goal. First of all, since the number of possible routes is exponentially high and the driving dynamics are continuous, we need to cut the search space drastically to maintain low computational time. This can be approached by applying only concrete alternatives of the corresponding operator or given a maximum detour factor that only regards all routes between two stations maintaining its maximum deviation (detour). Two possible routes are visualized in Figure 3.11. Secondly, the driving dynamics need to be discretized in order to maintain a small search space and stay at least linear with respect to the model, since it is in general at least quadratic [HP14].

Using the FDPEN technique we construct for each periodic train path L a corresponding FG $K_L = (R_L, F_L)$ such that between each two consecutive stations along side its given path a flow edge for each meaningful[16] route is contained in F_L.

Example 3.42 (Routes for Periodic Train Paths)**.** Let $\{A, B, C, D, E\}$ be a set of stations that are visualized as network with routes in Figure 3.12. Further, let 1 $(A \to B \to C)$ and 2 $(D \to B \to E)$ be periodic train paths such that we have two FGs $K_i = (R_i, F_i)$ $(i \in \{1, 2\})$ with

$$R_1 = \{A_1, B_1, C_1\} \qquad\qquad R_2 = \{D_2, B_2, E_2\}$$

[16]as described in the previous paragraph

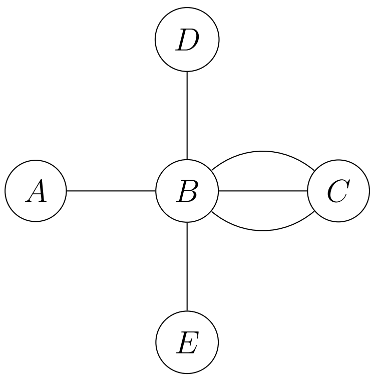


Figure 3.12: Exemplifying station network.

$$F_1 = \{e_1, \dots, e_4\} \qquad\qquad F_2 = \{e_5, e_6\}$$

as in Figure 3.13. Train path 1 has only a single route between A and B and three different routes between B and C, whereas train path 2 has no variation in its route which represents the classical timetabling with PESP.

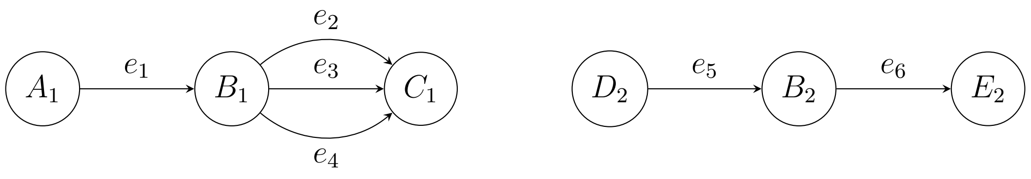


Figure 3.13: Exemplifying possible routes for periodic train path 1 (left) and 2 (right) as in Example 3.42.

Due to different infrastructural circumstances and possible different driving dynamics, the headway and driving[17] constraints may differ for each route. Hence, we need the constraints for each labeled flow edge of the corresponding DPEN separately.

Example 3.43 (FDPEN for Distinct Chain Paths)**.** Let 1 and 2 be periodic train paths as in Example 3.42 as well as its corresponding FGs K_1, K_2 and let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\{K_1, K_2\}$ with $T = 60$, $\mathcal{H} = \{n_1, \dots, n_6\}$ and $\mathcal{A} = \emptyset$ which is shown in Figure 3.14 whereas n_C and n_E represents the last departure nodes of train path 1 and 2, respectively. Further, we need a label function l such that for all $i \in \{1, \dots, 6\}$

$$l(e_i) = n_i$$

[17]including minimum and maximum stopping times

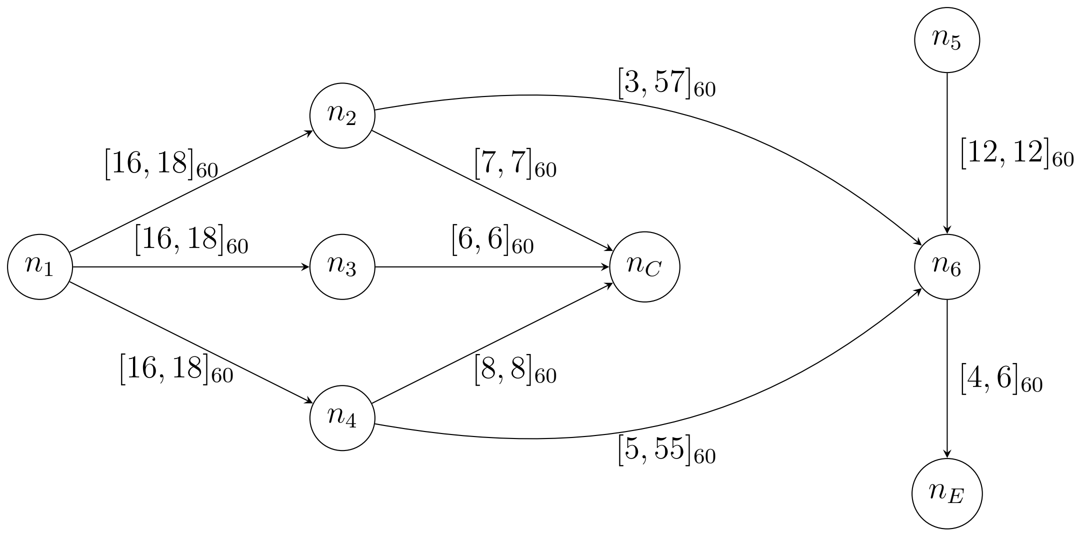


Figure 3.14: FDPEN as in Example 3.43.

that represent the departure events of each sub path (route) if needed. Please note, that the driving constraints between the nodes in $\{n_2, n_3, n_4\}$ and n_C are different which may be caused due to the different underlying infrastructure. This is as well manifested in the different headway constraints between the nodes in $\{n_2, n_4\}$ and n_6.

Calculating a solution for the given problem means finding a valid schedule for the DPEN and valid paths for each FG as in Definition 3.17.

Example 3.44 (Timetable for Distinct Chain Paths)**.** Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\{K_1, K_2\}$ as in Example 3.43. With the set of nodes $\mathcal{V}' = \{n_1, n_2, n_C, n_5, n_6, n_E\}$, the set of edges $\mathcal{E}' = \mathcal{E}$ and the schedule Π for $\mathcal{V}'$ such that

$$\begin{aligned}
\Pi(n_1) &= 0,\\
\Pi(n_2) &= 17,\\
\Pi(n_C) &= 24,\\
\Pi(n_5) &= 20,\\
\Pi(n_6) &= 32,\\
\Pi(n_E) &= 38,
\end{aligned}$$

the schedule Π is valid for $\mathcal{D}$ and $\mathcal{E}'$ which will be clarified in the following. The constraints that must hold under Π are highlighted in Figure 3.15 (left) for $\mathcal{D}$. Especially, the headway constraint for the labeled flow edge between n_2 and n_6 must hold under Π

since this route will be used by the resulting periodic train path.
Equations (3.1), (3.2) hold, because $n_C, n_E \in \mathcal{V}'$, $\mathcal{E}' = \mathcal{E}$ and it holds (3.3), because

$$\begin{aligned}
\Pi(n_2) - \Pi(n_1) = 17 \in [16,18]_{60} &\Rightarrow \Pi \models (n_1, n_2, [16,18]_{60})\\
\Pi(n_C) - \Pi(n_2) = 7 \in [7,7]_{60} &\Rightarrow \Pi \models (n_2, n_C, [7,7]_{60})\\
\Pi(n_6) - \Pi(n_2) = 15 \in [3,57]_{60} &\Rightarrow \Pi \models (n_2, n_6, [3,57]_{60})\\
\Pi(n_6) - \Pi(n_5) = 12 \in [12,12]_{60} &\Rightarrow \Pi \models (n_5, n_6, [12,12]_{60})\\
\Pi(n_E) - \Pi(n_6) = 6 \in [4,6]_{60} &\Rightarrow \Pi \models (n_6, n_E, [4,6]_{60})
\end{aligned}$$

as in Definition 3.3. Furthermore, all equations in Definition 3.17 are fulfilled, because we have for each FG in $\{K_1, K_2\}$ exactly one source and exactly one destination and the flow conservation holds as well, which is all denoted as

$$\begin{aligned}
n_1 \in \mathcal{V}' \cap \mathit{range}(l) = \{n_1, n_2, n_5, n_6\},\ l^{-1}(n_1) = (A_1, B_1) \in F,\ A_1 \in S_{K_1} &\Rightarrow (3.17),\\
n_5 \in \mathcal{V}' \cap \mathit{range}(l) = \{n_1, n_2, n_5, n_6\},\ l^{-1}(n_5) = (D_2, B_2) \in F,\ D_2 \in S_{K_2} &\Rightarrow (3.17),\\
n_2 \in \mathcal{V}' \cap \mathit{range}(l),\ l^{-1}(n_2) = (B_1, C_1) \in F,\ C_1 \in D_{K_1} &\Rightarrow (3.18),\\
n_6 \in \mathcal{V}' \cap \mathit{range}(l),\ l^{-1}(n_6) = (B_2, E_2) \in F,\ E_2 \in D_{K_2} &\Rightarrow (3.18),\\
\bigcup_{(m,B_1)\in I_K(B_1)} l(m, B_1) = \{n_1\}, \qquad \bigcup_{(B_1,o)\in O_K(B_1)} l(B_1, o) = \{n_2, n_3, n_4\}&\\
\Rightarrow |\{n_1\} \cap \mathcal{V}'| = |\{n_2, n_3, n_4\} \cap \mathcal{V}'| = 1 \in \{0,1\} &\Rightarrow (3.19)\\
\bigcup_{(m,B_2)\in I_K(B_2)} l(m, B_2) = \{n_5\}, \qquad \bigcup_{(B_2,o)\in O_K(B_2)} l(B_2, o) = \{n_6\}&\\
\Rightarrow |\{n_5\} \cap \mathcal{V}'| = |\{n_6\} \cap \mathcal{V}'| = 1 \in \{0,1\} &\Rightarrow (3.19)
\end{aligned}$$

with $\mathit{range}(l) = \mathcal{H}$ being the range of the label function l. Hence, Π is valid for $\mathcal{D}$, $\mathcal{E}'$ under $\{K_1, K_2\}$.
Finally, we can extract the paths ρ_1 and ρ_2 for the periodic train paths 1 and 2 from $\mathcal{V}'$ such that $\rho_1 = (e_1, e_2)$ and $\rho_2 = (e_5, e_6)$, because

$$\begin{aligned}
\rho_1 : \mathcal{V}' \cap \bigcup_{e\in F_1} l(e) &= \{n_1, n_2, n_5, n_6\} \cap \{n_1, n_2, n_3, n_4\} = \{n_1, n_2\},\\
\rho_2 : \mathcal{V}' \cap \bigcup_{e\in F_2} l(e) &= \{n_1, n_2, n_5, n_6\} \cap \{n_5, n_6\} = \{n_5, n_6\}
\end{aligned}$$

as in (3.26). The highlighted paths for the FGs are shown in Figure 3.15 (right).

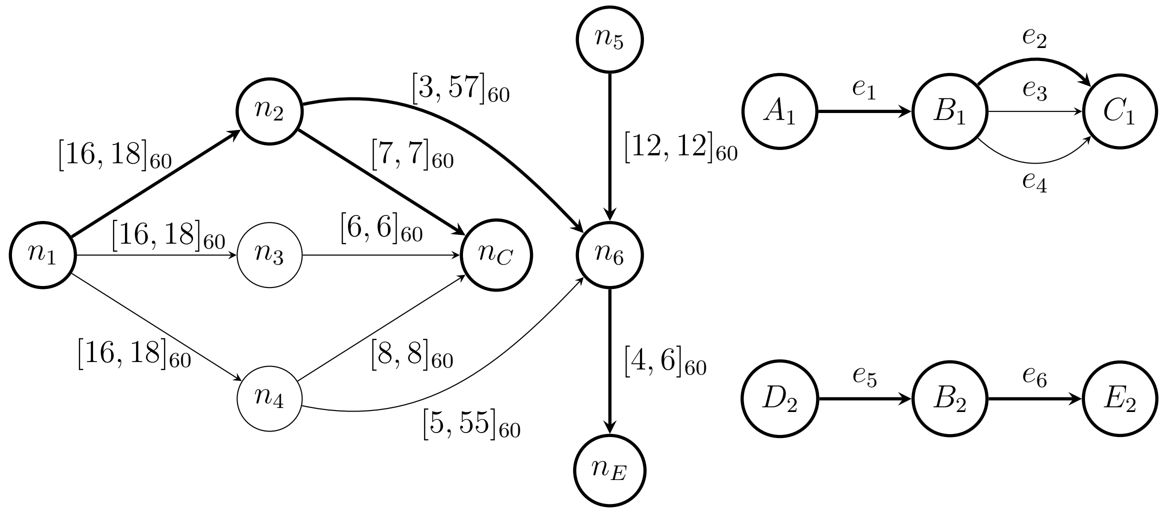


Figure 3.15: Highlighted constraints (left) and paths (right) for the valid schedule as in Example 3.44.

Using the SAT-based technique, we have to encode the FDPEN as in Section 3.2.1 with Definition 3.40. Because the introduction to this method is not yet fully implemented in the software system TAKT, there will be no computational results for this approach, yet. Furthermore, this is left for further research[18] but seems to provide very high potential [Wün16] especially on highly crowded sub paths or tracks with multiple choices for its corresponding main direction as already used in the rail freight train path approach in the following Section 3.3.2 and Section 5.4.

3.3.2 Duplicated Chain Paths

Inserting rail freight train paths into a given train path system – that is a public transport network with given periodic train paths – is an ever-growing need for railway infrastructure companies as DB Netz AG [FP14]. Thus, we need to tackle this problem based on the given structure (FDPESP) that we developed in Section 3.1.2.

The given input data is a set of source nodes and set of destination nodes that represent stations – respectively tracks in stations – of rail freight train paths. In Figure 3.16 we can see a railway network of Germany with its highlighted relation (source and destination) between Mainz and Karlsruhe. Thus, the flow of the respective relation is already given. However, the driving dynamics and the precise train path is yet open. Here, we discretize the given parameters based on a given detour factor [WON12; Opi09].

[18]which is under current scientific investigations at the Chair of Traffic Flow Science [Wün16]

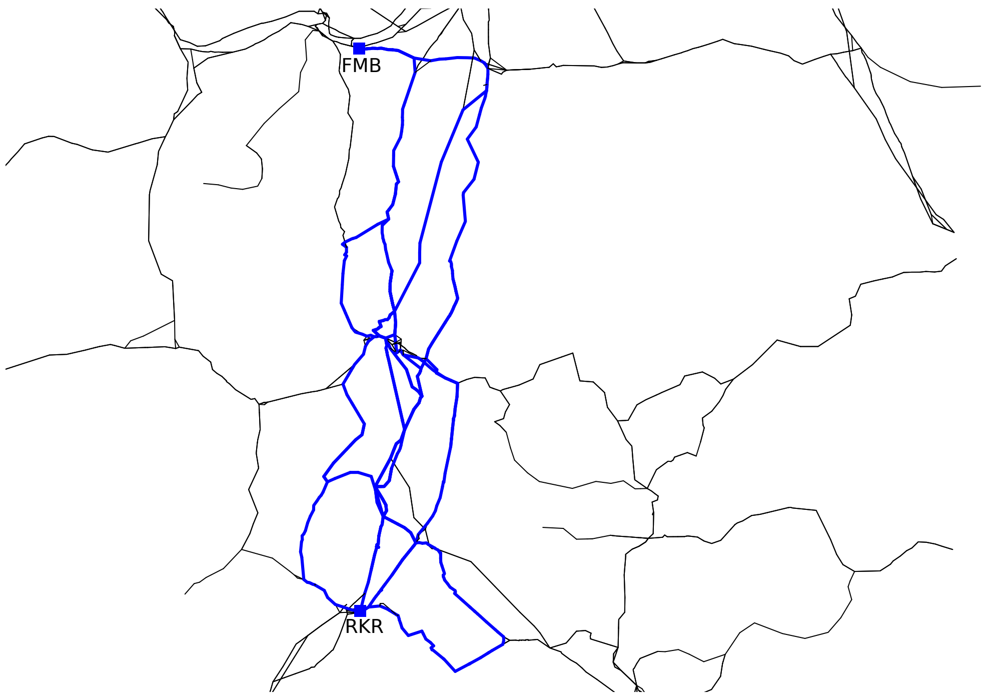


Figure 3.16: Relation of to be inserted rail freight train paths between FMB (Mainz-Bischofsheim, Germany) RKR (Karlsruhe, Germany) with possible routes.

Then, each possible train path is cut in small pieces[19] such that it possibly stops in a lot of stations in between which results in a train path sub network as needed for an FG. The flow nodes represent tracks in certain stations and the flow edges represent the infrastructure in between to reach these tracks, respectively their last signal.

Example 3.45 (Freight Train Sub Flow Graph)**.** Let $\{A, B, C, D, E\}$ be a set of stations with tracks in $\{1, 2\}$ as index. For example, D_2 means the track 2 in station D. We create an FG $K = (R, F)$ with

$$\begin{aligned} R &= \{A_1, A_2, B_1, C_1, C_2, D_1, E_1, E_2\} \\ F &= \{e_1, \dots, e_7\} \end{aligned}$$

as in Figure 3.17. For example, we have a sub train path (flow edge) $(B_1, C_2) \in F$ which means the resulting freight train may drive from track 1 in station B track 2 in station C. A resulting path could be possibly $\rho = (e_2, e_4, e_7)$ which is highlighted in Figure 3.18. Basically, it means that the resulting periodic train path travels from track 2 in station A to track 2 in station E via station C and D (on track 1).

[19]i. e., a sub train path

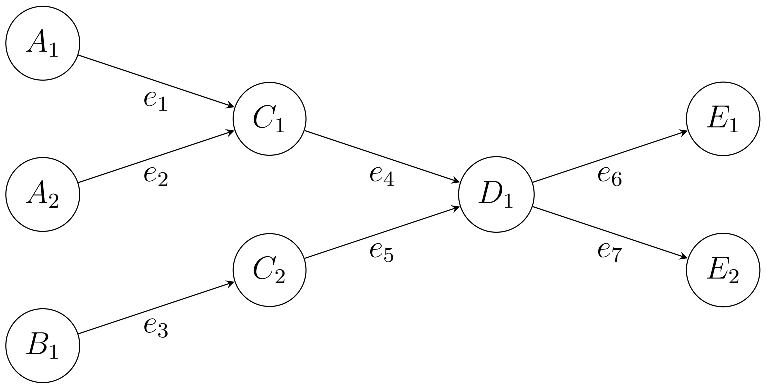


Figure 3.17: FG as in Example 3.45.

Duplicating the given FG results in more rail freight train paths. Subsequently, we need to copy the whole structure; that is the set of flow nodes and flow edges. This is based on the reason, that they use the exact same infrastructure, however, they may use different paths. Hence, we superscript each flow edge and flow node with a distinct number for each flow graph. Hence, the set of nodes and paths are disjoint. Let the number of to be inserted train paths be τ.

Example 3.46 (Duplicated Train Flow Graphs)**.** Let $K = (R, F)$ be an FG as in Example 3.45. With the number of to be inserted rail freight train paths $\tau = 4$ we have the set of FGs $\{K^1, \ldots, K^4\}$. Hence, we can construct each FG $K^i = (R^i, F^i)$ $(i \in \{1, \ldots, 4\})$ out of the base FG K such that

$$\begin{aligned} R^i &= \{A_1^i, A_2^i, B_1^i, C_1^i, C_2^i, D_1^i, E_1^i, E_2^i\}, \\ F^i &= \{e_1^i, \ldots, e_7^i\} \end{aligned}$$

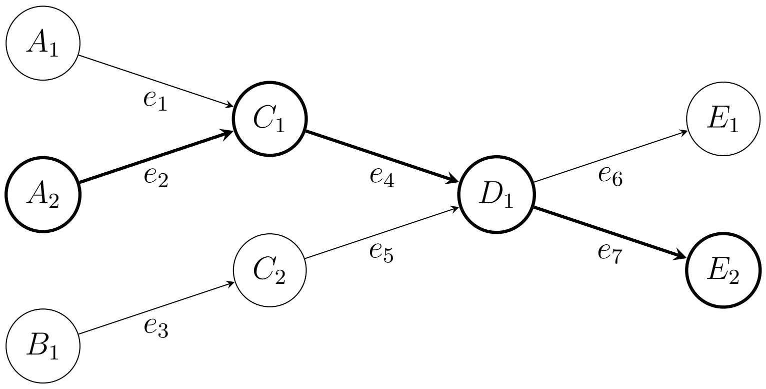


Figure 3.18: Path as in Example 3.45.

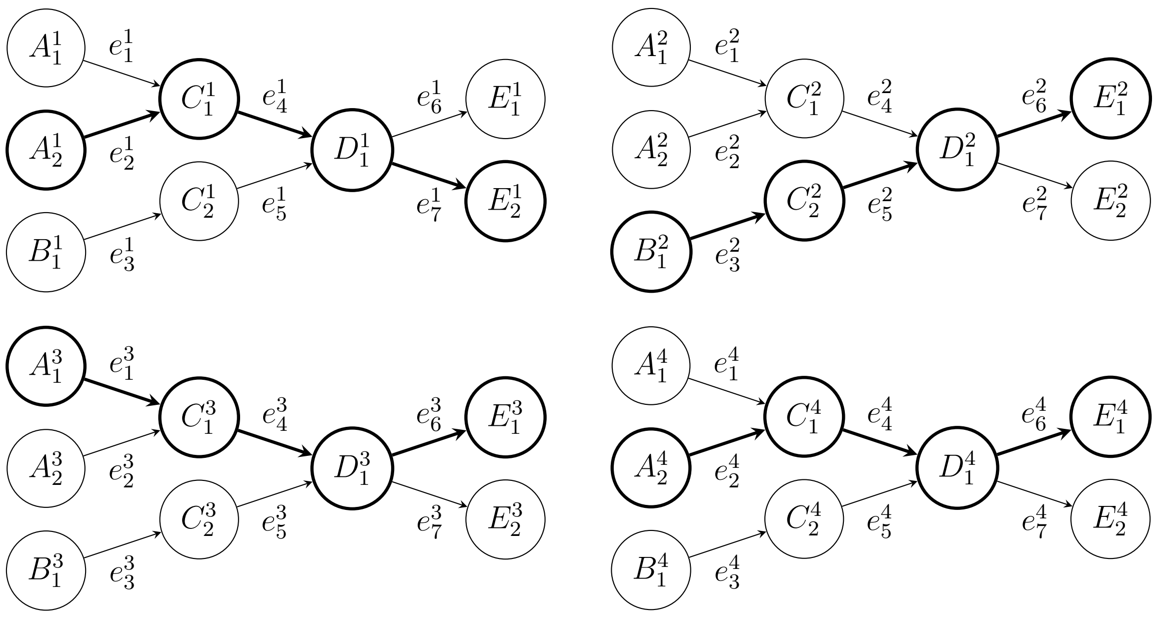


Figure 3.19: Duplicated FG as in Example 3.46.

with a possible solution (path) ρ_i for each FG K^i ($i \in \{1, \dots, 4\}$)

$$\begin{aligned}\rho_1 &= (e_2^1, e_4^1, e_7^1),\\ \rho_2 &= (e_3^2, e_5^2, e_6^2),\\ \rho_3 &= (e_1^3, e_4^3, e_6^3),\\ \rho_4 &= (e_2^4, e_4^4, e_6^4),\end{aligned}$$

which is highlighted in Figure 3.19 for each respective FG.

It remains to connect each flow edge to a respective node to the given DPEN. As described before, each DPEN consists of different constraint types. With respect to freight train paths, we mainly need headway and drive-stop constraints. Each flow edge (sub path) is connected to its successor edges by drive-stop constraints and has headway constraints for each intersecting train path with respect to the infrastructure [Opi09; Nac98]. This shall be shown by the following simplified network.

Example 3.47. Let $K^1 = (R^1, F^1)$ and $K^2 = (R^2, F^2)$ be duplicated FGs as visualized in Figure 3.20 such that for each $i \in \{1, 2\}$:

$$\begin{aligned}R^i &= \{A^i, B^i, C^i\},\\ F^i &= \{e_1^i, e_2^i\}.\end{aligned}$$

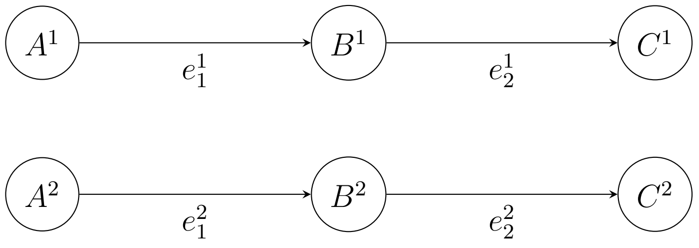


Figure 3.20: Simplified duplicated FG as in Example 3.47.

Further, let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\{K^1, K^2\}$ with period $T = 60$, $\mathcal{H} = \{n_1^1, n_2^1, n_1^2, n_2^2\}$ and $\mathcal{A} = \emptyset$ which is shown in Figure 3.21 alongside a label function l such that for all $i, j \in \{1, 2\}$

$$l(e_i^j) = n_i^j$$

that represent the departure events of each sub path if needed. The periodic events m_i ($i \in \{1, 2, 3\}$) may represent some passenger path like an inter city train. Each constraint within a (passenger or freight) periodic train path represents a drive-stop constraint and all constraints in between represent headways. Please note the wide spans within the freight train paths. These represent possible stopping times in station B for up to 30 min. These are parameters which may be adjusted accordingly.

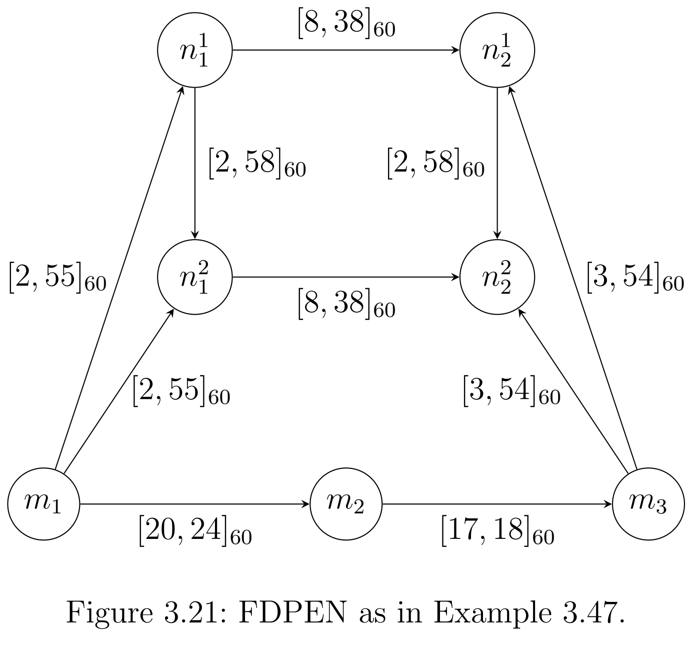


Figure 3.21: FDPEN as in Example 3.47.

In Figure 3.23 a timetable between Gutenfürst and Zwickau (Sachs) Hbf with a period of 120 min is shown, whereas only the passenger train paths' timetable can be seen in

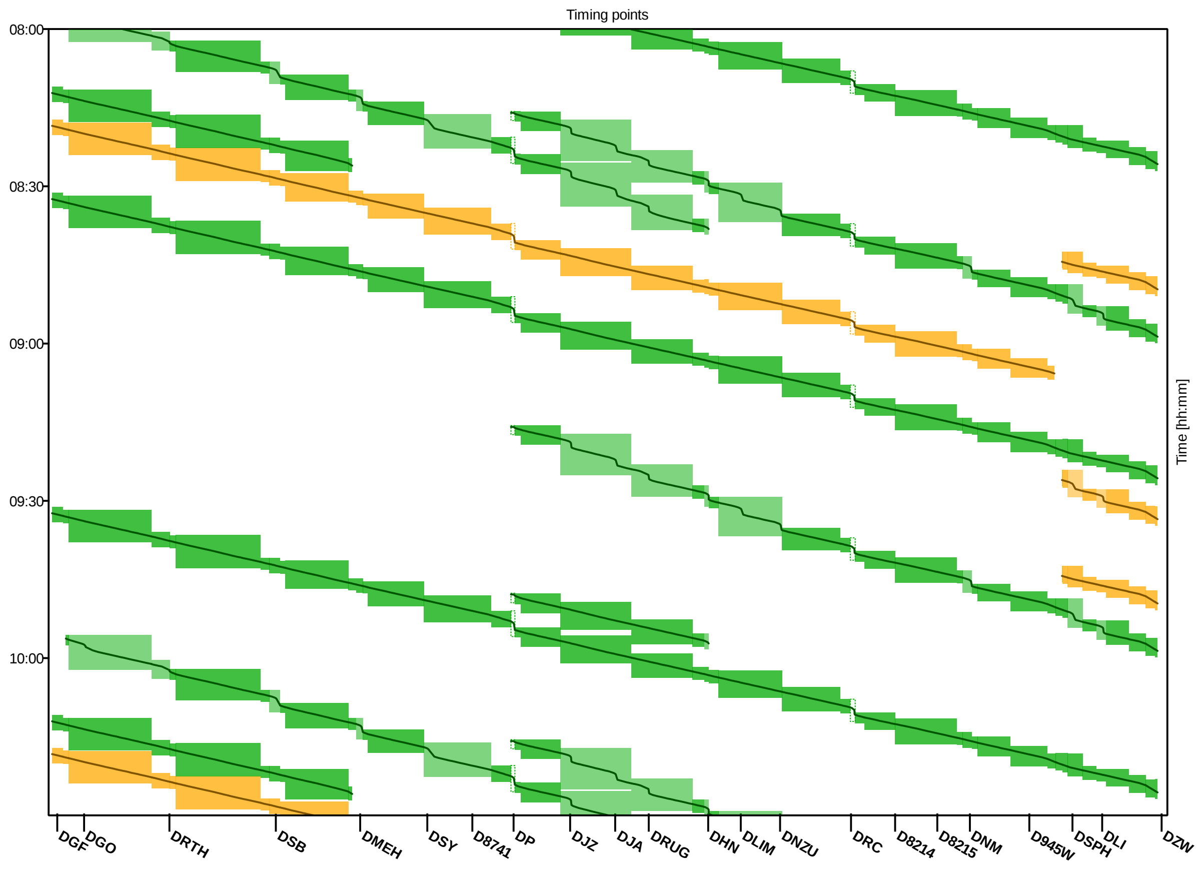


Figure 3.22: Timetable (time-distance graph) for passenger train paths between DGF (Gutenfürst, Germany) and DZW (Zwickau (Sachs) Hauptbahnhof, Germany).

Figure 3.22. The inserted periodic rail freight train paths are visualized in blue. If we discretize the time frame into $2\,h$ we detect 9 inserted paths into the existing passenger train path system maintaining a conflict-free timetable.

Subsequently, with the appropriate encoding enc_fdpen_l in Definition 3.40 we can solve these instances with a state-of-the-art SAT solver. In this example we can see headway constraints between freight train sub paths as well, since if both sub paths are in the resulting path, they need the safety time.

Experimental results for inserting rail freight trains into passenger train path systems can be found in Section 5.4.

3.3.3 Non-connected Flow Graphs

Track allocation is an important part of the railway timetabling process that assigns each periodic train path a track in each station it passes. For example, this problem is solved

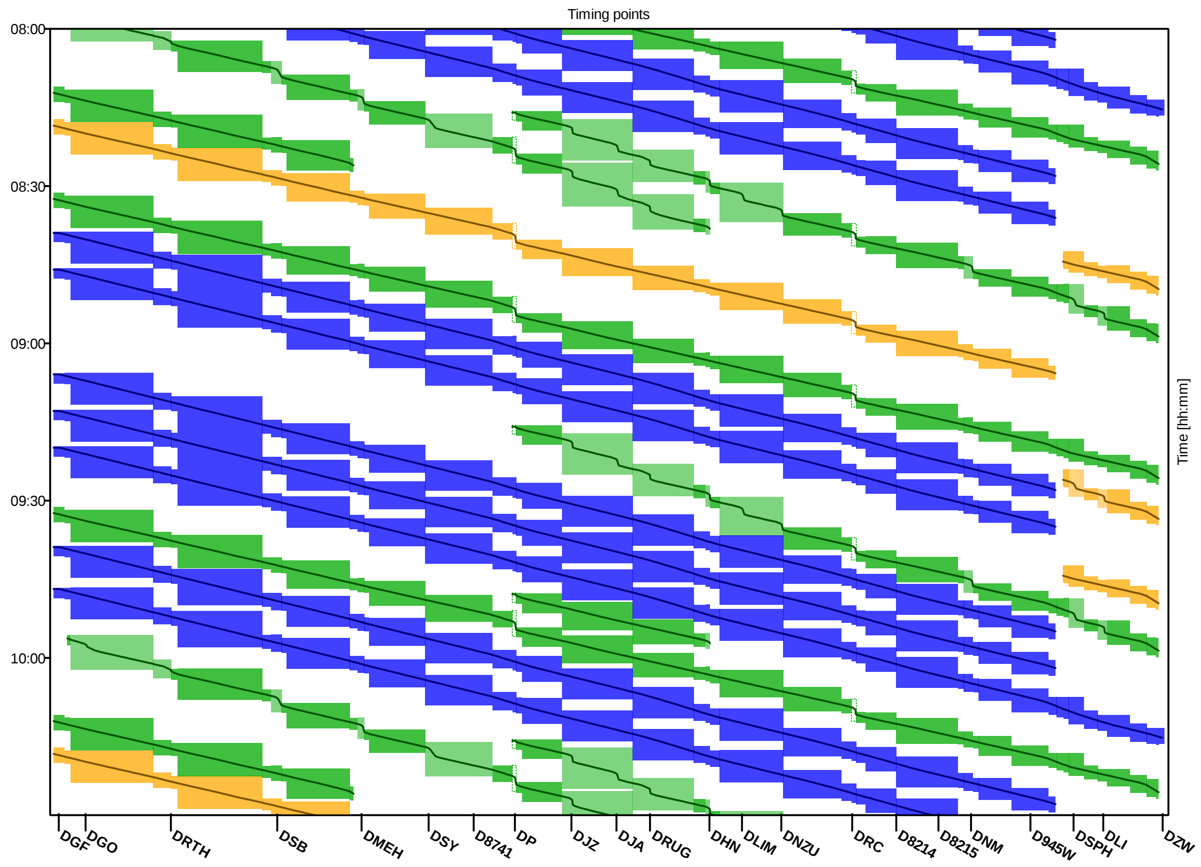


Figure 3.23: Timetable with inserted rail freight train paths (blue) between DGF (Gutenfürst, Germany) and DZW (Zwickau (Sachs) Hauptbahnhof, Germany).

in the software STATIONS [Zwa+96; ZKV01; Kro+09]. In Figure 3.24 a small station (Riesa) is visualized, where each box represents a blocking time (y-axis) of a periodic train path on its corresponding track (x-axis).

This process will be used after the timetabling step of the whole network. Regarding the track allocation process already in the previous phase would explode[20] the computational complexity. Thus, each train path uses the main track – or side track if stopping – in each station neglecting the headway constrained within a station. In the next phase, all possible routes through the station for each train path are regarded, requiring and holding the headway constraints.

Due to computation time reasons, each periodic train path in all stations has fixed departure times, excluding the one that is processed. In PESP terms this is reflected by fixed periodic events. Subsequently, the PEN can be cut into all constraints that only intersects with events of the processed station where each train path shall be allocated

[20]at least with the current state-of-the-art techniques and solvers

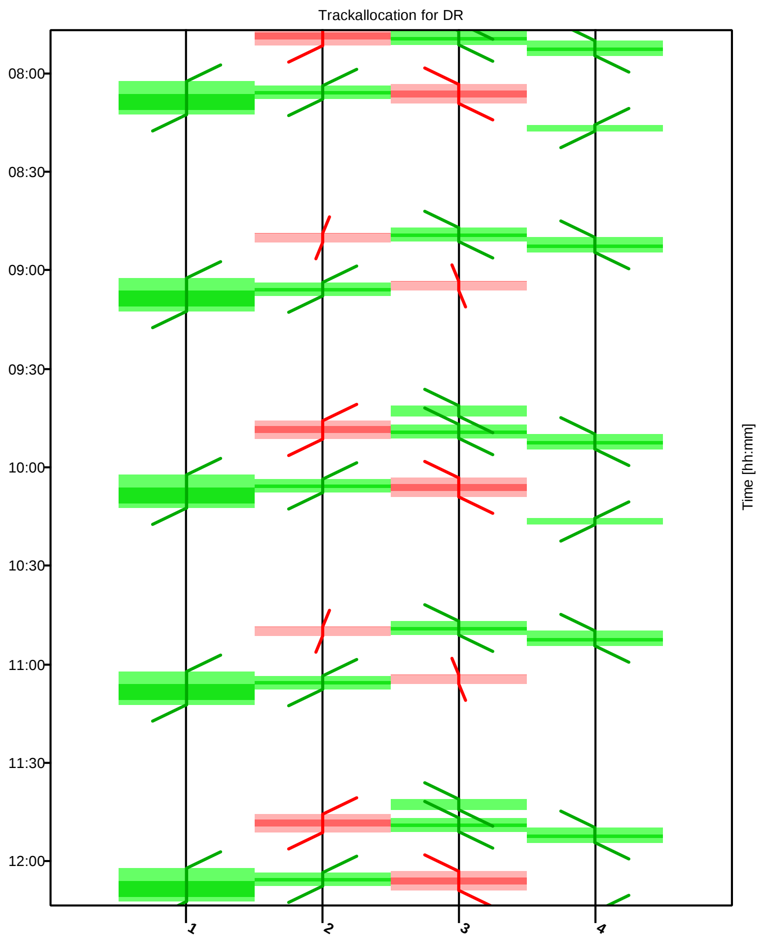


Figure 3.24: Track allocation for the small station DR (Riesa, Germany, 4 tracks).

to a certain track.

Each route r^L within a station for a train path L will be contained in the set of routes R_L. The set of routes is finite and given in advance due to infrastructural data and train type restrictions like track priorities and maximum detour factor. In Figure 3.25 4 possible routes for a train path in station Dresden Hbf are visualized that have been routed with the software system TAKT.

Using the FDPEN technique we construct for each train path L an FG $K_L = (R_L, F_L)$ such that for each route $r^L \in R_L$ the artificial nodes r_1^L and r_2^L are contained in R_L and the route itself represents the edge between the defined nodes with $r = (r_1^L, r_2^L) \in F_L$. Since the set of nodes of all pair of routes are disjoint, no route (flow edge) is connected to another one.

Example 3.48 (Flow Graphs for Track Allocation)**.** Let A and B be the only periodic train paths for the to be regarded station. Further, let $R_A = \{a^A, b^A\}$ and $R_B =$

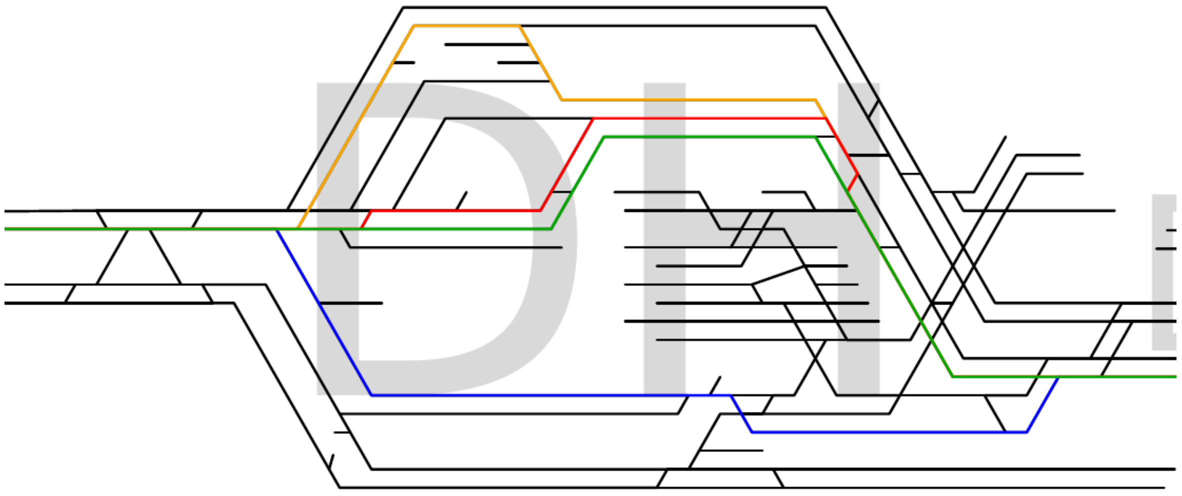


Figure 3.25: Four possible routes for a periodic train path in station DH (Dresden Hauptbahnhof, Germany).

$$\begin{tikzcd}
 & & a_1^B \arrow[r, "a^B"'] & a_2^B \\
a_1^A \arrow[r, "a^A"'] & a_2^A & & \\
 & & b_1^B \arrow[r, "b^B"'] & b_2^B \\
b_1^A \arrow[r, "b^A"'] & b_2^A & & \\
 & & c_1^B \arrow[r, "c^B"'] & c_2^B
\end{tikzcd}$$

Figure 3.26: Non-connected flow graphs K_A (left) and K_B (right) as in Example 3.48.

$\{a^B, b^B, c^B\}$ be the respective set of routes. Then, we can construct the FGs K_A and K_B for the periodic train paths A and B, respectively, such that

$$R_A = \{a_1^A, a_2^A, b_1^A, b_2^A\}, \qquad R_B = \{a_1^B, a_2^B, b_1^B, b_2^B, c_1^B, c_2^B\},$$
$$F_A = \{a^A, b^A\}, \qquad F_B = \{a^B, b^B\}.$$

Both graphs are displayed in Figure 3.26 with possible paths $\rho_A = (a^A)$ and $\rho_B = (c^B)$.

In the previous sections for the distinct and duplicated chain path cases, we needed to insert headway constraints for each labeled flow edge. Likewise, we have to introduce the needed label function and need to connect each corresponding node – that is the

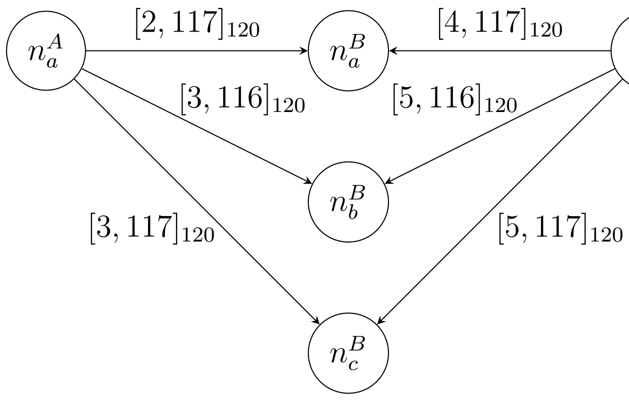


Figure 3.27: FDPEN as in Example 3.49.

departure event of the given route – to its required headway constraints to the other periodic train paths within the railway network. Again, these are based on the trains' driving dynamics and the underlying infrastructure.

Example 3.49 (FDPEN for Track Allocation)**.** Let K_A and K_B be the FGs as in Example 3.48. Further, let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\{K_A, K_B\}$ with the period $T = 120$, $\mathcal{H} = \{n_a^A, n_b^A, n_a^B, n_b^B, n_c^B\}$ be the set of decision nodes and $\mathcal{A} = \emptyset$ that is displayed in Figure 3.27 with a corresponding label function for each periodic train path $L \in \{A, B\}$ and each $r \in \{a, b\}$

$$l(r^L) = n_r^L$$

and $l(c^B) = n_c^B$. Because these are the only regarded trains, only the corresponding labeled nodes and its respective constraints are regarded in the FDPEN. We can see that only headway constraints are contained and the different driving dynamics are exemplified in the increasing lower bounds of route b for train path A.

If we evaluate a valid timetable for the given FDPEN, we can extract the used tracks of the given solution. An exemplifying scenario is shown in Figure 3.28 for a medium sized station. No blocked times are intersecting which indicates that the corresponding timetable is conflict-free.

The track allocation method offers a promising tool for handling the stations no more as black boxes. Experimental results suggest a sound integration into the timetabling process as proposed in Section 5.5.

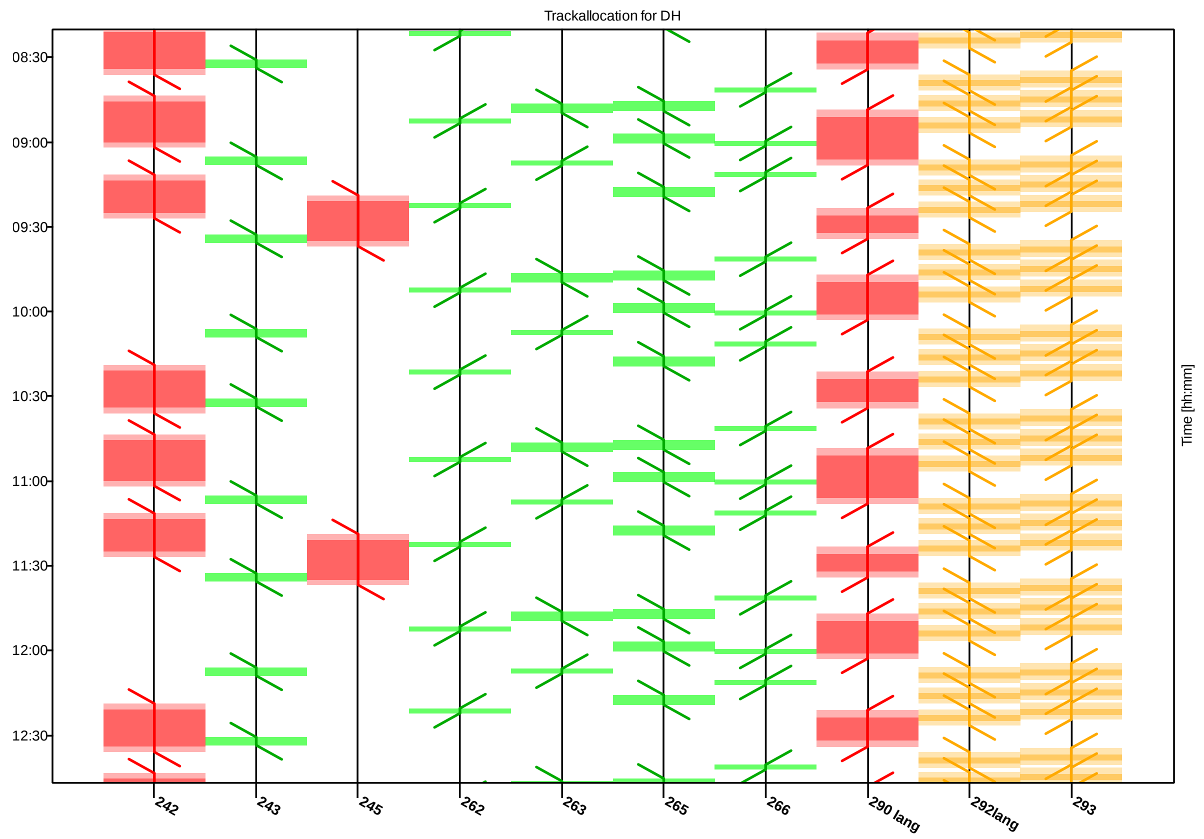


Figure 3.28: Allocated tracks for all train paths in station DH (Dresden Hauptbahnhof, Germany).

4 Optimization in Periodic Event Scheduling

In the previous section, the presented problems are introduced as decisional problems. Since all given problems are at least NP-complete, those tasks are until now not tractable [Coo71]. However, most applications require optimization possibilities in order to gain even better results [Opi09; Kro+08]. Consequently, this section covers an overview for some of the already developed optimization approaches, introduces new models for several applications and covers for most an appropriate MaxSAT encoding.

In Section 4.1 the respective MIP models are presented whereas in Section 4.2 the corresponding possible encodings to MaxSAT are shown, which ends with a soundness and completeness theorem for the encoding of optimizing FDPENs.

4.1 Linear Objectives and Applications

First of all, it is important to know that using MIP solvers to optimize the given instances, we need an initial solution, since state-of-the-art MIP solvers currently lack to calculate initial solutions efficiently [KN13] or even effectively [GRB15]. However, applications of the PESP model in real-world railway scenarios have a huge amount of hard constraints [Küm+13], especially headway, such that no 0 solution[1] can be provided to the solvers. Hence, using MIP techniques in the following for various periodic scheduling problems still needs the encodings in Section 3.2 alongside state-of-the-art SAT solvers to generate an initial solution. The base scheme for possible approaches is shown in Figure 4.1, whereas "heuristic" refers to iterative calls to SAT solvers. The first step ("model") and last step ("decode") are out of scope in this work. However, the center part is described in this work.

Nevertheless, formulating optimization problems in periodic scheduling problems as linear objectives are for many researches most intuitive and provides a generic, mathemat-

[1]0 solution means that all variables are set to 0 and is feasible for the given model

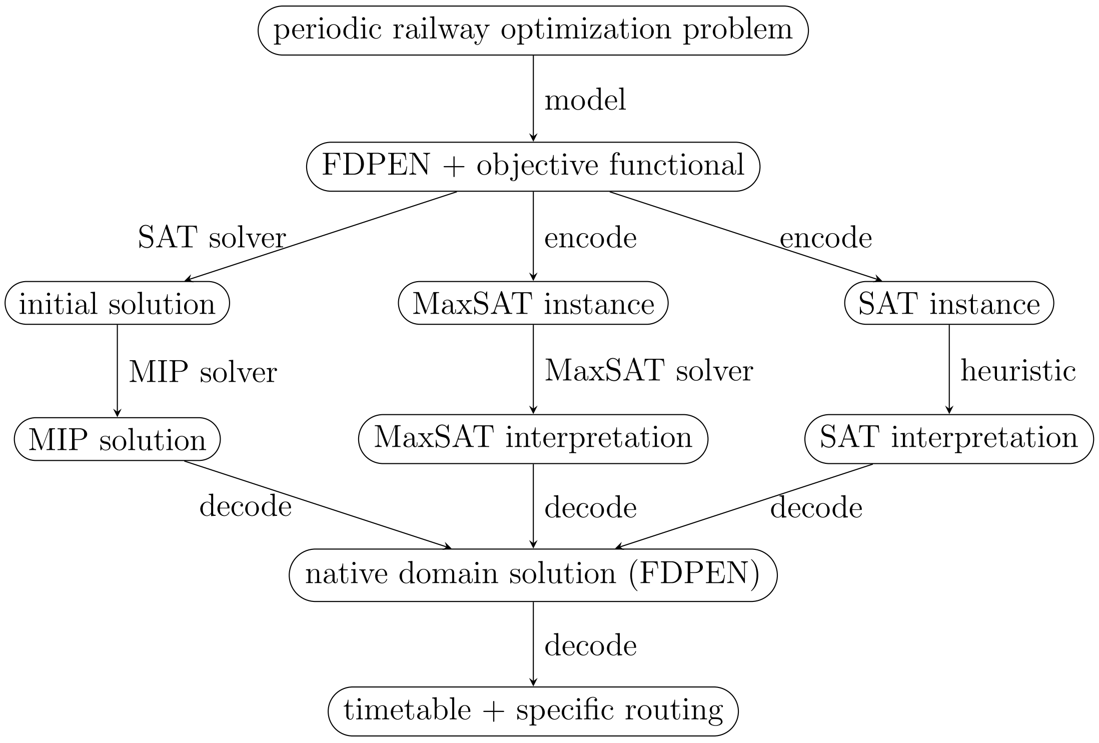


Figure 4.1: Approaches for optimization in periodic timetabling.

ical formalization of the underlying problem. Some of the introduced model formulations have been already introduced, however, we need them yet again for the corresponding MaxSAT encodings in Section 4.2. In Section 4.1.1 the classical optimization of weighted slacks will be shown. Then, in Section 4.1.2 the maximization of constraints that hold under a certain schedule is discussed and finally, in Section 4.1.3 optimization criteria for the FDPEN approaches are presented and discussed.

In the consecutive sections, we will use binary variables like $x \in \{0,1\}$ for MIP models. For convenience, we use as well its complement $\bar{x} = 1 - x$ without explicitly appending these constraints to the model itself.

4.1.1 Minimization of Weighted Slacks

This problem[2] has been introduced and discussed heavily in the literature [NO07; Nac98; Lie06; Kro+09; LM07a]. Given a PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ with constraints $e \in \mathcal{E}$ of the structure $a(e) = [l, u]_T$ we can formulate with a weight function ω such that

$$\omega : \mathcal{E} \to \mathbb{N} \cup \{\infty\} \tag{4.1}$$

[2]or variants of this problem

the set of to be optimized constraints with

$$\mathcal{E}_\omega := \{e \in \mathcal{E} \mid \omega(e) < \infty\}. \tag{4.2}$$

The remaining constraints $\mathcal{E} \setminus \mathcal{E}_\omega$ are not considered for the optimization. For example, in the software system TAKT all headway constraints have an infinite weight, because only symmetry, connection and waiting constraints are optimized [Opi09].

Consequently, for a schedule Π we can define the objective functional

$$\sum_{e=(i,j,[l,u]_T)\in\mathcal{E}_\omega} \omega(e)\big(((\Pi(j) - \Pi(i)) \mod T) - l\big) \to \min \tag{4.3}$$

that minimizes the sum of weighted slacks that represent the deviation of each corresponding constraint's lower bound. Subsequently, we can calculate the slack for each constraint $e = (i, j, [l, u]_T) \in \mathcal{E}_\omega$ with

$$\sigma(e, \Pi) := ((\Pi(j) - \Pi(i)) \mod T) - l \tag{4.4}$$

Because 4.3 is not linear, the PESP constraint $e = (i, j, [l, u]_T)$ can be formulated [Odi94] with the modulo parameter $z_e \in \mathbb{Z}$ for each constraint with

$$l \leq \Pi(j) - \Pi(i) - z_e \cdot T \leq u. \tag{4.5}$$

Subsequently, the objective functional alongside its model can be reformulated linearized as

$$\sum_{e=(i,j,[l,u]_T)\in\mathcal{E}_\omega} \omega(e)\big(\Pi(j) - \Pi(i) - l - z_e \cdot T\big) \to \min \tag{4.6}$$

subject to

$$\begin{aligned}
&l \leq \Pi(j) - \Pi(i) - z_e \cdot T \leq u, \quad \forall e = (i, j, [l_u]_T) \in \mathcal{E} \\
&\Pi(j), \Pi(i) \in [0, T-1] \subseteq \mathbb{Z}, \quad \forall i, j \in \mathcal{V} \\
&z_e \in \mathbb{Z}, \quad \forall e \in \mathcal{E}
\end{aligned}$$

which is indeed linear, since $\omega(e)$ is constant.

The applications of this method have been discussed in the literature [Opi09; Nac98; Kro+09] and can be used for example for resolving local conflicts [Gro12b] or minimizing passenger waiting times, which is schematically exemplified for one train path in Figure 4.2.

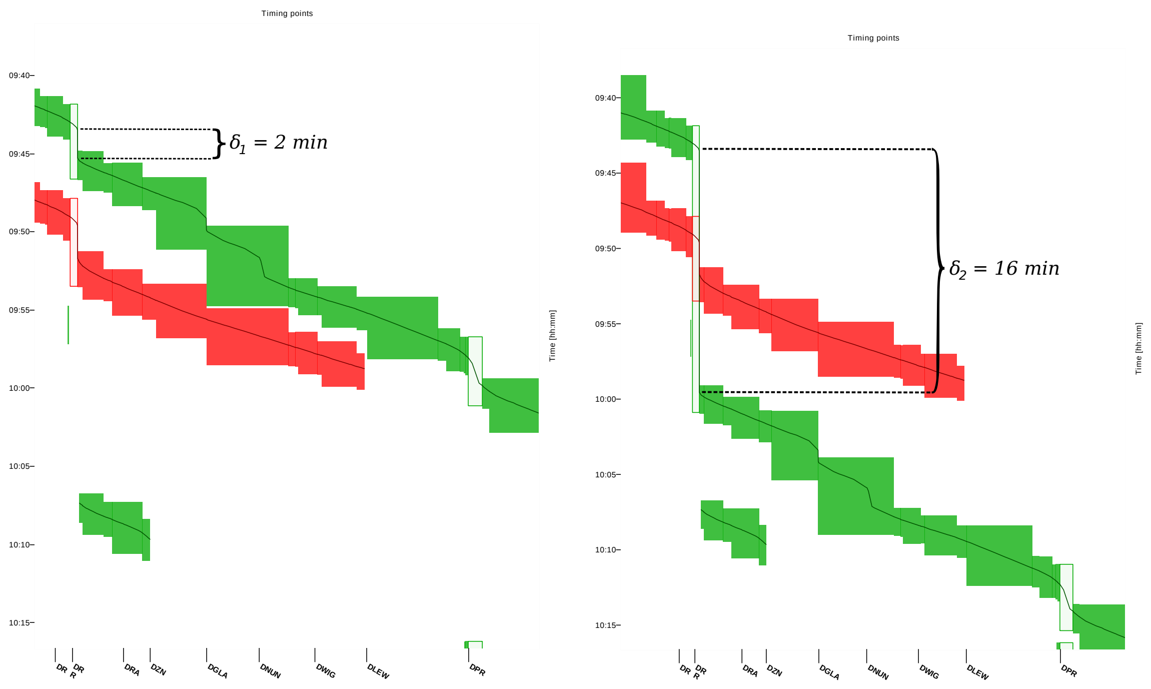


Figure 4.2: Regional train path (green) with 2 min slack (left) and 16 min slack (right) in a station.

4.1.2 Satisfiable Constraint Maximization

Maximizing the number of constraints that hold under a schedule has two major roles in railway timetabling. On the one hand, in case the PEN is infeasible, two possible ways can be developed. Firstly, a smallest infeasible subnetwork (local conflict) could be extracted [Opi09; Gro12b] or secondly, a maximum feasible PEN with respect to the number of constraints can be generated such that only possibly few constraints do not hold under the schedule. On the other hand, having a feasible PEN is often extended by connections resulting of passenger flows such that as many connections as possible shall hold for transfers between two trains [Opi09]. The latter has already been developed and successfully applied by Liebchen [Lie08; Lie06].

Both cases need the structure of decisional PENs in form of DPENs. Hence, we need a DPEN $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$. Furthermore, for convenience we can assume that the set of decisional nodes is empty such that $\mathcal{H} = \emptyset$. Again, we apply the weight function as in (4.1) with

$$\forall e \in \mathcal{E} \setminus \mathcal{A} : \omega(e) = \infty, \forall e \in \mathcal{A} : \omega(e) < \infty \tag{4.7}$$

which implies that only decisional constraints have a weight less than ∞. Consequently,

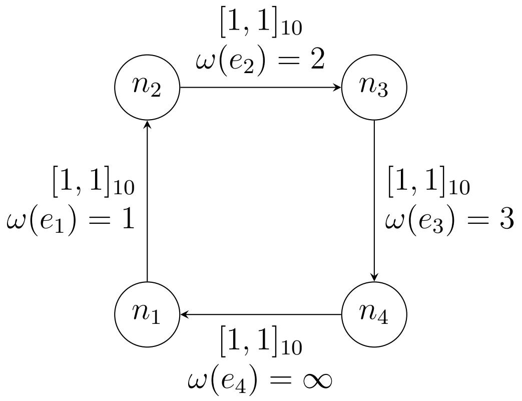


Figure 4.3: Decisional PEN as in Example 4.1

we can introduce the objective functional with a set of edges $\mathcal{E}' \subseteq \mathcal{E}$ such that Π is valid for $\mathcal{V}' = \mathcal{V}$ and $\mathcal{E}'$ as in Definition 3.3 with

$$\sum_{e \in \mathcal{E}' \cap \mathcal{A}} \omega(e) \to \max . \tag{4.8}$$

If the weight function only maps the constraints to 1 such that for each decisional edge $e \in \mathcal{A}$ it holds $\omega(e) = 1$ we can simplify (4.8) to

$$|\mathcal{E}' \cap \mathcal{A}| \to \max .$$

Example 4.1. Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, 10, \emptyset, \mathcal{A})$ be a DPEN with $\mathcal{V} = \{n_1, \ldots, n_4\}$, $\mathcal{E}$ such that $e_i = (n_i, n_{(i \mod 4)+1}, [1,1]_{10} \in \mathcal{E}$ $(i \in \{1, \ldots, 4\})$ and the set of decisional constraints $\mathcal{A}$ such that $e_j \in \mathcal{A}$ $(j \in \{1,2,3\})$. Further let ω be the weight function such that for each $e_j \in \mathcal{A} : \omega(e_j) = j$ $(j \in \{1,2,3\})$ and $\omega(e_4) = \infty$. The graph is visualized in Figure 4.3.
Let $\mathcal{E}_1 = (e_1, e_2, e_4) \subseteq \mathcal{E}$, $\mathcal{E}_2 = (e_2, e_3, e_4) \subseteq \mathcal{E}$ be sets of edges and let Π_1, Π_2 be schedules for $\mathcal{V}$ with

$$\begin{aligned} \mathcal{E}_1 &: \Pi_1(n_4) = 0, \Pi_1(n_1) = 1, \Pi_1(n_2) = 2, \Pi_1(n_3) = 3, \\ \mathcal{E}_2 &: \Pi_2(n_2) = 0, \Pi_2(n_3) = 1, \Pi_2(n_4) = 2, \Pi_2(n_1) = 3. \end{aligned}$$

Then, Π is valid for $\mathcal{E}_1$ and $\mathcal{E}_2$, because for each $e_i \in \mathcal{E}_j$ $(j \in \{1,2\})$ it holds

$$e_i = (n_i, n_{(i \mod 4)+1}, [1,1]_{10}) : \Pi_j(n_{(i \mod 4)+1} - \Pi(n_i) \equiv_{10} 1 \in [1,1]_{10}$$

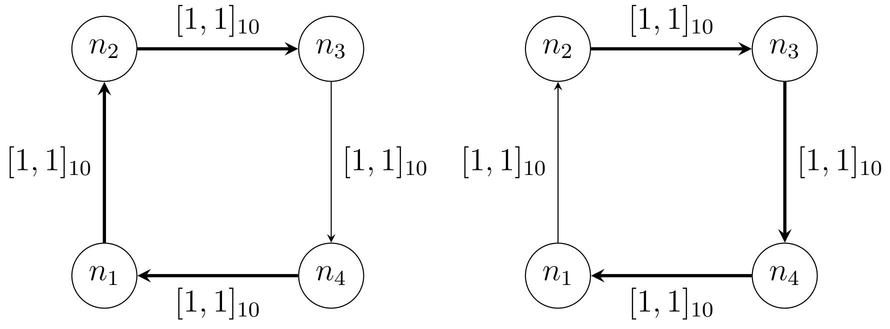


Figure 4.4: Highlighted constraints for $\mathcal{E}_1$ (left) and $\mathcal{E}_2$ (right) as in Example 4.1

The highlighted constraints for $\mathcal{E}_1$ and $\mathcal{E}_2$ can be seen in Figure 4.4. The evaluated objective functional in (4.8) can be evaluated such that

$$\begin{aligned} \mathcal{E}_1 : \sum_{e \in \mathcal{E}_1 \cap \mathcal{A}} \omega(e) &= \sum_{e \in \{e_1, e_2\}} \omega(e) = 1 + 2 = 3, \\ \mathcal{E}_2 : \sum_{e \in \mathcal{E}_2 \cap \mathcal{A}} \omega(e) &= \sum_{e \in \{e_2, e_3\}} \omega(e) = 2 + 3 = 5. \end{aligned} \tag{4.9}$$

Hence, with respect to maximization $\mathcal{E}_2$ is the maximum for the objective functional in (4.8).

Please note, that the domain in (4.8) is based on the varying set $\mathcal{E}'$ of the sum. Subsequently, the objective functional is not linear. Hence, we need to expand the constraints in (4.5) in two separate constraints

$$\begin{aligned} l \leq \Pi(j) - \Pi(i) - z_e \cdot T& \\ \Pi(j) - \Pi(i) - z_e \cdot T &\leq u. \end{aligned} \tag{4.10}$$

Then, we use the well-established big-M technique [BT05] for each decisional constraint $e \in \mathcal{A}$ with a binary variable $x_e \in \{0, 1\}$ such that

$$\begin{aligned} l \leq \Pi(j) - \Pi(i) - z_e \cdot T + M \cdot \bar{x}_e& \\ \Pi(j) - \Pi(i) - z_e \cdot T - M \cdot \bar{x}_e &\leq u. \end{aligned}$$

with $M \in \mathbb{R}, M \gg 0$ being a large number. We could easily calculate smaller upper bounds to M such that the MIP solvers have better numerical stability, however, this will be out of scope for this work. The binary variable x_e has the semantical meaning that if $x_e = 1$ the constraint e must hold under the schedule. Finally, we can establish the whole MIP formulation such that

$$
\begin{aligned}
&\sum_{e\in\mathcal{A}} \omega(e)x_e \to \max \\
&\text{subject to} \\
&\quad l \le \Pi(j) - \Pi(i) - z_e \cdot T + M \cdot \bar{x}_e \\
&\quad \Pi(j) - \Pi(i) - z_e \cdot T - M \cdot \bar{x}_e \le u, \quad \forall e = (i, j, [l,u]_T) \in \mathcal{A} \\
&\quad l \le \Pi(j) - \Pi(i) - z_e \cdot T \le u, \quad \forall e = (i, j, [l,u]_T) \in \mathcal{E} \setminus \mathcal{A} \\
&\quad \Pi(j), \Pi(i) \in [0, T-1] \subseteq \mathbb{Z}, \quad \forall i, j \in \mathcal{V} \\
&\quad z_e \in \mathbb{Z}, \quad \forall e \in \mathcal{E} \\
&\quad x_e \in \{0,1\}, \quad \forall e \in \mathcal{A}.
\end{aligned}
\tag{4.11}
$$

Having this model, we can encode it to a propositional formula (corresponding MaxSAT problem) which is given in Section 4.2.2 that can be compared for runtime and primal values.

Example 4.2. Let $\mathcal{D}$ be the DPEN as in Example 4.1. Using the model in (4.11) we get

$$
\begin{aligned}
&x_{e_1} + 2x_{e_2} + 3x_{e_3} \to \max \\
&\text{subject to} \\
&\quad 1 \le \Pi(n_{i+1}) - \Pi(n_i) - z_{e_i} \cdot T + M \cdot \bar{x}_{e_i} \\
&\quad \Pi(n_{i+1}) - \Pi(n_i) - z_{e_i} \cdot T - M \cdot \bar{x}_{e_i} \le 1, \quad \forall e_i = (n_i, n_{i+1}, [l,u]_T) \in \mathcal{A} \\
&\quad 1 \le \Pi(n_1) - \Pi(n_4) - z_{e_4} \cdot T \le 1 \\
&\quad \Pi(n_i) \in [0, 9] \subseteq \mathbb{Z}, \quad \forall i \in \{1, \dots, 4\} \\
&\quad z_{e_i} \in \mathbb{Z}, \quad \forall i \in \{1, \dots, 4\} \\
&\quad x_{e_i} \in \{0,1\}, \quad \forall i \in \{1,2,3\}.
\end{aligned}
$$

Further let the schedule Π for $\mathcal{V}$, $\mathcal{E}_1 \subseteq \mathcal{E}$ and $\mathcal{E}_2 \subseteq \mathcal{E}$ be as in Example 4.1. Then, with $\mathcal{E}' \in \{\mathcal{E}_1, \mathcal{E}_2\}$ and

$$
x_{e_i} = \begin{cases} 1, & e_i \in \mathcal{E}' \\ 0, & e_i \notin \mathcal{E}' \end{cases}
$$

we can imply for the objective functional that

$$
\mathcal{E}_1 : x_{e_1} + 2x_{e_2} + 3x_{e_3} = 1 + 2 \cdot 1 + 3 \cdot 0 = 3,
$$

$$\mathcal{E}_2 : x_{e_1} + 2x_{e_2} + 3x_{e_3} = 0 + 2 \cdot 1 + 3 \cdot 1 = 5$$

which are indeed the same values as in (4.9).

Applying this model to the connection (transfer) optimization, such that as many connections for the passengers in a railway network shall hold, is successfully applied in the software system TAKT.

Extending the model to set of constraints[3] can be achieved in two ways. Firstly, we can simply define a set of set of constraints $\mathcal{T} \subset \mathcal{P}(\mathcal{E})$ with each set of constraints in $\mathcal{T}$ to be pairwise disjoint. Then, define a weight mapping ω for $\mathcal{T}$ which results in the objective function for the indexed binary variables x

$$\sum_{\mathcal{C} \in \mathcal{T}} \omega(\mathcal{C}) x_{\mathcal{C}} \to \max .$$

A set of constraints is domain dependent and could for example represent a geographic part of the network or parts of train paths with respect to their constraints. Since train paths in PENs are represented as paths of nodes connected with constraints which represent driving and stopping times, we can use the tool of flow graphs as well for sets of nodes and thus its corresponding constraints. Then, we need simply to maximize the FGs in $\mathcal{S}$ for a corresponding FDPEN. However, we either have to adapt the definition for valid schedules in Definition 3.17 or in each FG we need a dummy flow edge that is not connected to the remaining network and a dummy periodic event that has no constraints to the remaining PEN. Then, we can use the same definition and can optimize the to be used flow graphs by excluding the dummy flow edges in the objective functional. Each of which can be combined for example with the duplicated chain path approach in Section 4.1.3 such that not just the total driving time is minimized but the to be inserted amount of train paths as well which has been already pre-analyzed for non-periodic freight train paths [KGO15].

4.1.3 Optimization in Periodic Event Networks with Flows

Optimizing FDPENs – respectively its applications – is based on minimizing the sum of slacks and flow edge weights on the used paths. In public railway timetabling the sum of slacks and flow edge weights correspond to the sum of stopping times in each station and the driving times for each sub train path, respectively, for the to be inserted train path

[3]as already introduced for local conflicts [Gro12b]

that consists of a route and a timetable. This results in the *minimization of the total travel time* for all train paths[4].

Consequently, for an FG $K = (R, F)$ we need to assign to each flow edge $e \in F$ a weight $\gamma(e)$. For convenience, we may use its corresponding periodic event with the label function such that $\gamma(e) = \gamma(l(e))$ for an FDPEN $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$. As presented above, in railway applications, this weight represents the driving time from a station to its consecutive station.

In the encoding of FGs in Definition 3.37 we used the propositional (binary) decision variables in

$$U = \{s_n \in \mathcal{R} \mid n \in \mathcal{H} \cap \mathit{range}(l)\} \tag{4.12}$$

for a set of decision nodes $\mathcal{H}$. Likewise, we use these variables within the domain $\{0, 1\}$. Subsequently, we can add this structure to the to be minimized objective functional

$$\sum_{n \in \mathcal{H}} \gamma(n) s_n.$$

For convenience, since the label function l is bijective, we equivalently use the indexes for a flow edge e and its corresponding labeled periodic event $l(e)$ with

$$s_{l(e)} \equiv s_e$$

such that both terms are even isomorphic.

Furthermore, as already defined in (4.3) respectively (4.6) in Section 4.1.1 we need the slack time for each constraint $e = (i, j, [l, u]_T)$ with

$$\Pi(j) - \Pi(i) - l - z_e T. \tag{4.13}$$

With respect to FDPENs we have to enhance (4.13) by the propositional decision variables in U as in (4.12) for each $e = (i, j, [l, u]_T)$ such that

$$(\Pi(j) - \Pi(i) - l - z_e T) s_i s_j$$

has to be minimized, because the slack shall only be used if $s_i = s_j = 1$. Subsequently,

[4]respectively for all relations between its sources and destinations

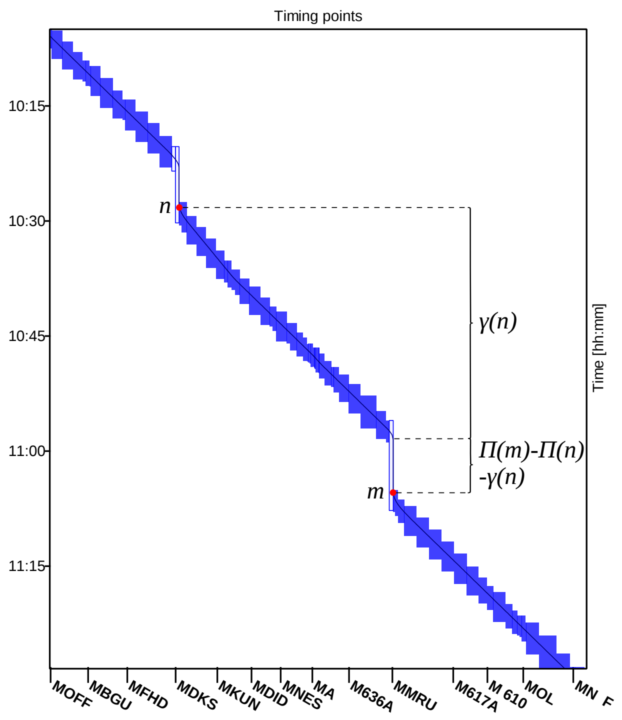


Figure 4.5: Driving time $\gamma(n)$ and slack between n and m for a rail freight train path.

this results in the objective functional for the set of to be minimized edges[5]

$$\mathcal{E}_\gamma := \{e = (i, j, c) \in \mathcal{E} \mid i, j \in U\} \tag{4.14}$$

such that

$$\sum_{n \in \mathcal{H}} \gamma(n) s_n + \sum_{e=(i,j,[l,u]_T) \in \mathcal{E}_\gamma} (\Pi(j) - \Pi(i) - l - z_e T) s_i s_j \to \min . \tag{4.15}$$

Figure 4.5 visualizes an rail freight train path giving examples for the weight $\gamma(n)$ and possible slack according to (4.13) for the periodic events n and m. Because the driving dynamics is fixed, $\gamma(n)$ is a fixed value (constant).

In the previous Section 4.1.2 we used the big-M method for activating each constraint depending on its decisional variables corresponding to its constraints. Likewise, we apply this for each constraint $e = (i, j, [l, u]_T) \in \mathcal{E}_\gamma$ with the binary decision variables $s_i, s_j \in U$ such that

$$\begin{aligned} l \leq \Pi(j) - \Pi(i) - z_e T + M\bar{s}_i + M\bar{s}_j \\ \Pi(j) - \Pi(i) - z_e T - M\bar{s}_i - M\bar{s}_j \leq u. \end{aligned} \tag{4.16}$$

[5]more precisely it is the slack that shall be minimized for each constraint

Again, we could bound M according to the period T which will be skipped here. Yet, we leave this number as "large" such that the constraints are satisfied if $s_j = 0$ or $s_i = 0$. Combining the objective functional in (4.15) subject to the PESP constraints as in (4.10) and (4.16) result in the following definition.

Definition 4.3 (Model FDPEN)**.** *Let $\mathcal{S}$ be a set of FGs and let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$. Then,*

$$\sum_{n \in \mathcal{H}} \gamma(n) s_n + \sum_{e=(i,j,[l,u]_T) \in \mathcal{E}_\gamma} (\Pi(j) - \Pi(i) - l - z_e T) s_i s_j \to \min \tag{4.17}$$

subject to

$$\sum_{e \in O_K(S_K)} s_e = \sum_{e \in I_K(D_K)} s_e = 1, \quad \forall K \in \mathcal{S} \tag{4.18}$$

$$\sum_{e \in I(n)} s_e - \sum_{e \in O(n)} s_e = 0, \quad \forall K \in \mathcal{S}, \forall n \in R \setminus (S_K \cup D_K) \tag{4.19}$$

$$l \leq \Pi(j) - \Pi(i) - z_e T + M\bar{s}_i + M\bar{s}_j \tag{4.20}$$

$$\Pi(j) - \Pi(i) - z_e T - M\bar{s}_i - M\bar{s}_j \leq u, \quad \forall e = (i, j, [l, u]_T) \in \mathcal{E}_\gamma \tag{4.21}$$

$$l \leq \Pi(j) - \Pi(i) - z_e \cdot T \leq u, \quad \forall e = (i, j, [l, u]_T) \in \mathcal{E} \setminus \mathcal{E}_\gamma$$

$$\Pi(j), \Pi(i) \in [0, T-1] \subseteq \mathbb{Z}, \quad \forall i, j \in \mathcal{V}$$

$$z_e \in \mathbb{Z}, \quad \forall e \in \mathcal{E}$$

$$s_n \in \{0, 1\}, \quad \forall n \in \mathcal{H}.$$

is the model for minimizing the sum of slacks and flow edge weights for all FGs in $\mathcal{S}$.

In Definition 3.17, we declared that each path needs exactly one source, exactly one destination and the flow conservation has to be satisfied, which is stated in (4.18) and (4.19), respectively. Since (4.15) is cubic – and thus, (4.17) as well – we have to linearize it in order to pass the corresponding MIP instance to an according solver. Hence, the following theorem gives an equivalent formulation with respect to the to be minimized objective functional which covers a bijection of the solution space for the relevant constraints.

Theorem 4.4 (Linearization)**.** *Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$. Then, the objective functional in* (4.17) *subject to* (4.21) *and the variable* *definitions is equivalent to*

$$\sum_{n\in\mathcal{H}} \gamma(n)s_n + \sum_{e\in\mathcal{E}_\gamma} \delta_e \to \min \tag{4.22}$$

subject to

$$\Pi(j) - \Pi(i) - z_eT - M\bar{s}_i - M\bar{s}_j - \delta_e \leq l, \quad \forall e = (i,j,[l,u]_T) \in \mathcal{E}_\gamma \tag{4.23}$$
$$\delta_e - Ms_i \leq 0, \quad \forall e = (i,j,[l,u]_T) \in \mathcal{E}_\gamma \tag{4.24}$$
$$\delta_e - Ms_j \leq 0, \quad \forall e = (i,j,[l,u]_T) \in \mathcal{E}_\gamma \tag{4.25}$$
$$\Pi(j), \Pi(i) \in [0, T-1] \subseteq \mathbb{Z}, \quad \forall i,j \in \mathcal{V}$$
$$\delta_e \in [0, u-l] \subseteq \mathbb{Z}, \quad \forall e = (i,j,[l,u]_T) \in \mathcal{E}_\gamma$$
$$z_e \in \mathbb{Z}, \quad \forall e \in \mathcal{E}$$
$$s_n \in \{0,1\}, \quad \forall n \in \mathcal{H}.$$

Proof. For each constraint $e = (i,j,[l,u]_T) \in \mathcal{E}_\gamma$ let $c := u - l$ be constant. Then, it holds for (4.21)

$$\begin{aligned} &\Pi(j) - \Pi(i) - z_eT - M\bar{s}_i - M\bar{s}_j \leq u \\ \overset{u=c+l}{\Leftrightarrow}\ &\Pi(j) - \Pi(i) - z_eT - M\bar{s}_i - M\bar{s}_j \leq c + l \\ \Leftrightarrow\ &\Pi(j) - \Pi(i) - z_eT - M\bar{s}_i - M\bar{s}_j - c \leq l \end{aligned} \tag{4.26}$$

Since the first term in the objective functionals are equal, it remains to show that the second terms

$$\sum_{e=(i,j,[l,u]_T)\in\mathcal{E}_\gamma} (\Pi(j) - \Pi(i) - l - z_eT)s_is_j$$

and

$$\sum_{e\in\mathcal{E}_\gamma} \delta_e$$

are equal and thus, with respect to an arbitrary but fixed constraint $e = (i,j,[l,u]_T)$

$$x = (\Pi(j) - \Pi(i) - l - z_eT)s_is_j \tag{4.27}$$

and

$$y = \delta_e \tag{4.28}$$

are equal ($x = y$) subject to its respective constraints.

We must distinguish two cases: 1. Without loss of generality let $s_i = 0$ (could be j as well). Then, in (4.27) it follows $x = 0$, since all terms are factors. Furthermore, in (4.28) it follows $y = 0$, because with (4.24) it can be implied

$$\delta_e \leq M s_i = M \cdot 0 = 0$$

and consequently it holds $x = y$. Additionally, (4.21) and (4.23) hold in both cases since $s_i = 0$ and hence, $M > u$ and $M > l$, respectively.
2. it remains to show for $s_i = s_j = 1$ it holds $x = y$. We can conclude for (4.27) that $x = \Pi(j) - \Pi(i) - l - z_e T$. On the one hand, we have to show that both constraints (4.21) and (4.23) are equivalent for all variable assignments and secondly, that the terms x and y are equal. With $c = \delta_e$ we see directly that (4.26) equals (4.23). This can be done, because δ_e is a free parameter within its bound since $s_i = s_j = 1$ and thus, (4.24) and (4.25) do not restrict δ_e at all which implies it can be freely chosen in $[0, u - l]$ whose upper bound equals c. Since y shall be minimized, we can construct in the optimal case ($y \to \min$) an minimal δ_e that solves the model in Theorem 4.4. Since δ_e is minimal, we can conclude for (4.23)

$$\Pi(j) - \Pi(i) - z_e T - M\bar{s}_i - M\bar{s}_j - \delta_e = l.$$

This can be done, because it can be simply, indirectly shown that an $\delta'_e > \delta_e$ (with $\Pi(j) - \Pi(i) - z_e T - M\bar{s}_i - M\bar{s}_j - \delta'_e \leq l$) would lead to a suboptimal case. Additionally, because $s_i = s_j = 1$ we know it is equivalent to

$$\Pi(j) - \Pi(i) - z_e T - \delta_e = l$$

which can be further transformed to

$$\Pi(j) - \Pi(i) - z_e T - l = \delta_e$$

and finally it can be concluded that

$$x = y.$$

Since e is chosen arbitrary it holds for any constraint and subsequently, both objective functionals are equal with respect to any fulfilling variable assignment. □

Since no variables are multiplied with other variables in (4.22) and in the modified and new constraints, the model is *linear*. The remaining constraints in Definition 4.3 that are not covered in Theorem 4.4 do not influence the linearization and thus, can simply be added to the model in the theorem maintaining equivalence.

The variable δ_e has the semantic influence that it equals 0 if one of the corresponding labeled nodes (i or j) is not part of the schedule ((4.24), (4.25)), or equals the slack $(\Pi(j) - \Pi(i) - l - z_e T)$ otherwise. The slack is achieved by the constraint (4.23) as shown in the proof of the theorem. Hence, it combines the cubic part of the objective function in one variable.

Extending the model for decisional FGs allows to (for example) maximize the number of to be inserted duplicated FGs[6]. Therefore, we need new decisional variables u_K for each FG K. Then, we can simply maximize the number of to be inserted FGs in $\mathcal{S}$ such that

$$\sum_{K \in \mathcal{S}} u_K \to \max \tag{4.29}$$

which can be extended to constant weights as well. In the MIP model this can simply be achieved by the big-M method. The appropriate MaxSAT approach will shortly be discussed in Section 4.2.3. This method is already successfully applied for non-periodic train paths [KGO15].

Distinct and Duplicated Chain Paths

Both applications refer to different path choices which are represented as flow edges and activity times which represent the PESP constraints (see Section 3.3.1 and Section 3.3.2). Hence, the weights of the flow edges represent the driving times and the PESP constraints represent waiting and transfer times; as well as headway times which are not regarded in the objective functional. This results in a direct application of the objective functional in (4.15). In case of applying this to real-world scenarios we should use the linearized objective functional in Theorem 4.4 or the to be introduced MaxSAT approach in Section 4.2.3.

Non-connected Flow Graphs

If the departure times for each periodic train path is already given and fixed, and the allocation of the tracks is left, we can skip the slack time as in (4.13). This results in a simplified objective functional such that

$$\sum_{n \in \mathcal{H}} \gamma(n) s_n \to \min .$$

[6] e. g., rail freight train paths

In case of track allocation, each flow edge (labeled node) is weighted by domain-specific priorities and driving times [WKO15].

4.2 Encoding Objectives as Boolean Optimization Problems

This section covers a waste majority of MaxSAT encodings of the presented objective functionals in Section 4.1. Since both problems, MIP and MaxSAT, are optimization problems with conjuncted linear constraints there is often a straight forward way to encode the respective methods.

4.2.1 Minimization of Weighted Slacks

In Section 4.1.1 the linear objective functional is presented for minimizing weighted slacks for constraints. Here, we want to optimize the same structure, yet, with the MaxSAT model that needs a propositional formula in WCNF. Firstly, we need to introduce propositional decision variables for the maximum slack

$$\theta_e := u - l \tag{4.30}$$

of each constraint $e = (n, m, [l, u]_T) \in \mathcal{E}_\omega$ with $\mathcal{E}_\omega$ as in (4.2) such that

$$p_{e,i} \in \{f, t\},\ i \in \{0, \dots, \theta_e\}$$

with the semantics whether the current used slack $\sigma(e, \Pi)$ as in (4.4) is less or equal than i such that

$$p_{e,i} :\Leftrightarrow \Pi(m) - \Pi(n) \in [l, l+i]_T,\ i \in \{0, \dots, \theta_e\}.$$

for a schedule Π. Thus, $p_{e,0}$ is equivalent to the respective constraint using no slack $\sigma(e, \Pi) = 0$. Secondly, we need to modify the encoding function *enc_con* as in Definition 2.44 which encodes each given constraint $e = (n, m, [l, u]_T) \in \mathcal{E}_\omega$ into a propositional formula such that

$$\begin{aligned} enc\_con_\theta(e) = {} & enc\_con(n, m, [l, l+\theta_e]_T) \\ & \wedge \bigwedge_{i \in \{0,\dots,\theta_e\}} p_{e,i} \wedge \left(\neg p_{e,i} \vee enc\_con((n, m, [l, l+i]_T)\right) \end{aligned} \tag{4.31}$$

which conditionally encodes each upper bound slack $(l + i)$ separately. Hence, by setting each decisional variable $p_{e,i}$ to f, we have the base-encoded to be optimized PEN. This encoding will be done for all constraints.

In order to minimize the objective function as in (4.3) we have to use the decisional propositional variables $p_{e,i}$ such that

$$\sum_{e\in\mathcal{E}_\omega}\sum_{i=0}^{\theta_e}\omega(e)\cdot p_{e,i}\to\max \tag{4.32}$$

which maximizes the weighted number of the decisional variables. Subsequently, the weighted slacks are minimized.

Applying (4.32) subject to (4.31) conjunctively connected to $\Omega_\mathcal{N}$ as in (2.7) will not be accepted syntactically[7] by MaxSAT solvers. Hence, we have to use the weighted formula for a weighted partial MaxSAT solver as in Definition 2.21.

First of all, we need to encode each periodic event in $\mathcal{V}$ with the weight ∞ because each potential must be defined uniquely for the resulting schedule. Hence, we modify $\Omega_\mathcal{N}$ such that for each encoded periodic event $n\in\mathcal{V}$

$$\mathit{enc\_max}(n) := (\infty, \neg q_{n,-1}) \wedge (\infty, q_{n,T-1}) \bigwedge_{i\in[0,T-1]} (\infty, \neg q_{n,i-1}\vee q_{n,i})$$

we can conjunctively connect each encoded periodic event with

$$\Omega_\mathcal{N}^{max} := \bigwedge_{n\in\mathcal{V}} \mathit{enc\_max}(n) \tag{4.33}$$

Likewise, each constraint $e\in\mathcal{E}$ needs a weight $\omega=\infty$ such that

$$\mathit{enc\_con\_max}(e) := \bigwedge_{r\in\zeta(a(e))} (\infty, \mathit{enc\_rec}(r)) \tag{4.34}$$

and appropriately applied to $\mathit{enc\_con}_\theta(e)$ for each to be minimized constraint $e\in\mathcal{E}_\omega$ with $e=(n,m,[l,u]_T)$ yields

$$\begin{aligned}\mathit{enc\_con\_max}_\theta(e) := {} & \mathit{enc\_con\_max}(n,m,[l,l+\theta_e]_T)\\ & \wedge \bigwedge_{i\in\{0,\dots,\theta_e\}} \Big((\omega(e), p_{e,i})\end{aligned} \tag{4.35}$$

[7]this is reasoned because all previous encodings map to a propositional formula in CNF but here we need a propositional formula in WCNF

$$\wedge \bigwedge_{r\in\zeta([l,l+i]_T)} (\infty, \neg p_{e,i} \vee enc\_rec(r))\Big)$$

which simply adds to each clause the weight ∞, excluding the clauses in (4.35) which are attached to the corresponding constraint's weight $\omega(e)$.

Consequently, using (4.32) for a PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$, results in the encoding

$$enc\_pen\_max(\mathcal{N}) := \Omega_{\mathcal{N}}^{max} \wedge \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\omega} enc\_con\_max(e) \wedge \bigwedge_{e\in\mathcal{E}_\omega} enc\_con\_max_\theta(e).$$

The not to be minimized constraints $e \in \mathcal{E} \setminus \mathcal{E}_\omega$ are all weighted in its respective clauses with ∞ such that they must hold for any valid schedule. However, the to be optimized constraints in $\mathcal{E}_\omega$ with its corresponding decision variables are encoded with the weight $\omega(e)$. Decoding a satisfying interpretation has no necessity to be changed and thus, ξ as in (2.8) can be used directly.

Comparing the values of the objective functional for the interpretation of a weighted formula as in Definition 2.21 and the linear objective as in (4.3) for an to be optimized PEN $\mathcal{N}$, an interpretation J and its corresponding schedule Π with (2.8) can be achieved (without proof) by

$$\sum_{e\in\mathcal{E}_\omega}\sum_{i=1}^{\theta_e} \omega(e)\cdot p_{e,i} = \sum_{e\in\mathcal{E}_\omega} \omega(e)\cdot\sigma(e,\Pi)$$

in case of a global optimum. Hence, we can compare the objective values for both approaches.

Finally, state-of-the-art MaxSAT solvers can be applied, that efficiently optimize the objective functional in (4.32) subject to the encoded PESP instance that are reflecting to be optimized PENs. Results on large and complex networks tend to need a lot of computation time, however, for medium-sized instances they already suggesting promising solutions and solving time as presented in Section 5.3.

Extending the set of constraints e of the form $[l,u]_T$ to the general structure $\mathcal{Z}$ as Definition 2.29 can be done with the advanced encoding in Section 2.3.3 by either setting l_e to the smallest value in the interval $[0, T-1]$ such that[8]

$$l_e := \min((\mathcal{Z}\cap[0,T-1])\cup\{0\})$$

or giving each l_e for the respective constraint separately. Then, we modify the encod-

[8]we set the minimum to 0 if $\mathcal{Z}$ is empty to maintain soundness and infeasibility

ing $enc\_con\_max_\theta$ for $\mathcal{Z}' := [l_e, l_e]_T$ to

$$enc\_con\_max(e) \wedge \bigwedge_{i \in \mathcal{Z} \cap [l_e, l_e+T-1], \mathcal{Z}' := \mathcal{Z}' \cup [i,i]_T} \Big((\omega(e), p_{e,i}) \wedge \bigwedge_{r \in \zeta(n,m,\mathcal{Z}')} (\infty, \neg p_{e,i} \vee enc\_rec(r)) \Big)$$

with ζ as in (2.17) which simply modifies $i \in \{0, \ldots, \theta_e\}$ to $i \in \mathcal{Z} \cap [l_e, l_e + T - 1], \mathcal{Z}' := \mathcal{Z}' \cup [i,i]_T$ that iterates over all possible slacks for the constraint.

4.2.2 Satisfiable Constraint Maximization

In Section 4.1.2 the maximization of number of constraints that hold under a schedule is presented and transformed into a corresponding MIP model. Likewise, we are going to encode the problem into a propositional formula in WCNF.[9]

For a DPEN $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ ($\mathcal{H} = \emptyset$) with the set of edges in $\mathcal{A}$ as in (4.7) q_e we introduce propositional decision variables q_e ($e \in \mathcal{E} \setminus \mathcal{A}$), which means semantically for a model J and the corresponding schedule Π (via ξ) as in (2.8)

$$J \models q_e \Rightarrow \Pi \models e.$$

Achieving the maximization as in (4.8) we have to use a propositional formula in WCNF for a weight function ω (see (4.7))

$$\mathcal{F} := \Omega_{\mathcal{N}}^{max} \wedge \bigwedge_{e \in \mathcal{E} \setminus \mathcal{A}} enc\_con\_max(e) \wedge \bigwedge_{e \in \mathcal{A}} \Big((\omega(e), q_e) \wedge \bigwedge_{r \in \zeta(a(e))} (\infty, \neg q_e \vee enc\_rec(r)) \Big) \tag{4.36}$$

with enc_con_max as in (4.34). All constraints that are not in the set of decisional constraints must hold for any satisfying interpretation (enc_con_max). If an interpretation J for the corresponding MaxSAT problem is found the maximum sum of weighted number of decisional variables q_e is set and thus, the corresponding constraints in $\mathcal{A}$ hold, because for an extracted schedule Π it holds

$$\begin{aligned} \forall e \in \mathcal{A} : J \models q_e &\overset{J \models \mathcal{F}}{\Rightarrow} J \models \bigwedge_{r \in \zeta(a(e))} (\infty, \neg q_e \vee enc\_rec(r)) \\ &\overset{q_e^J = t}{\Rightarrow} J \models \bigwedge_{r \in \zeta(a(e))} (\infty, enc\_rec(r)) \\ &\overset{\text{Lem 2.24}}{\Rightarrow} J \models \bigwedge_{r \in \zeta(a(e))} enc\_rec(r) \end{aligned}$$

[9]Hence, we are going to optimize the given problems with a state-of-the-art MaxSAT solver.

$$\overset{\text{Def 2.44}}{\Rightarrow} J \models enc\_con(e)$$
$$\overset{\text{Thm 2.47}}{\Rightarrow} \Pi \models e$$

which represents the soundness proof.

Example 4.5. Let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, 10, \emptyset, \mathcal{A})$ be a DPEN as in Example 4.1. Then, using the encoding as in (4.36) we get

$$\Omega_{\mathcal{N}}^{max} = \bigwedge_{n \in \mathcal{V}} enc\_max(n) = enc\_max(n_1) \wedge \ldots \wedge enc\_max(n_4)$$

$$\bigwedge_{e \in \mathcal{E} \setminus \mathcal{A}} enc\_con\_max(e) = \bigwedge_{e \in \{e_4\}} enc\_con\_max(e) = enc\_con\_max(e_4)$$

$$\begin{aligned}
&\bigwedge_{e \in \mathcal{A}} \Big((\omega(e), q_e) \bigwedge_{r \in \zeta(a(e))} (\infty, \neg q_e \vee enc\_rec(r)) \Big) \\
&= \bigwedge_{e \in \{e_1, e_2, e_3\}} \Big((\omega(e), q_e) \bigwedge_{r \in \zeta(a(e))} (\infty, \neg q_e \vee enc\_rec(r)) \Big) \\
&= \Big((1, q_{e_1}) \bigwedge_{r \in \zeta(a(e_1))} (\infty, \neg q_{e_1} \vee enc\_rec(r)) \Big) \\
&\quad \wedge \Big((2, q_{e_2}) \bigwedge_{r \in \zeta(a(e_2))} (\infty, \neg q_{e_2} \vee enc\_rec(r)) \Big) \\
&\quad \wedge \Big((3, q_{e_3}) \bigwedge_{r \in \zeta(a(e_3))} (\infty, \neg q_{e_3} \vee enc\_rec(r)) \Big).
\end{aligned}$$

Both methods – which are the MIP and MaxSAT approach – are successfully applied in the software system TAKT for large-scaled instances.

4.2.3 Optimization in Periodic Event Networks with Flows

In Section 4.2.1 we introduced propositional decision variables for minimizing slacks. Here, we will deduce an approach for minimizing the objective functional in (4.15)[10] with a sound[11] MaxSAT approach. Please note, that both the satisfiable constraint maximization and the minimization of the objective functional for FDPENs can simply be combined, yet, will not be part of this section.

Let $\mathcal{S}$ be a set of FGs and $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$. In comprehension, we have available the propositional variables for each decision node s_n $(n \in \mathcal{H})$ and

[10]respectively (4.22)
[11]and even complete

the propositional slack variables $p_{e,i}$ ($e \in \mathcal{E}_\gamma, i \in \{1, \ldots, \theta_e\}$) with $\mathcal{E}_\gamma$ as in (4.14) for a weight function γ. Furthermore, for a label function l we have the base encoding for FDPENs *enc_fdpen*$_l$ with Definition 3.40. Since *enc_fdpen*$_l$ is in CNF and not in WCNF, we need a transformation for each sub encoding as already partly applied in Section 4.2.1.

Consequently, we can already use $\Omega^{max}_{\mathcal{N}}$ as in (4.33) for the MaxSAT node encoding and *enc_con_max* as in (4.34) for the hard constraints. We need to modify slightly the encoding for flow graphs *enc_fg*$_l$ as in Definition 3.37 with the weight ∞ for an FG $K \in \mathcal{S}$ such that

$$\begin{aligned}
\mathit{enc\_fg\_max}_l(K) := {} & \Big(\infty, \bigvee_{n \in S_K} \bigvee_{e \in O(n)} s_{l(e)}\Big) \\
& \wedge \Big(\bigwedge_{n \in S_K} \bigwedge_{e \in O(n)} \bigwedge_{m \in S_K} \bigwedge_{e' \in O(m), e \neq e'} (\infty, \neg s_{l(e)} \vee \neg s_{l(e')})\Big) \\
& \wedge \Big(\infty, \bigvee_{n \in D_K} \bigvee_{e \in I(n)} s_{l(e)}\Big) \\
& \wedge \Big(\bigwedge_{n \in D_K} \bigwedge_{e \in I(n)} \bigwedge_{m \in D_K} \bigwedge_{e' \in I(m), e \neq e'} (\infty, \neg s_{l(e)} \vee \neg s_{l(e')})\Big) \\
& \wedge \Big(\bigwedge_{n \in R \setminus (S_K \cup D_K)} \bigwedge_{e \in I(n)} \bigwedge_{e' \in I(n), e \neq e'} (\infty, \neg s_{l(e)} \vee \neg s_{l(e')})\Big) \\
& \wedge \Big(\bigwedge_{n \in R \setminus (S_K \cup D_K)} \bigwedge_{e \in O(n)} \bigwedge_{e' \in O(n), e \neq e'} (\infty, \neg s_{l(e)} \vee \neg s_{l(e')})\Big) \\
& \wedge \Big(\bigwedge_{(n,m) \in F, m \notin D_K} (\infty, \neg s_{l(n,m)} \bigvee_{e \in O_K(m)} s_{l(e)})\Big),
\end{aligned}$$

because all flow graphs have to fulfill the conditions for a valid schedule as in Definition 3.17. Since each clause in *enc_fg_max*$_l(K)$ is weighted, it is a propositional formula in WCNF. For the minimization of weighted slacks we have to modify the encoding function *enc_con_max*$_\theta$ in (4.35). We used in Definition 3.31 the encoding function *enc_dec* for decision nodes and constraints. Likewise, combining this approach with the MaxSAT approach in Section 4.2.1 for the minimization of slacks for the to be optimized constraints in $\mathcal{E}_\gamma$ we get the appropriate encoding for constraints with the following definition.

Definition 4.6 (MaxSAT Encode Decision Edges). *Let $e = (i, j, \mathcal{Z}_e) \in \mathcal{E}$ be a constraint, $\mathcal{A} \subseteq \mathcal{E}$ be a set of constraints and $\mathcal{H} \subseteq \mathcal{V}$ be a set of nodes. Further, let $S = \{r_e \in \mathcal{R} \mid e \in \mathcal{A}\} \cup \{s_n \in \mathcal{R} \mid n \in \{i, j\} : n \in \mathcal{H}\}$ be the set of propositional decision variables.*

Then

$$enc\_dec\_max : \mathcal{E} \to \mathcal{M}(\mathcal{R})$$
$$e \mapsto \bigvee_{p \in S} (\neg p) \vee \begin{cases} enc\_con\_max_\theta(e), & e \in \mathcal{E}_\gamma \\ enc\_con\_max(e), & e \notin \mathcal{E}_\gamma \end{cases}$$

is the MaxSAT decision order encoding mapping *of the constraint e with respect to nodes i, j.*

Please note that as it stands *enc_dec_max* does not map to a propositional formula in WCNF, since it is a disjunction with $enc\_con\_max_\theta$ and $enc\_con\_max_\theta$. However, we can simply form an semantically equivalent formula with De Morgan's laws such that

$$\bigvee_{p \in S} (\neg p) \vee enc\_con\_max(e) \equiv \bigwedge_{r \in \zeta(a(e))} (\infty, \bigvee_{p \in S} (\neg p) \vee enc\_rec(r)), \tag{4.37}$$

$$\begin{aligned} \bigvee_{p \in S} (\neg p) \vee enc\_con\_max_\theta(e) \overset{(4.37)}{\equiv} & \big(\bigvee_{p \in S} (\neg p) \vee enc\_con\_max((n, m, [l, l + \theta_e]_T)\big) \\ & \wedge \bigwedge_{i \in \{0,\ldots,\theta_e\}} \Big((\omega(e), p_{e,i}) \qquad (4.38) \\ & \wedge \bigwedge_{r \in \zeta([l, l+i]_T)} (\infty, \bigvee_{p \in S} (\neg p) \vee \neg p_{e,i} \vee enc\_rec(r)) \Big). \qquad (4.39) \end{aligned}$$

Consequently, it is in WCNF and can be applied to an according MaxSAT solver. Please note, that in (4.38) the decisional part $(\bigvee_{p \in S} (\neg p))$ is omitted, because it is subsumed by (4.39) because all $p_{e,i}$ are free in case one p in S is mapped to false (f). In case of minimization of waiting time in railway timetabling, we will without loss of generality assume that in some cases $\omega(e) = 1$.[12]

Furthermore, we need to optimize the first part of the objective functional in (4.17)

$$\sum_{n \in \mathcal{H}} \gamma(n) s_n \to \min .$$

This goal is achieved[13] by extending $\Omega_{\mathcal{N}}^{max}$ such that it is conjuncted with

$$\bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n).$$

[12]for example in the proof of Theorem 4.8
[13]which is shown in Theorem 4.8

Combining all of which leads to the following definition of encoding FDPENs with respect to the objective functional in (4.17).

Definition 4.7 (MaxSAT Encoding FDPEN)**.** *Let $\mathcal{S}$ be a set of FGs. Further let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and l be a label function into the set of nodes $\mathcal{H}$. Then*

$$\begin{aligned} &\mathit{enc\_fdpen\_max}_l : \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{S}) \to \mathcal{M}(\mathcal{R}) \\ &\qquad (\mathcal{D}, \mathcal{S}) \mapsto \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg\_max}_l(K) \wedge \Omega_{\mathcal{N}}^{max} && (4.40) \\ &\qquad\qquad \wedge \bigwedge_{e \in \mathcal{E}} \mathit{enc\_dec\_max}(e) \wedge \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n) && (4.41) \end{aligned}$$

is the FDPESP MaxSAT encoding *of $\mathcal{D}$ for $\mathcal{S}$.*

The definition conjuncts each previously introduced part for the encoding which combines the FG and node (event) encoding in (4.40) and the hard and to be optimized constraints in (4.41). We used in (2.8) in Section 2.3.1 the function ξ to extract the values for the encoded events. Likewise, since the node encoding is not changed[14], we apply this here for extracting a schedule.

The following theorem gives the soundness and completeness to the approach in Section 4.1.3 and thus, the objective functional can be solved with an appropriate MaxSAT solver equivalently.

Theorem 4.8 (Soundness Completeness MaxSAT FDPESP Encoding)**.**
Let $\mathcal{S}$ be a set of FGs. Further let $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ be an FDPEN for $\mathcal{S}$ and l be a label function into the set of nodes $\mathcal{H}$, $\mathcal{V}' \subseteq \mathcal{V}$ be a set of nodes and let $\mathcal{E}' \subseteq \mathcal{E}$ be a set of constraints of $\mathcal{D}$. With

$$\mathcal{F} := \mathit{enc\_fdpen\_max}_l(\mathcal{D}, \mathcal{S}) \in \mathcal{M}(\mathcal{R})$$

being the encoded propositional formula in WCNF of $\mathcal{D}$ for $\mathcal{S}$. It holds

$$\exists J : J \models \mathcal{F} \Leftrightarrow \exists \Pi : \Pi \textit{ minimizes } g \textit{ and is valid for } \mathcal{D},\ \mathcal{E}' \textit{ under } \mathcal{S}$$

with J being an interpretation, g the objective functional in (4.15) *and Π a schedule of $\mathcal{V}'$.*

[14] besides the prepended ∞

Proof. For convenience, in order to prevent exploding the already long proof, we will use in this proof "conjunction" and "comma" quite loosely. They will have in most cases semantically equivalent meanings and will be given explicitly.

$$\begin{aligned}
&\exists J : J \models \mathcal{F} \\
\overset{\text{Def 4.7}}{\Leftrightarrow}\ &\exists J : J \models \Big(\bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg\_max}_l(K) \wedge \Omega_{\mathcal{N}}^{max} \\
&\qquad \wedge \bigwedge_{e \in \mathcal{E}} \mathit{enc\_dec\_max}(e) \wedge \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n)\Big) \\
\overset{\text{Def 2.12}^{15}}{\Leftrightarrow}\ &\exists J : J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg\_max}_l(K), J \models \Omega_{\mathcal{N}}^{max}, \\
&J \models \bigwedge_{e \in \mathcal{E}} \mathit{enc\_dec\_max}(e), J \models \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n) \\
\overset{\text{Lem 2.24}}{\Leftrightarrow}\ &\exists J : J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}}, \\
&J \models \bigwedge_{e \in \mathcal{E}} \mathit{enc\_dec\_max}(e), J \models \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n) \\
\overset{\text{Def 4.6}}{\Leftrightarrow}\ &\exists J : J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}}, \\
&J \models \bigwedge_{e \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}_\theta(e), \\
&J \models \bigwedge_{e \in \mathcal{E} \setminus \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(e), \\
&J \models \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n) \\
\overset{(4.35)}{\Leftrightarrow}\ &\exists J : J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}}, \\
&J \models \bigwedge_{e \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \Big(\mathit{enc\_con\_max}(n, m, [l, l + \theta_e]_T) \wedge \\
&\bigwedge_{i \in \{0, \ldots, \theta_e\}} \big((\omega(e), p_{e,i}) \wedge \bigwedge_{r \in \zeta([l, l+i]_T)} (\infty, \neg p_{e,i} \vee \mathit{enc\_rec}(r))\big)\Big), \\
&J \models \bigwedge_{e \in \mathcal{E} \setminus \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(e), \\
&J \models \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n) \\
\overset{\text{(4.30),Ex 2.15}}{\Leftrightarrow}\ &\exists J : J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},
\end{aligned}$$

$$J \models \bigwedge_{e \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \Big(\bigwedge_{i \in \{0,\dots,\theta_e\}} \big((\omega(e), p_{e,i}) \wedge$$

$$\bigwedge_{r \in \zeta([l,l+i]_T)} (\infty, \neg p_{e,i} \vee \mathit{enc\_rec}(r))\big)\Big),$$

$$J \models \bigwedge_{e=(n,m,[l,u]_T) \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),$$

$$J \models \bigwedge_{e \in \mathcal{E} \setminus \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(e),$$

$$J \models \bigwedge_{n \in \mathcal{H}} (\gamma(n), \neg s_n)$$

$$\overset{\text{Def } 2.23,(2.2)}{\Leftrightarrow} \exists J : \sum_{n \in \mathcal{H}} \gamma(n) s_n \to \min,$$

$$J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},$$

$$J \models \bigwedge_{e \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \Big(\bigwedge_{i \in \{0,\dots,\theta_e\}} \big((\omega(e), p_{e,i}) \wedge$$

$$\bigwedge_{r \in \zeta([l,l+i]_T)} (\infty, \neg p_{e,i} \vee \mathit{enc\_rec}(r))\big)\Big),$$

$$J \models \bigwedge_{e=(n,m,[l,u]_T) \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),$$

$$J \models \bigwedge_{e \in \mathcal{E} \setminus \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(e),$$

$$\overset{\text{Ex } 2.15}{\Leftrightarrow} \exists J : \sum_{n \in \mathcal{H}} \gamma(n) s_n \to \min,$$

$$J \models \bigwedge_{K \in \mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},$$

$$\forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow J \models (\bigwedge_{i \in \{0,\dots,\theta_e\}} \big((\omega(e), p_{e,i}) \wedge$$

$$\bigwedge_{r \in \zeta([l,l+i]_T)} (\infty, \neg p_{e,i} \vee \mathit{enc\_rec}(r))\big))\Big),$$

$$J \models \bigwedge_{e=(n,m,[l,u]_T) \in \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),$$

$$J \models \bigwedge_{e \in \mathcal{E} \setminus \mathcal{E}_\gamma} \bigvee_{p \in S} (\neg p) \vee \mathit{enc\_con\_max}(e),$$

$$\overset{\text{Def } 2.23,(2.2)}{\Leftrightarrow} \exists J : \sum_{n \in \mathcal{H}} \gamma(n) s_n \to \min,$$

$$
\begin{aligned}
& J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},\\
& \forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow \big(\sum_{i=0}^{\theta_e} \omega(e) p_{e,i} \to \max,\\
& \qquad J \models \bigwedge_{i\in\{0,\dots,\theta_e\}} \bigwedge_{r\in\zeta([l,l+i]_T)} (\infty, \neg p_{e,i} \vee \mathit{enc\_rec}(r))\big)\Big),\\
& J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),\\
& J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e),\\
\overset{\text{Def 2.27,maximization}}{\Leftrightarrow}\; & \exists J : \sum_{n\in\mathcal{H}} \gamma(n) s_n \to \min,\\
& J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},\\
& \forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow \big(\exists! i \in \{i,\dots,\theta_e\} :\\
& \sum_{i=0}^{\theta_e} \omega(e) p_{e,i} \to \max, \forall k \geq i :\\
& \qquad J \models \bigwedge_{r\in\zeta([l,l+k]_T)} (\infty, \mathit{enc\_rec}(r))\big)\Big),\\
& J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),\\
& J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e),\\
\overset{\text{Def 2.32,Thm 2.47,(2.8)}}{\Leftrightarrow}\; & \exists J : \Pi = \xi(J), \sum_{n\in\mathcal{H}} \gamma(n) s_n \to \min,\\
& J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},\\
& \forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow \big(\exists! i \in \{0,\dots,\theta_e\} : \forall k \geq i :\\
& \qquad \Pi \models (n, m, [l, l+k]_T),\\
& J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T)\big)\Big),\\
& J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e),\\
\overset{\text{Def 2.27}}{\Leftrightarrow}\; & \exists J : \Pi = \xi(J), \sum_{n\in\mathcal{H}} \gamma(n) s_n \to \min,
\end{aligned}
$$

$$
\begin{aligned}
& J \models \bigwedge_{K\in\mathcal{S}} enc\_fg_l(K), J \models \Omega_{\mathcal{N}}, \\
& \forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow \big(\exists! i \in \{0,\dots,\theta_e\} : \\
& \qquad \Pi \models (n,m,[l,l+i]_T), \Pi \not\models (n,m,[l,l+i-1]_T))\big)\Big) \\
& J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee enc\_con\_max(n,m,[l,u]_T), \\
& J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee enc\_con\_max(e), \\
\overset{\text{minimization}}{\Leftrightarrow}\ & \exists J : \Pi = \xi(J), \sum_{n\in\mathcal{H}} \gamma(n) s_n \to \min, \\
& J \models \bigwedge_{K\in\mathcal{S}} enc\_fg_l(K), J \models \Omega_{\mathcal{N}}, \\
& \forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow \\
& \qquad \exists! \min_{i\in\{0,\dots,\theta_e\}} (\Pi \models (n,m,[l,l+i]_T))\Big) \\
& J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee enc\_con\_max(n,m,[l,u]_T), \\
& J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee enc\_con\_max(e), \\
\overset{\text{Thm } 2.47,(2.8)}{\Leftrightarrow}\ & \exists \Pi : J = \xi^{-1}(\Pi), \sum_{n\in\mathcal{H}} \gamma(n) s_n \to \min, \\
& J \models \bigwedge_{K\in\mathcal{S}} enc\_fg_l(K), J \models \Omega_{\mathcal{N}}, \\
& \forall e \in \mathcal{E}_\gamma : \Big(\forall p \in S, J \models p \Rightarrow \\
& \qquad \Pi(j) - \Pi(i) - l - z_e T \to \min\Big) \\
& J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee enc\_con\_max(n,m,[l,u]_T), \\
& J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee enc\_con\_max(e), \\
\overset{\forall e\in\mathcal{E}_\gamma,\text{implication}}{\Leftrightarrow}\ & \exists \Pi : J = \xi^{-1}(\Pi), \sum_{n\in\mathcal{H}} \gamma(n) s_n \to \min, \\
& J \models \bigwedge_{K\in\mathcal{S}} enc\_fg_l(K), J \models \Omega_{\mathcal{N}}, \\
& \sum_{e=(i,j,[l,u]_T)\in\mathcal{E}_\gamma} (\Pi(j) - \Pi(i) - l - z_e T) s_i s_j \to \min
\end{aligned}
$$

$$J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),$$

$$J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e),$$

$$\overset{\text{comma}\equiv\text{conjunction}}{\Leftrightarrow} \exists \Pi : J = \xi^{-1}(\Pi),$$

$$\sum_{n\in\mathcal{H}} \gamma(n)s_n + \sum_{e=(i,j,[l,u]_T)\in\mathcal{E}_\gamma} (\Pi(j) - \Pi(i) - l - z_e T)s_i s_j \to \min,$$

$$J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},$$

$$J \models \bigwedge_{e=(n,m,[l,u]_T)\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(n,m,[l,u]_T),$$

$$J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e),$$

$$\Leftrightarrow \exists \Pi : \Pi \text{ minimizes } g \text{ and with } J = \xi^{-1}(\Pi) :$$

$$J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},$$

$$J \models \bigwedge_{e\in\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e)$$

$$J \models \bigwedge_{e\in\mathcal{E}\setminus\mathcal{E}_\gamma} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e)$$

$$\overset{\mathcal{E}=\mathcal{E}_\gamma\cup(\mathcal{E}\setminus\mathcal{E}_\gamma)}{\Leftrightarrow} \exists \Pi : \Pi \text{ minimizes } g \text{ and with } J = \xi^{-1}(\Pi) :$$

$$J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}},$$

$$J \models \bigwedge_{e\in\mathcal{E}} \bigvee_{p\in S} (\neg p) \vee \mathit{enc\_con\_max}(e)$$

$$\overset{\text{Def } 3.31,(4.37)}{\Leftrightarrow} \exists \Pi : \Pi \text{ minimizes } g \text{ and with } J = \xi^{-1}(\Pi) :$$

$$J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \Omega_{\mathcal{N}} \wedge \bigwedge_{e\in\mathcal{E}} \mathit{enc\_dec}(e)$$

$$\overset{\text{Def } 3.32}{\Leftrightarrow} \exists \Pi : \Pi \text{ minimizes } g \text{ and with } J = \xi^{-1}(\Pi) :$$

$$J \models \bigwedge_{K\in\mathcal{S}} \mathit{enc\_fg}_l(K), J \models \mathit{enc\_dpen}(\mathcal{D})$$

$$\overset{\text{proof Thm } 3.41}{\Leftrightarrow} \exists \Pi : \Pi \text{ minimizes } g \text{ and is valid for } \mathcal{D}, \mathcal{E}' \text{ under } \mathcal{S}$$

□

[15]Since it is an formula in WCNF, a split into subformulas is not equivalent. However, as proposed we will connect all structures in the end again with a conjuncted meaning with respect to the

This theorem allows us to equally apply MaxSAT solvers for optimizing FDPENs with respect to the objective functional in (4.15) maintaining feasibility with respect to Definition 3.17. Finally, we can conduct that previously presented applications[16] can be correctly solved by this approach.

Maximizing the number of to be inserted FGs is briefly introduced in Section 4.1.3. Encoding this goal with as MaxSAT problem can be simply done by using the introduced propositional decision variables u_K for each K and modify the encoding *enc_fg_max* such that for each K we use

$$\neg u_K \vee \mathit{enc\_fg\_max}(K)$$

which can be simply transformed semantically equivalent into a propositional formula in WCNF with de Morgan's rules as in Example 2.15. The optimization part (see (4.29)) would be simply the conjunction of all introduced variables such that

$$\bigwedge_{K \in \mathcal{S}} (1, u_K)$$

with $\mathcal{S}$ being the set of to be maximized FGs. Please note, that the weight 1 could simply be set to a domain specific weight.

optimization.

[16]see Section 3.3

5 Computational Results

This chapter covers computational results on real-world data for the applied models of the previous sections. If possible, different solvers for the same tasks will be used. The underlying hardware consists of an Intel® Core™ i7-4790K CPU and 32 GB RAM. However, the memory limit has never been reached for any used solver and instance. The general hardware is chosen to show the possible application of the algorithms for commodity personal computers. Furthermore, the software system TAKT can be used on such hardware systems efficiently. If possible, all 8 CPUs' cores are use but only the real expired time, which contains possible encoding times as well, will be presented.

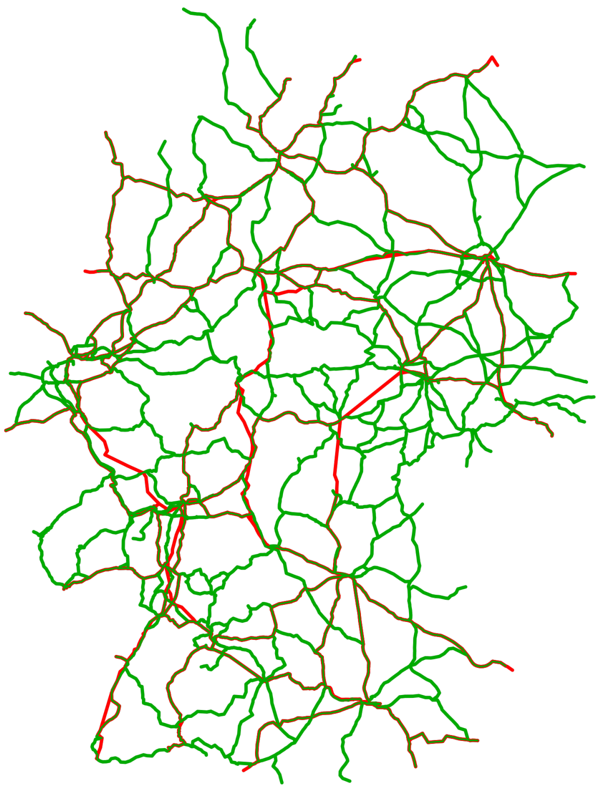


Figure 5.1: The whole intercity network (red) of Germany combined with the most important regional train paths (blue) (instance *wg*).

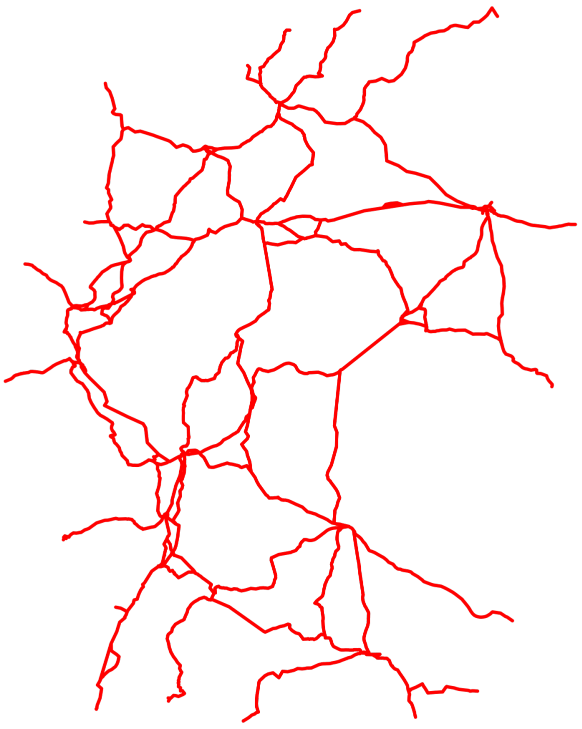


Figure 5.2: The whole intercity network of Germany (instance wg_2).

Different heuristics may use several state-of-the-art SAT solvers in parallel. However, here we only use the SAT solver GLUCOSE (Version 4.0) [AS12; ALS13]. As MIP solver we use *Gurobi 6.0.3* [GRB15]. The initial solution for the MIP solver is generated by the PESP to SAT method as in Section 2.3.2. For the MaxSAT methods, we apply the solver *open-wbo* [MML14].

The section covers an introduction into the instances in Section 5.1 and is followed up by computational results for the base and advanced encoding and the minimization of weighted slacks in Section 5.2 and Section 5.3, respectively. Furthermore, the calculations with FDPENs for duplicated and non-connected flow graphs are presented accordingly in Section 5.4 and Section 5.5.

5.1 Instances

All instances are generated by the software system TAKT and cover the necessary constraints for real-world railway timetabling [Opi09], like headway, connection and symmetry. The based data are provided by the most important German railway infrastructure company DB Netz AG that work in close collaboration with the Chair of Traffic Flow Science (TU Dresden). The headway constraints are based on a microscopic grained infrastructure in order to ensure the timetables to be conflict-free maintaining an efficient use of the capacity. Hence, this type covers around 80 % of all constraints [Küm+13]. The PESP instances[1] cover networks of south-east Germany (*p_2*, *p_{15}*, *b*, *seg_1*, *seg_2*, *dp_{43}*, *dup*), south-west Germany (*swg_1*, *swg_2*, *swg_3*, *swg_4*, *mb_{12}*), the whole inter city network of Germany (*fernsym*, *wg_3*) and the whole intercity network of Germany combined with the most important regional train paths (*wg*). All of which have a base period of $T = 120$. Table 5.1 shows for all used instances the number of nodes and edges which range from 211 constraints (*p_2*) up to 85 034 constraints (*wg*) for base PENs and 343 765 constraints (*mg_{12}*) for FDPENs, and the maximum relaxation (slack) parameter θ (if needed) as in Section 4.2.1. The ascending order for the number of constraints is chosen, since this seems to have the highest impact on computation times. The used infrastructure networks of *wg* and *wg_2* are given in Figure 5.1 and Figure 5.2, respectively. The largest PEN (*wg*) consists of 1 470 train paths that shall be planned. The largest network consists of 3929 stations[2] and total railway track length of approx. 25 000 km.

Different sections may use different instances for the respective purposes and will be given explicitly. Additional information, like flow graphs, size of encodings, are given in the respective sections.

As divided previously, Section 5.2 shows the solution times of the efficient base encodings. Furthermore, Section 5.3 covers the minimization of weighted slacks, respectively resolving infeasible periodic event networks. In the end, Section 5.4 and Section 5.5 shows the successful application of FDPENs. All given approaches are already used in the software system TAKT and thus, is already used by practitioners like staff of DB Netz AG.

5.2 Base and Advanced SAT Encoding

The base of all SAT encodings represents the approach in Section 2.3.2 which has been shown to be the current best method to solve PESP [Gro+12a]. This section covers the computational results compared with the advanced SAT encoding in Section 2.3.3. We show that the SAT-based approaches can outperform a native domain solver by up to four orders of magnitude. For comparison, in the literature [Gro+12a] the SAT solver RISS by Norbert Manthey and Peter Steinke [HMS10] is used.

[1]Please note, the name of each instance has historical reasons and is maintained for better recognition and usage of future works.

[2]a station is given if it has at least two junctions and a at least one platform

Table 5.1: PESP instances and relaxation parameters.

	PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$		relaxation (slack)
instance	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	θ
dr	14	6	-
p_2	137	211	5
dh	173	222	-
p_{15}	251	689	5
k	76	788	-
dp_{43}	1 074	5 768	15
swg_2	82	1 228	-
fernsym	135	3 368	-
swg_3	216	3 162	-
b	953	6 305	-
swg_4	191	7 227	-
swg_1	259	7 729	-
seg_1	3 195	12 839	-
seg_2	3 221	14 417	-
we	3 263	17 107	-
wg_2	3 652	18 881	4
dup	1 405	22 724	30
save	14 017	81 452	-
wg	14 241	85 034	5
mb_{12}	3 457	343 765	60

The evaluated PENs can be seen in Table 5.2. The native domain solver for solving PESP instances is described and analyzed by Opitz [Opi09] and Nachtigall [Nac98]. The solver is equipped with a decision tree method with respect to the events' potentials. Additionally, constraint propagation techniques are employed by propagating the current valid assignments of the events across the network. Once the propagation detects a potential with an empty domain, a conflict is found. In this case, the tree search will backtrack. This solver will be referred as PESPSOLVE.

The runtime timeout is set to 24 h. The results in Table 5.2 clearly show that the SAT-based approaches perform much better than PESPSOLVE for both encodings. The runtime per PESP instance for the SAT approaches is the sum of the encoding time and the solving time. Comparing the run times of PESPSOLVE and the SAT approaches[3] always results in a significant speedup. Concerning the number of instances, that can be solved within the timeout, the SAT-based approaches also show a better performance

[3]i. e., *base* + GLUCOSE and *advanced* + GLUCOSE

Table 5.2: PESP instances and corresponding solving times in seconds.

instance	PESPSOLVE	*base* + GLUCOSE in *s*	*advanced* + GLUCOSE in *s*
k	32 481	11	10
swg_2	361	2	1
fernsym	1 599	3	1
swg_3	41	2	1
b	timeout	95	43
swg_4	731	5	3
swg_1	timeout	5	2
seg_1	timeout	8	7
seg_2	timeout	5	5
we	timeout	timeout	timeout
save	timeout	24	10

than PESPSOLVE. By using a SAT-based approach, all except one of the tested networks (*we*) can be solved. The different solving times compared to the literature [Gro+12a] can be reduced to the usage of a different SAT solver and better equipped hardware.

It can be concluded as well that the advanced encoding outperforms the base encoding slightly. The reduction to SAT allows to tackle a whole set of larger instances, which could not be solved before. Furthermore, it can be applied to iterative techniques as the binary search heuristic [Gro+15b] in Section 5.3 which requires a lot of instances to be solved in short time in order to be used efficiently by engineers.

5.3 Minimization of Weighted Slacks

This section covers a computational comparison for real-world public railway transport networks of both a binary heuristic [Gro+15b] and MaxSAT approach in Section 4.2.1 in order to resolve infeasible PENs. Additionally for comparison, the MIP approach in Section 4.1.1 [Nac98; ON08] will be applied on each instance.

The given scenarios were previously all infeasible with respect to a valid timetable and thus, are relaxed by the maximum slack parameter θ in order to use the minimum amount of relaxation minutes.[4]

As reference point we are using the binary search heuristic[5] [Gro+15b] such that the

[4]The maximum slack θ is only applied to relaxable constraints whose weight is less than ∞, like waiting (stopping) and symmetry constraints. Constraints like minimal headway are not relaxable in order to preserve conflict freeness.

[5]since this has been already successfully applied in the past [Küm+13]

objective functional values of both the MaxSAT and MIP approach are compared to the point in time when the heuristic finishes. Since the heuristic stops in a possible local minimum, the other approaches[6] are allowed to compute even further but are stopped if the value of the objective is no more improved in a reasonable time frame.

Table 5.3: Computational times and optimization values for the instances

	binary heuristic		MaxSAT			MIP		
instance	time in s	obj.	time in s	bound	obj.	time in s	bound	obj.
p_2	6	4	5	4	**4**	12	4	4
p_{15}	5	**5**	5	5	**5**	337	5	5
dp_{43}	125	48	2	47	**47**	3 300	0	97
wg_2	1 659	87	6 306	78	**78**	10 090	0	117
wg	49 964	**2 452**	60 587	451	8 859	59 582	458	3 436

In Table 5.3 all computational times, the objective functional values and for the MaxSAT and MIP approach the best lower bounds are presented. For each instance, the best objective value is highlighted and in case it is the same, the approach with the lowest needed time is highlighted. Besides instance wg, the MaxSAT method seems promising with respect to objective value and computation time. However, the binary heuristic approach tends[7] to get faster better solutions. Furthermore, combining both approaches in parallel outperform each MIP run besides the lower bound of wg.

In Figure 5.3 the comparison of the objective values (y axis) against the time (x axis) for most instances is presented. For instance wg, it can be seen that the MIP approach finds faster a better primal solution, however, the heuristic approach finds the best objective value after it finishes and crosses the solution quality of the MIP approach at 3 000 s. Instance p_{15} follows the same procedure (log-scaled), however, all approaches result in the same objective value 5 which represents the global optimum. The MIP approach finds after 88 s the best primal, yet, needs 337 s to verify its optimality. Omitting the MaxSAT approach in Figure 5.3 for instance dp_{43} is based on the fast computation time of its run.

The engineering impact of the instances' solutions and the general need to do timetable optimize are discussed in detail in the literature [Küm+15; Opi09]. Any improvement of the process with respect to computation time or value of the objective leads to a faster or

[6]which are global optimization approaches

[7]see Fig. 5.3

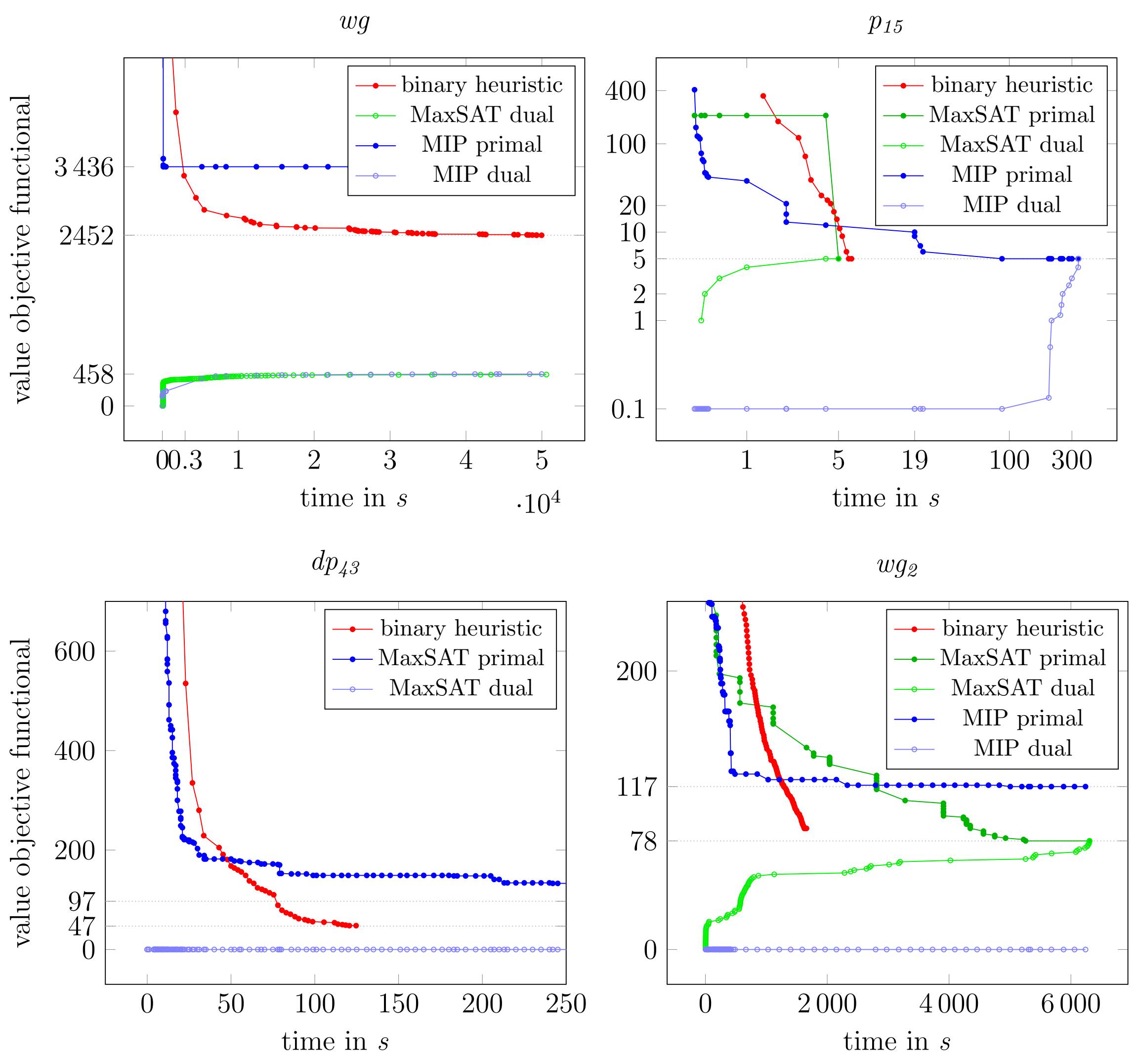


Figure 5.3: Comparison of computational results for instances wg, p_{15}, dp_{43}. wg_2

better timetable, respectively. Consequently, based on the computational results a hybrid approach should be investigated such that all solvers run in parallel or the solution of one solver's run may be passed as initial solution for the other methods.

5.4 Duplicated Chain Paths

Inserting periodic rail freight paths into an public railway transport system is an important task for practitioners [FP14]. The introduced application of FDPENs in Section 3.3.3 for this problem will be shown for two instances in this section. The used maximum relaxation parameters as in Table 5.1 represent the maximum waiting time for each stop of each to be inserted rail freight train path.

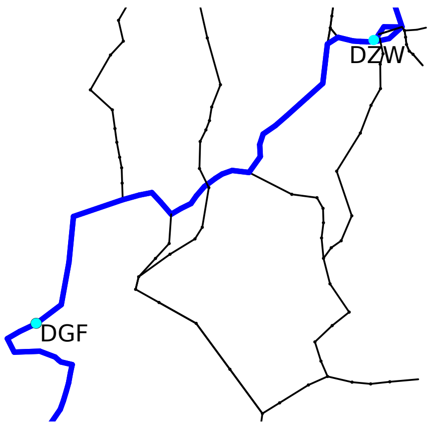


Figure 5.4: Relation of to be inserted rail freight train paths between DGF (Gutenfürst, Germany) and DZW (Zwickau (Sachs) Hauptbahnhof, Germany) with possible routes.

The already addressed example in Section 3.3.2 will be given in more detail in this section. Figure 5.4 displays the relation of the to be inserted flow graphs from FMB (Mainz-Bischofsheim) to RKR (Karlsruhe). It consists of 77 flow edges for each flow graph. The number of FGs[8] is determined to be 7. In Figure 5.5 a timetable for the passenger trains is represented. Without the to be inserted rail freight train paths the PEN consists of 866 periodic events and 2 676 constraints with a base period of 120 min which represent a total of 72 passenger train paths (8 long-distance, 64 regional) on the corresponding corridor. The newly to be freight train paths blows up the respective FDPEN to 1 405 periodic events and 22 724 constraints that consists in major of minimum headway constraints due to the duplicated paths (flow graphs).

The respective SAT encoding as in Section 3.2.2 consists of 78 246 variables and 2 422 069 clauses. The computation time (including the encoding time) needed was 8 s and results in the needed amount of to be inserted freight train paths in Figure 5.6.

This rather easy scenario leads to the expectation of more difficult parts of Germany with a wider spread with respect to distance and flow edges used. Thus, the following real-world instance represents the most difficult part of Germany for inserting periodic rail freight train paths.[9] The infrastructural relation of the highly frequented corridor lays between Mannheim (Germany) and Basel (Switzerland) that is presented in Figure 5.7.

The public rail transport system consists of 87 periodic passenger train paths and

[8]that is the number of to be inserted rail freight rain paths
[9]that is common sense of the staff of DB Netz AG.

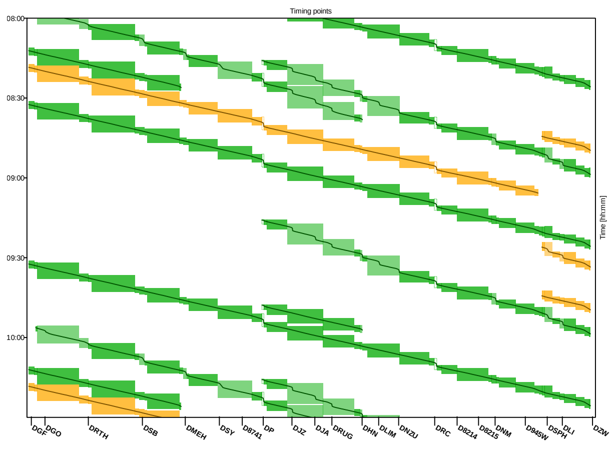


Figure 5.5: Timetable (time-distance graph) for passenger train paths between DGF (Gutenfürst) and DZW (Zwickau (Sachs) Hauptbahnhof, Germany).

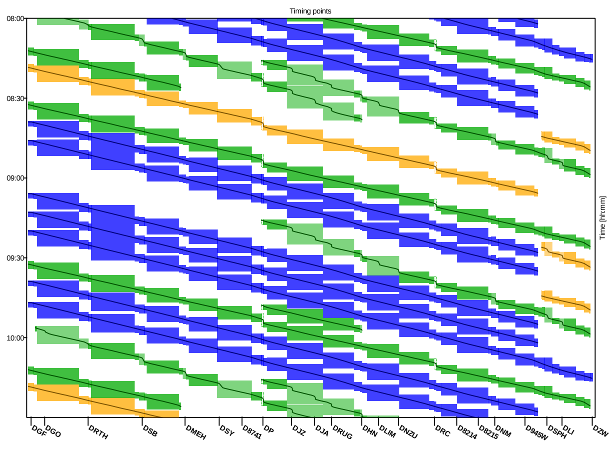


Figure 5.6: Timetable with inserted rail freight train paths (blue) between DGF (Gutenfürst, Germany) and DZW (Zwickau (Sachs) Hauptbahnhof, Germany).

the number of to be inserted rail freight train paths is 12. The number of flow edges (small construction parts) for each flow graph is 241. The passenger train paths instance (the PEN) consists of 565 periodic events and 4 195 constraints. The timetable of the passenger trains is given in Figure 5.8.

Combining this with the FGs to a FDPEN results in an instance of 3 457 periodic events and 343 765 constraints. This huge instance results in an encoded propositional formula consisting of 357 393 variables and 42 769 822 clauses which could be solved by the SAT solver in 476 s including the encoding time. The resulting timetable for the instances is visualized in Figure 5.9.

Table 5.4 shows the problem sizes with the extracted passenger train system as in Lemma 3.5 for each FDPEN $\mathcal{D} = (\mathcal{V}, \mathcal{E}, T, \mathcal{H}, \mathcal{A})$ with the used computation times of the

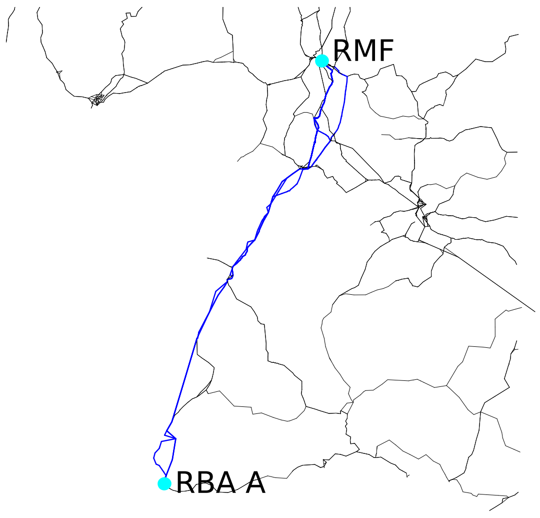


Figure 5.7: Relation of to be inserted rail freight train paths between RMF (Mannheim, Germany) and RBA A (Basel, Switzerland) with possible routes.

Table 5.4: Instances with corresponding passenger network and encoding sizes.

	FDPEN $\mathcal{D}$		encoding $\mathcal{F}$		
instance	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|vars(\mathcal{F})\|$	$\|\mathcal{F}\|$	comp. time in s
dup	1 405	22 724	78 246	2 422 069	8
mb_{12}	3 457	343 765	357 393	42 769 822	476

encoding and the SAT solver.

To the best of the authors knowledge, since no other software, nor operator of the largest German railway infrastructure company DB Netz AG, could automatically insert the same amount of periodic rail freight train paths with the same quality or time results in the conduction that this approach is the current state of the art [WON12]. Furthermore, this approach (in TAKT) is used by DB Netz AG to plan all relations in Germany for possible upcoming timetables.

5.5 Non-connected Flow Graphs

Allocating tracks of stations to certain train paths is the application introduced in Section 3.3.3 for non-connected FDPENs. This section shows the solution for several stations based on real-world data that have been partly shown previously, yet here more in detail.

The previously rather small station DR (4 tracks, Riesa, Germany) as depicted in Figure 5.10 consists of 10 periodic train paths, that can be divided into 4 long-distance

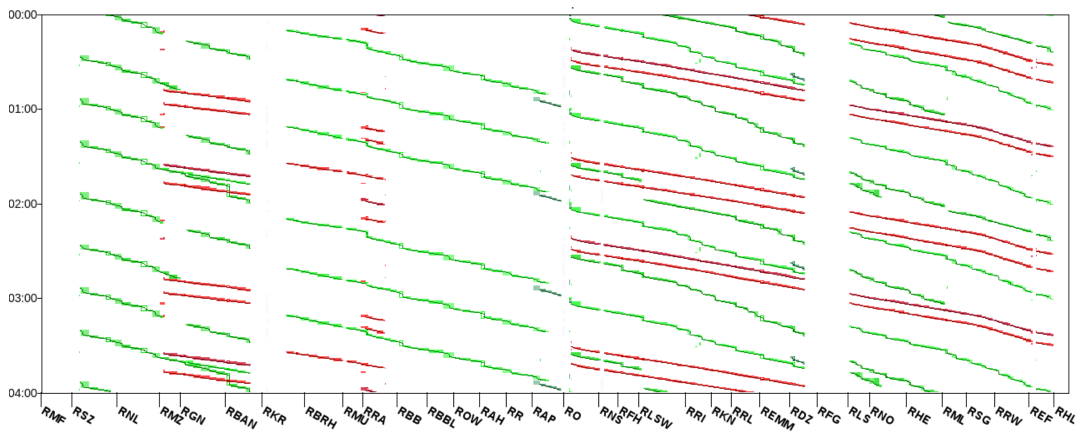


Figure 5.8: Timetable (time-distance graph) for passenger train paths between Mannheim (Germany) and Basel (Switzerland).

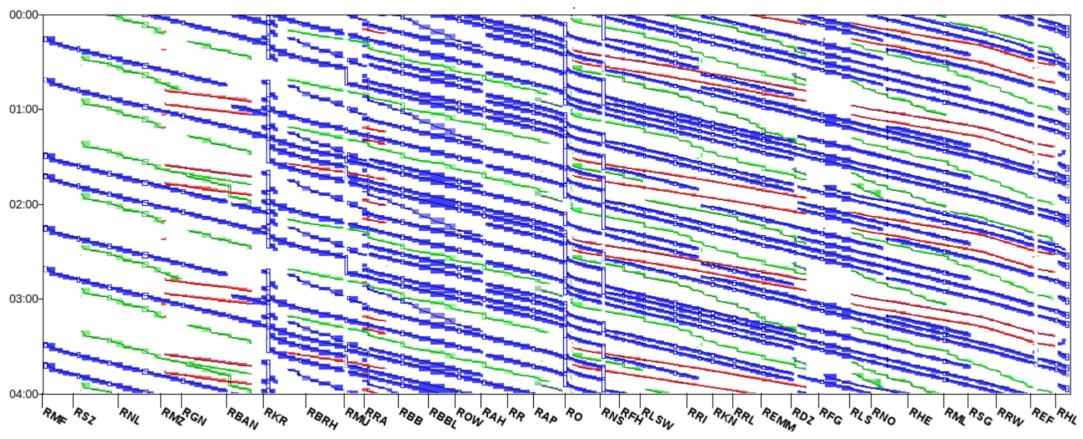


Figure 5.9: Timetable with inserted rail freight train paths (blue) between Mannheim (Germany) and Basel (Switzerland).

train paths (period of 120 minutes) and 6 regional train paths (periods of 60 and 120 minutes). It consists of a total of 14 FGs (possible routes) for all periodic train paths and results in an FDPEN of 14 periodic events and 6 constraints (instance *dr*).

The computation of the timetable took less than a second and is depicted previously in Figure 3.24 which can be seen as a conflict free track allocation since no box is intersecting.

Another applied use case is the station DH (Dresden main station, Germany) which covers a total of 32 periodic train paths, whereas there are 8 long-distance trains and 24 regional trains. All train paths may choose different routes according to the station path rules[10] which results in a total of 173 flow graphs (routes). Hence, the resulting

[10]see Figure 3.25

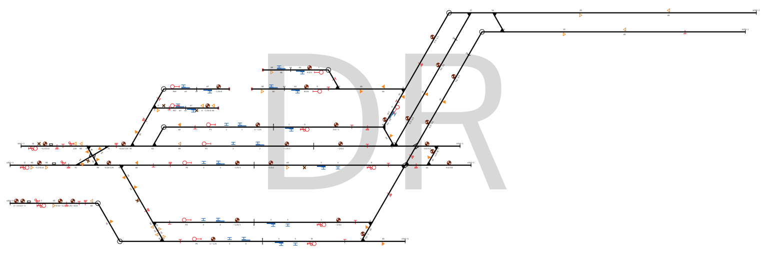


Figure 5.10: Infrastructure with train paths of station DR (Riesa, Germany).

FDPEN consists of 173 periodic events and 222 constraints (instance *dh*).

The solution could be calculated in less than 1 second and can be seen again in Figure 5.11. Furthermore, if a not a valid track allocation can be done, we can search for the minimal conflict as presented in the literature [Gro12b]. Please note, that currently for this iterative process we fix all nodes (events) to its previously calculated departure times of the timetabling process. A more general way would be achieved by using the distinct chain path approach as in Section 3.3.1 which is currently under research and development for to be evaluated use cases [Wün16].

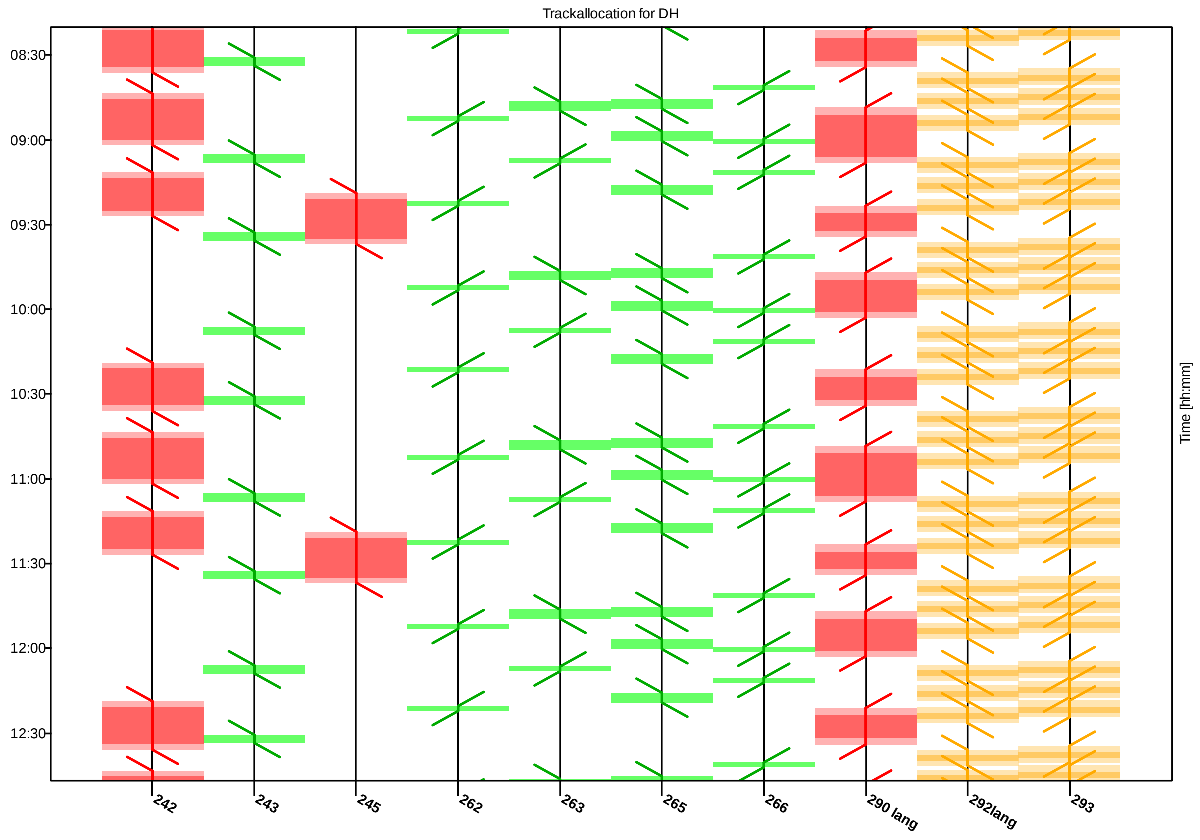


Figure 5.11: Allocated tracks for all train paths in station DH (Dresden Hauptbahnhof, Germany).

6 Conclusion and Outlook

In this work, we have shown and computationally verified several important theoretical and experimental results in the operations research area of solving domain-specific problems in periodic event scheduling. Firstly, we have shown a novel approach for extending periodic timetabling with decisional flows (transportation networks) alongside possible applications for this new model. Furthermore, a complexity classification for periodic event scheduling with flows is given. The introduction of an appropriate SAT encoding is concluded by a soundness and completeness theorem. Additionally, we presented the possibility of optimization extensions with several objective functionals yet again with appropriate (Max)SAT encodings. Additionally, soundness and completeness theorems offer us the possibility to use these encoded propositional formulas in state-of-the-art solvers correctly.

All these theoretical insights have been compared, analyzed and verified in computational tests. The results suggest a very promising usage of the introduced approaches for integration of routing and timetabling, planning of periodic rail freight train paths and track allocation. The state-of-the-art SAT and MaxSAT (and sometimes MIP) solvers could handle most instances in reasonable time frames.

For future outlook, we can now handle larger and even more complex networks and hence, it opens a lot of new possibilities to solve new real-world scenarios for different applications. Nevertheless, a possibly better encoding could result in even faster computational times or better objective functional values. Additionally, new applications and use cases for the new approach need to be researched in order to fully (partly) utilize the new modeling power of periodic event scheduling with flows. A further extension would be the possible encoding (modeling) of dynamic velocities for the small construction parts (sub train paths, flow edges) such that they do not need to be discretized. This could result in a smaller propositional formula and more flexibility (degrees of freedom) for the railway timetabling problem. However, encoding real (in $\mathbb{R}$) numbers in SAT is not possible[1] and again, needs to be discretized, which probably reduces possible

[1]since the variable assignment's domain is $\{f, t\}$

modeling approaches to MIP.

Finally, it can be concluded that integrating path choices with periodic timetabling for public railway transport networks opens a wide variety of applications and supports drastically the timetabling process in several domains. The major improvements in state-of-the-art (Max)SAT solvers and the efficient encodings for the integrated model allow us to solve these problems satisfactorily with a promising outlook for further scientific investigations.

Appendix

We want to briefly introduce the possibility of a new native-domain PESP solver that can learn and backjump over several decision levels which is already successfully applied in the SAT community [Bie+09].

CDCL SAT Solver Currently all (most) state-of-the-art (non-probabilistic) SAT solvers that solve propositional formulas in CNF use the so called Conflict Driven and Clause Learning (CDCL) technique [Bie+09; ALS13]. This approach does not simply backtrack in case a propositional variable needs to be assigned to true and false, but analyzes the conflict why it did occur by an implication graph. Consequently, it will backjump over several decision levels and even can learn a clause which prevents that the algorithm will ever again get into the same conflict [Bie+09]. Here, we try to apply and transform this technique for PESP.

Partial Schedule and Constraint Propagation In order to assign possible values (potentials) to periodic events we introduce the structure of partial schedules. With $\mathcal{N} = (\mathcal{V}, \mathcal{E}, T)$ being a PEN and $x_1, x_2, \ldots \in \mathcal{V}$ being periodic events,

$$\mathcal{T} = ((x_1, \mathcal{Z}_1), (x_2, \mathcal{Z}_2), \ldots)$$

is called *partial schedule* for $\mathcal{V}$ if $\mathcal{Z}_i \subseteq [0, T-1]$.

A *partial* schedule $\mathcal{T}$ is a schedule for $\mathcal{V}$, if

$$\forall x \in \mathcal{V} : \mathcal{T}_x := \begin{cases} \bigcap_{(x,\mathcal{Z}) \in \mathcal{T}} \mathcal{Z}, & \exists (x, \mathcal{Z}) \in \mathcal{T} \\ [0, T-1]_T, & \text{else} \end{cases} : |\mathcal{T}_x| = 1.$$

Thus, it holds for all events $x \in \mathcal{V}$: $T_x = i$ for $\mathcal{T}_x = \{i\}$. We can append a new assignment to the partial schedule via $\circ$ such that

$$((x_1, \mathcal{Z}_1), \ldots, (x_k, \mathcal{Z}_k)) \circ (x_{k+1}, \mathcal{Z}_{k+1}) := ((x_1, \mathcal{Z}_1), \ldots, (x_k, \mathcal{Z}_k), (x_{k+1}, \mathcal{Z}_{k+1})).$$

Furthermore, an event is called *free* if and only if $|\mathcal{T}_x| > 1$. The method of *constraint propagation* is well-known [Nac98] and is iteratively applied until no more changes do appear for

$$a = (i, j, [l, u]_T) \in \mathcal{E} : \mathcal{Z}_j := \mathcal{Z}_j \cap ([l, u]_T + \mathcal{Z}_i).$$

Combining this with a simple backtracking algorithm results in a possible algorithm to solve PESP [Nac98; Opi09] which performs worse than the PESP to SAT approach [Gro+12a] (see Section 5.2).

Decision Level For learning PESP solvers we need as well *decision levels* $i \in \mathbb{N}$ for each event n that is denoted as n^i. Once a potential (node assignment) is added as *decision* to the partial schedule, we increase the decision level by one and mark the event with an overline. E. g.,

$$\mathcal{T} = ((\overline{n}^1, \{0\}), (m^1, \{15, 17\}), (p^1, \{17, 18\}), (\overline{m}^2, \{15\}), (p^2, \{18\}))$$

is a partial schedule with devision levels.

Example Network Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, 10)$ be a PEN as in Figure A.1 with the set of nodes $\mathcal{V} = \{n, m, q, r, s\}$. Then, these constraints can be disjunctively formulated as

$$\begin{aligned}\mathcal{E} =\{&(r, s, [1, 1]_{10} \cup [3, 3]_{10}),\\ &(r, q, [1, 1]_{10} \cup [6, 6]_{10} \cup [9, 9]_{10}),\\ &(q, s, [1, 1]_{10} \cup [3, 3]_{10} \cup [7, 7]_{10},\\ &(n, q, [1, 1]_{10} \cup [5, 6]_{10})\}\end{aligned}$$

with an exemplifying valid schedule Π such that

$$\Pi(n) = 0, \Pi(m) = 4, \Pi(q) = 1, \Pi(r) = 5, \Pi(s) = 8.$$

Implication Graph Every implication done by constraint propagation (for example, $(p^1, \{17, 18\})$) is based on a *reason* that is based either on a previous decision (variable/node assignment) or on a propagation. With these information we can construct a so called *implication graph* with nodes

$$(x^i, \mathcal{Z}_x) \in \mathcal{T}$$

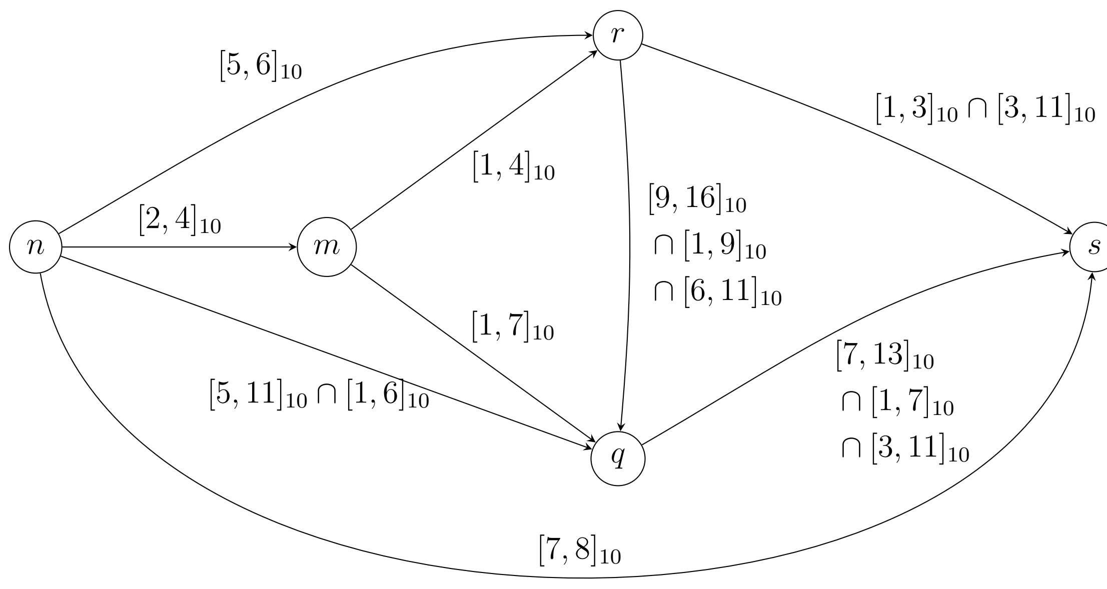


Figure A.1: Example periodic event network for the learning PESP solver.

and edges which represent implications between two nodes $(x^i, \mathcal{Z}_x) \in \mathcal{T}$ that are the reasons induced by the constraint propagation.

For the previously introduced exemplifying PEN $\mathcal{N}$ we use for the backtracking approach the sequence $n \to m \to q \to r \to s$ as variable assignment method. Using constraint propagation results in the partial schedule (with decision levels)

$$\begin{aligned}\mathcal{T} = ((\overline{n}^1, \{0\}), (m^1, \{2,3,4\}), (r^1, \{5,6\}), (q^1, \{1,5,6\}), (s^1, \{7,8\}),\\ (\overline{m}^2, \{2\}), (q^2, \{5,6\}), (\overline{q}^3, \{5\}), (r^3, \{6\}), (s^3, \{8\}), (s^3, \emptyset)).\end{aligned}$$

The empty set ($\emptyset$) indicates that we ran into a conflict and would normally need to backtrack to level 3 $(\overline{q}^3, \{5\})$. However, we want to use a new approach. Consequently, we can construct for the partial schedule $\mathcal{T}$ the previously introduced implication graph that is depicted in Figure A.2. Please not that in this figure the implications from $(\overline{m}^2, \{2\})$ are omitted for a better overview.

Backjumping and Learning Now, we can backjump to the so called 1-UIP (1-unique implication point) [Bie+09], because $\neg(\overline{q}^3, \{5\}) \vee \neg(\overline{n}^1, \{0\})$ implies a backjump to decision level 1 with

$$\mathcal{T} = ((\overline{n}^1, \{0\}), (q^1, \{0, \ldots, 4, 6, \ldots, 9\})).$$

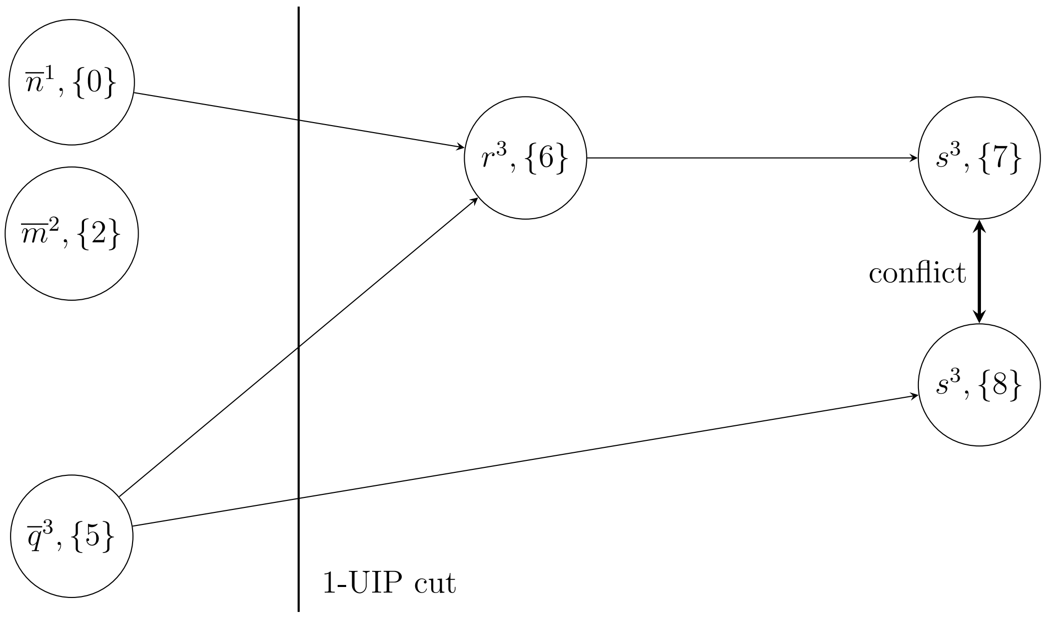


Figure A.2: Implication graph for the given partial schedule.

Applying again constraint propagation we get the previously introduced valid schedule Π without doing any further decision. In case of "normal" backtracking we would have tested all not valid combinations for q and m until we would have finally chosen the correct potential $\Pi(m) = 4$.

We can conduct that we are now able to not just backtrack by one decision level, but jump over several decision levels avoiding the same conflict we may have driven into. The question arises whether we can learn further information by the implication graph as in the CDCL SAT solvers [Bie+09] that can learn clauses (lemmas) that can be connected conjunctively to the propositional formula.

For this example we can imply

$$\begin{aligned}&\neg(q, \{5\}) \vee \neg(n, \{0\}) \equiv \neg((q, \{5\}) \wedge (n, \{0\}))\\ &\Leftrightarrow \neg((n, q, [5, 5]_{10}) \in \mathcal{E} \Leftrightarrow (n, q, [6, 14]_{10}) \in \mathcal{E}.\end{aligned}$$

This constraint can directly be connected to the PEN $\mathcal{N}$ (respectively the set of constraints $\mathcal{E}$). In the next run the algorithm will never be driven into the same conflict again.

Yet another question arises: How can we handle cuts which consists of more than three nodes? How can the algorithm lean these information? Is this possible with classical

PESP? For example, we could learn

$$\begin{aligned}
&\neg(q,\{5\}) \vee \neg(n,\{0\}) \vee \neg(m,\{3\}) \\
\equiv\ &(\neg(q,\{5\}) \vee \neg(n,\{0\})) \vee (\neg(n,\{0\}) \vee \neg(m,\{3\})) \\
\equiv\ &((n,q,[6,14]_{10})) \in \mathcal{E} \vee ((n,m,[4,12]_{10}) \in \mathcal{E}).
\end{aligned}$$

This *disjunction* is a major problem, because in PESP all constraints are connected *conjunctively*, since all constraints must hold for a valid schedule. Hence, a first solution would be to use a new structure separately for the learnt disjunctive constraints. The probably better solution would be to transform this into a conjunctive form which is left open for further research.

Bibliography

[ALS13] G. Audemard, J.-M. Lagniez, and L. Simon. „Improving Glucose for Incremental SAT Solving with Assumptions: Application to MUS Extraction“. In: *16th International Conference on Theory and Applications of Satisfiability Testing (SAT'13)*. Vol. 7962. Lecture Notes in Computer Science. Springer, 2013, pp. 309–317.

[AS12] G. Audemard and L. Simon. „GLUCOSE 2.1: Aggressive, but Reactive, Clause Database Management, Dynamic Restarts (System Description)“. In: *Pragmatics of SAT 2012 (POS'12)*. SAT'2012. June 2012.

[BDP04] A. Bar-Noy, V. Dreizin, and B. Patt-Shamir. „Efficient algorithms for periodic scheduling“. In: *Computer Networks* 45.2 (2004), pp. 155–173.

[Bie+09] A. Biere, M. Heule, H. van Maaren, and T. Walsh, eds. *Handbook of Satisfiability.* IOS Press, 2009.

[Bie13] A. Biere. „Lingeling, Plingeling and Treengeling Entering the SAT Competition 2013“. In: *In Proceedings of SAT Competition 2013*. Vol. B-2013-1. University of Helsinki: Department of Computer Science Series of Publications, 2013.

[Bon08] J. A. Bondy. *Graph theory.* New York: Springer, 2008.

[BT05] R. Bosch and M. Trick. „Integer programming“. In: *Search methodologies.* Springer US, 2005, pp. 69–95.

[Bud+10] G. Budai, G. Maróti, R. Dekker, D. Huisman, and L. Kroon. „Rescheduling in passenger railways: the rolling stock rebalancing problem“. In: *Journal of Scheduling* 13.3 (June 2010), pp. 281–297.

[Cai+07] G. C. Caimi, M. Fuchsberger, M. Laumanns, and K. Schüpbach. „09. Periodic Railway Timetabling with Event Flexibility“. In: *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07).* Ed. by C. Liebchen, R. K. Ahuja, and J. A. Mesa.

Vol. 7. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2007.

[CCT10] V. Cacchiani, A. Caprara, and P. Toth. „Scheduling Extra Freight Trains on Railway Networks“. In: *Transportation Research Part B: Methodological* 44.2 (2010), pp. 215–231.

[CDE08] C. Cadar, D. Dunbar, and D. R. Engler. „KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs.“ In: *OSDI*. Ed. by R. Draves and R. van Renesse. USENIX Association, 2008, pp. 209–224.

[CGT15] V. Cacchiani, L. Galli, and P. Toth. „A tutorial on non-periodic train timetabling and platforming problems“. In: *EURO Journal on Transportation and Logistics*. Vol. 4(3). Springer, 2015, pp. 285–320.

[Che10] J.-C. Chen. „A new SAT encoding of the at-most-one constraint“. In: *Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation* (Sept. 2010).

[Cla+01] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. „Bounded Model Checking Using Satisfiability Solving“. In: *Form. Methods Syst. Des.* 19 (1 July 2001), pp. 7–34.

[Coo71] S. A. Cook. „The complexity of theorem-proving procedures“. In: *Proceedings of the third annual ACM symposium on Theory of computing*. STOC '71. Shaker Heights, Ohio, United States: ACM, 1971, pp. 151–158.

[DW83] M. D. Davis and E. J. Weyuker. *Computability, complexity, and languages - fundamentals of theoretical computer science*. Computer science and applied mathematics. Academic Press, 1983, pp. I–XIX, 1–425.

[FM06] Z. Fu and S. Malik. „On Solving the Partial MAX-SAT Problem“. In: *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*. SAT'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 252–265.

[FP14] M. Feil and D. Pöhle. „Why Does a Railway Infrastructure Company Need an Optimized Train Path Assignment for Industrialized Timetabling?“ In: *Proceedings of International Conference on Operations Research*. Aachen, 2014.

[Fri+05] A. M. Frisch, T. J. Peugniez, A. J. Doggett, and P. W. Nightingale. „Solving Non-Boolean Satisfiability Problems with Stochastic Local Search: A Comparison of Encodings“. In: *J. Autom. Reason.* 35.1-3 (Oct. 2005), pp. 143–179.

[GRB15] Z. Gu, E. Rothberg, and R. Bixby. *Gurobi 6.0.3.* Houston, TX: Gurobi Optimization, Inc., May 2015.

[Gro+12a] P. Großmann, S. Hölldobler, N. Manthey, K. Nachtigall, J. Opitz, and P. Steinke. „Solving Periodic Event Scheduling Problems with SAT“. In: *IEA/AIE.* Vol. 7345. LNAI. Springer, 2012, pp. 166–175.

[Gro+12b] P. Großmann, R. Weiß, J. Opitz, and K. Nachtigall. „Automated Generation and Optimization of Public Railway and Rail Freight Transport Time Tables“. In: *Trans & Motauto '12. International Scientific-Technical Conference.* Vol. 2. Scientific-Technical Union of Mechanical Engineering, 2012.

[Gro+13] P. Großmann, A. Labinsky, J. Opitz, and R. Weiß. „Capacity-Utilized Integration and Optimization of Rail Freight Train Paths into 24 Hours Timetables“. In: *MT-ITS.* TUDpress, 2013, pp. 389–396.

[Gro+15a] P. Großmann, J. Opitz, R. Weiß, and M. Kümmling. „Extending Periodic Event Scheduling by Decisional Flow Transportation Networks“. In: *27th European Conference On Operational Research.* Glasgow, Scotland, presentation, 2015.

[Gro+15b] P. Großmann, J. Opitz, R. Weiß, and M. Kümmling. „On Resolving Infeasible Periodic Event Networks“. In: *Proceedings of the 13th Conference on Advanced Systems in Public Transport (CASPT) 2015.* Erasmus University, 2015.

[Gro11] P. Großmann. *Polynomial Reduction from PESP to SAT.* Tech. rep. 4. Technische Universität Dresden, Germany, Oct. 2011.

[Gro12a] P. Großmann. „Automatic Scheduling of Periodic Event Networks by SAT Solving“. In: *Operations Research Proceedings 2012.* Springer, 2012, pp. 481–486.

[Gro12b] P. Großmann. *Extracting and Resolving Local Conflicts in Periodic Event Networks.* Diploma thesis. TU Dresden, 2012.

[Gro13] P. Großmann. „On Extracting Minimally Infeasible Periodic Event Networks“. In: *26th European Conference On Operational Research.* Rome, Italy, presentation, 2013.

[HMS10] S. Hölldobler, N. Manthey, and A. Saptawijaya. „Improving resource-unaware SAT solvers“. In: *Logic for Programming, Artificial Intelligence, and Reasoning.* Ed. by C. Fermüller and A. Voronkov. Vol. 6397. LNCS. Springer, 2010, pp. 357–371.

[Hoc12] D. S. Hochbaum. *Lecture Notes for IEOR 266: The Pseudoflow Algorithm Graph Algorithms and Network Flows.* 2012.

[Hog06] L. Hogben. *Handbook of Linear Algebra.* 1st ed. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, Nov. 2006.

[HP14] I. Hansen and J. Pachl. *Railway Timetabling & Operations: Analysis, Modelling, Optimisation, Simulation, Performance Evaluation.* Eurailpress in DVV Media Group, 2014.

[KGO15] M. Kümmling, P. Großmann, and J. Opitz. „Maximisation of homogeneous rail freight train paths at a given level of quality“. In: *27th European Conference On Operational Research.* Glasgow, Scotland, presentation, 2015.

[KK07] W. Klieber and G. Kwon. „Efficient CNF encoding for selecting 1 from N objects“. In: *International Workshop on Constraints in Formal Verification* (2007).

[KN13] E. Klotz and A. M. Newman. „Practical Guidelines for Solving Difficult Linear Programs“. In: *Surveys in Operations Research and Management Science* 18.1–2 (2013), pp. 1–17.

[Kro+08] L. Kroon, G. Maróti, M. R. Helmrich, M. Vromans, and R. Dekker. „Stochastic improvement of cyclic railway timetables“. In: *Transportation Research Part B: Methodological* 42.6 (2008), pp. 553–570.

[Kro+09] L. G. Kroon, D. Huisman, E. J. W. Abbink, P.-J. Fioole, M. Fischetti, G. Maróti, A. Schrijver, A. Steenbeek, and R. Ybema. „The New Dutch Timetable: The OR Revolution.“ In: *Interfaces* 39.1 (2009), pp. 6–17.

[Kro+14] L. G. Kroon, L. W. P. Peeters, J. C. Wagenaar, and R. A. Zuidwijk. „Flexible Connections in PESP Models for Cyclic Passenger Railway Timetabling“. In: *Transportation Science* 48.1 (2014), pp. 136–154.

[Küm+13] M. Kümmling, P. Großmann, K. Nachtigall, J. Opitz, and R. Weiß. „The State-of-the-art Realization of Automatic Railway Timetable Computation“. In: *Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transportation Systems 2013*. Ed. by T. Albrecht, B. Jaekel, and M. Lehnert. Verkehrstelematik 3. Dresden: TUDpress, 2013, pp. 397–405.

[Küm+15] M. Kümmling, P. Großmann, K. Nachtigall, J. Opitz, and R. Weiß. „A state-of-the-art realization of cyclic railway timetable computation“. In: *Public Transport* 7.3 (2015), pp. 281–293.

[Lie06] C. Liebchen. „Periodic Timetable Optimization in Public Transport“. PhD thesis. 2006.

[Lie08] C. Liebchen. „The First Optimized Railway Timetable in Practice“. In: *Transportation Science* 42.4 (2008), pp. 420–435.

[LM07a] C. Liebchen and R. H. Möhring. „The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables-and Beyond“. In: *Proceedings of the 4th International Dagstuhl, ATMOS Conference on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*. ATMOS’04. Dagstuhl Castle, Germany: Springer-Verlag, 2007, pp. 3–40.

[LM07b] C. Liebchen and R. H. Möhring. „The modeling power of the periodic event scheduling problem: railway timetables-and beyond“. In: *Proceedings of the 4th international Dagstuhl, ATMOS conference on Algorithmic approaches for transportation modeling, optimization, and systems*. ATMOS’04. Dagstuhl Castle, Germany: Springer, 2007, pp. 3–40.

[LW82] J. Y.-T. Leung and J. Whitehead. „On the complexity of fixed-priority scheduling of periodic, real-time tasks“. In: *Performance Evaluation* 2.4 (1982), pp. 237–250.

[Man15] N. Manthey. „Towards next generation sequential and parallel SAT solvers“. In: *Constraints* 20.4 (2015), pp. 504–505.

[MML14] R. Martins, V. Manquinho, and I. Lynce. „Open-WBO: A Modular MaxSAT Solver,“ in: *Theory and Applications of Satisfiability Testing – SAT 2014*. Ed. by C. Sinz and U. Egly. Vol. 8561. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 438–445.

[MMP09] V. Manquinho, J. Marques-Silva, and J. Planes. „Algorithms for Weighted Boolean Optimization“. In: *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing.* SAT'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 495–508.

[MS96] J. P. Marques-Silva and K. A. Sakallah. „GRASP – a New Search Algorithm for Satisfiability“. In: *Proceedings of the 1996 IEEE/ACM International Conference on Computer-aided Design.* ICCAD '96. San Jose, California, USA: IEEE Computer Society, 1996, pp. 220–227.

[MS99] J. P. Marques-Silva and K. A. Sakallah. „GRASP: A Search Algorithm for Propositional Satisfiability“. In: *IEEE Transactions on Computers* 48.5 (May 1999), pp. 506–521.

[MZ06] I. Mironov and L. Zhang. „Applications of SAT solvers to cryptanalysis of hash functions“. In: *Theory and Applications of Satisfiability Testing 2006.* 2006, pp. 102–115.

[Nac98] K. Nachtigall. „Periodic Network Optimization and Fixed Interval Timetable“. Habilitation thesis. University Hildesheim, 1998.

[NO07] K. Nachtigall and J. Opitz. „A Modulo Network Simplex Method for Solving Periodic Timetable Optimisation Problems“. In: *OR.* Ed. by J. Kalcsics and S. Nickel. Springer, 2007, pp. 461–466.

[NO08] K. Nachtigall and J. Opitz. „Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations“. In: *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08).* Ed. by M. Fischetti and P. Widmayer. Vol. 9. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2008.

[Odi94] M. A. Odijk. *Construction of periodic timetables, Part 1: A cutting plane algorithm.* 1994.

[ON08] J. Opitz and K. Nachtigall. „Taktfahrlagenplanung auf komplexen Infrastrukturen mittels iterativer lokaler Konfliktauflösung“. In: *Eisenbahntechnische Rundschau Nr. 6* (2008), pp. 369–373.

[Opi09] J. Opitz. *Automatische Erzeugung und Optimierung von Taktfahrplänen in Schienenverkehrsnetzen.* PhD thesis, Reihe: Logistik, Mobilitöät und Verkehr. Wiesbaden: Gabler Verlag | GWV Fachverlage GmbH, 2009, p. 293.

[PF15] D. Pöhle and M. Feil. „The Future Industrialized Timetable Process and Optimization for Rail Freight in Germany“. In: *IT15.Rail - International Railway Conference*. Zürich, 2015.

[PW14] D. Pöhle and W. Weigand. „The Importance of Automatic Timetabling for a Railway Infrastructure Company“. In: *Operations Research Proceedings 2012*. Ed. by S. Helber, M. Breitner, D. Rösch, C. Schön, J. G. von der Schulenburg, P. Sibbertsen, M. Steinbach, S. Weber, and A. Wolter. Operations Research Proceedings. Springer International Publishing, 2014, pp. 487–492.

[Sch14] M. Schmidt. *Integrating Routing Decisions in Public Transportation Problems*. Vol. 89. Optimization and Its Applications. Springer, 2014.

[SE05] N. Sörensson and N. Een. *MiniSat v1.13 - A SAT solver with conflict-clause minimization. 2005. SAT-2005 Poster. 1 Perhaps under a generous notion of "part-time", but still concurrently taking a statistics course and leading a normal life*. Tech. rep. 2005.

[Sel+15] P. Sels, K. Meisch, T. Moller, J. Parbo, T. Dewilde, D. Cattrysse, and P. Vansteenwegen. „Towards a Better Train Timetable for Denmark Reducing Total Expected Passenger Time“. In: *Proceedings of the 13th Conference on Advanced Systems in Public Transport (CASPT) 2015*. Erasmus University, 2015.

[SS10] M. Schmidt and A. Schöbel. „The Complexity of Integrating Routing Decisions in Public Transportation Models“. In: *Proceedings of ATMOS10*. Ed. by T. Erlebach and M. Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 156–169.

[SS15] M. Schmidt and A. Schöbel. „Timetabling with Passenger Routing“. In: *OR Spectrum* 37 (2015), pp. 75–97.

[SU89] P. Serafini and W. Ukovich. „A Mathematical Model for Periodic Scheduling Problems“. In: *SIAM J. Discrete Math.* 2.4 (1989), pp. 550–581.

[TTB11] T. Tanjo, N. Tamura, and M. Banbara. „A Compact and Efficient SAT-Encoding of Finite Domain CSP“. In: *SAT*. 2011, pp. 375–376.

[Vee+14] L. Veelenturf, M. Kidd, V. Cacchiani, L. Kroon, and P. Toth. *A railway timetable rescheduling approach for handling large scale disruptions*. ERS-2014-010-LIS. July 2014.

[WKO15] R. Weiß, M. Kümmling, and J. Opitz. „On Optimally Allocating Tracks in Complex Railway Stations“. In: *International Conference on Operations Research*. Vienna, Austria, presentation, 2015.

[WL08] W. Weigand and D. Lübke. *Das System Bahn*. Hamburg: DVV Media Group GmbH und DVV Rail Media (Eurailpress), 2008.

[WON12] R. Weiß, J. Opitz, and K. Nachtigall. „A novel approach to strategic planning of rail freight transport“. In: *Operations Research Proceedings 2012*. Springer, 2012, pp. 463–468.

[Wün16] L. Wünsch. *Potenziale alternativer Laufwege bei der Erzeugung und Optimierung streng getakteter Fahrpläne im Schienenpersonenverkehr*. Diploma thesis. TU Dresden, 2016.

[ZKV01] P. J. Zwaneveld, L. G. Kroon, and S. P. Van Hoesel. „Routing trains through a railway station based on a node packing model“. In: *European Journal of Operational Research* 128.1 (2001), pp. 14–33.

[Zwa+96] P. J. Zwaneveld, L. G. Kroon, H. E. Romeijn, M. Salomon, S. Dauzere-Peres, S. P. Van Hoesel, and H. W. Ambergen. „Routing trains through railway stations: Model formulation and algorithms“. In: *Transportation science* 30.3 (1996), pp. 181–194.