SBB - DSOR collaboration

Jérôme De Boeck, Bernard Ries, David Schindl

March 15, 2023

Contents

1 Introduction

Railway planning is a complex procedure divided into several sub-problems, as illustrated in Figure 1. The collaboration between the SBB and the DSOR service of the University of Fribourg aimed at studying optimization methods to help the SBB with the crew scheduling part of this process. The SBB

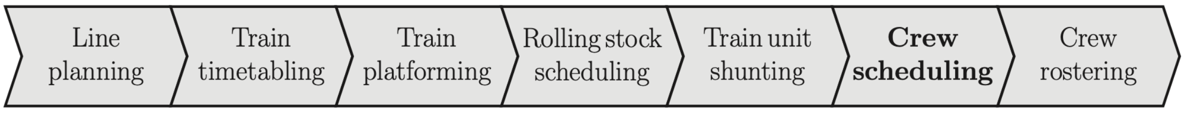

Figure 1: Railway planning process

currently performs the crew scheduling in two phases. In the first phase, Crew Requirements (CRs) that train drivers must cover are assigned to depots. In the second phase, tours are generated for each group in each depot based on the CRs assigned in the first phase to cover all CRs. The first phase is currently performed manually by the SBB. DSOR explored to what extent algorithmic methods could improve these assignments and decrease the total cost of tours generated in the second phase.

The main criteria for our research team to answer the questions of the SBB is to propose a transposable methodology to reality as much as possible. Regular meetings with the SBB have led us to a good understanding of ground rules and filter, in agreement with the SBB, those that are of the most significant importance. Significant care has been brought to legal and technical constraints included in our models to avoid correcting the solutions obtained by our methods.

This report is organized as follow. Section 2 describes the project definition, from the initial demands to the final research directions. The required data and its collection are presented in Section 3. The modelization of the data and the algorithms developed are detailed in Sections 4 and 5. Section 6 presented algorithmic numerical results and case studies before concluding in Section 7.

2 Project definition

2.1 SBB initial research questions

The crew scheduling problem is divided into two phases at the SBB based on the train timetable and the rolling stocks. Firstly, the CRs in the timetables are assigned to depots by the SBB planners. Secondly, tours for train drivers are computed for each group in each depot by the Phönix optimizer of Algomia. This approach considered the crew scheduling problem as two decoupled phases illustrated in Figure 2. The initial demand of the SBB was to study algorithmic methods to help the planners in their

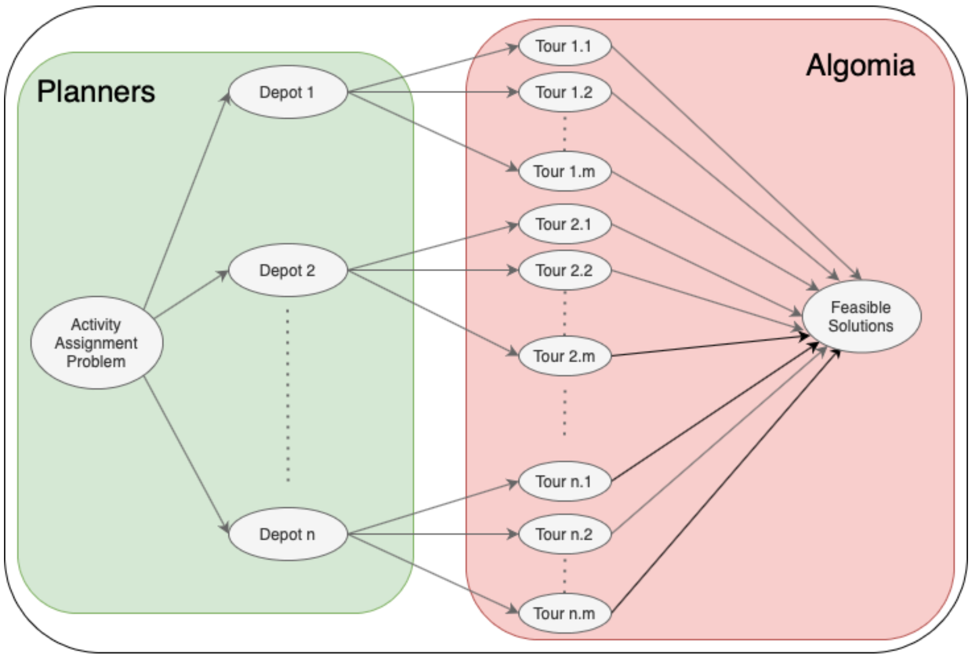

Figure 2: Decoupled crew scheduling phases

assignment task and to provide insight for long-term strategic decisions. Another underlying question was to determine what level of detail was to be considered in the algorithmic methods developed to have sufficient reliability in the decisions provided.

2.2 Timeline of the project

Jérôme De Boeck of the DS & OR service has worked full-time on this project under the supervision of David Schindl and Bernard Ries. We also hired a student to work with us for implementation purposes from 05/22 to 12/22. Online meetings with the SBB have been organized every two weeks to discuss the research directions, the data, and the assumptions made. Four meetings were organized in the offices of the SBB to present the advances of our work. Table 1 contains a timeline of the project.

Literature review

We first performed a literature review of railway crew scheduling problems. These problems have been studied for over 30 years. No paper was found considering a decoupled approach, as it is only possible to

Period	Focus	Outcome
03/22	Literature review	All literature found focuses on the coupled approach (assignment to depot and tour generation as a single problem).
29/03/22	Kick-off meeting	Discussion of the literature review and the first data delivery.
04/22 - 06/22	Discussion on the decoupled approach with the SBB	We have tried to find indicators to assess the quality of an assignment in the decoupled approach, but this research direction was abandoned in June 2022.
04/22 - 06/22	First data delivery	Significant effort to understand and correct the first data delivery before being able to use it in our algorithms.
05/22 - 07/22	First implementations	Price-and-Branch algorithm implemented and tested on the first data delivery.
29/06/22	Second meeting in Bern	Presentation of our first algorithms, no results yet as data was still being correct. The SBB mentioned the importance of developing methods leading to information for strategic decisions.
08/22 - 09/22	Case study design	Tuning of our models and algorithm for strategic decision making.
09/22 - 11/22	Second data delivery	Larger data set than first delivery. The same amount of effort was needed to correct and interpret the data.
05/10/22	Third meeting in Bern	Presentation of the case studies and numerical results on the first data set. Decision to confirm our results by testing them with Phönix.
10/22 - 12/22	Algorithmic improvements	Improvement of the implementation of our algorithms and design of an Iterative Price-and-Branch algorithm to improve the solution quality.
10/22 - 12/22	Validation with Phönix	The encoding of the input file for Phönix was not clear and tests on our input and the SBB input lead to errors. The use of Phönix to validate results was abandoned in 12/22.
12/22 - 02/22	Final tests	Numerical experiments on the second data set of the case studies.
16/02/22	Final meeting in Bern	Summary of the project, the final results, and the content of the final report.

Table 1: Project timeline

evaluate the quality of an assignment of CRs to depots by considering the possible tours of an assignment. All papers either consider the assignment is already performed or the assignment problem and the tour generation as a single problem.

Kick-off meeting

The literature review findings were presented in Bern during the kick-off meeting with the SBB, during which we discussed some alternatives. We also discussed during the kick-off meeting the first data delivery that was crucial for our research to know what size of the instances we were going to work on.

Discussion on the decoupled approach

We have had regular discussions with the SBB to find indicators to assess the quality of an assignment in a decoupled approach (without considering the generation of tours) with no success. We concluded that the planners have a significant amount of soft knowledge they use when assigning CRs to depots and do keep partial track of what tour will be feasible as in a coupled approach. The decoupled approach was abandoned in June 2022 to focus on the coupled approach traditionally used in the related literature. In this coupled approach, we assign CRs to depots by considering the possible resulting tours and the corresponding total work time as illustrated in Figure 3.

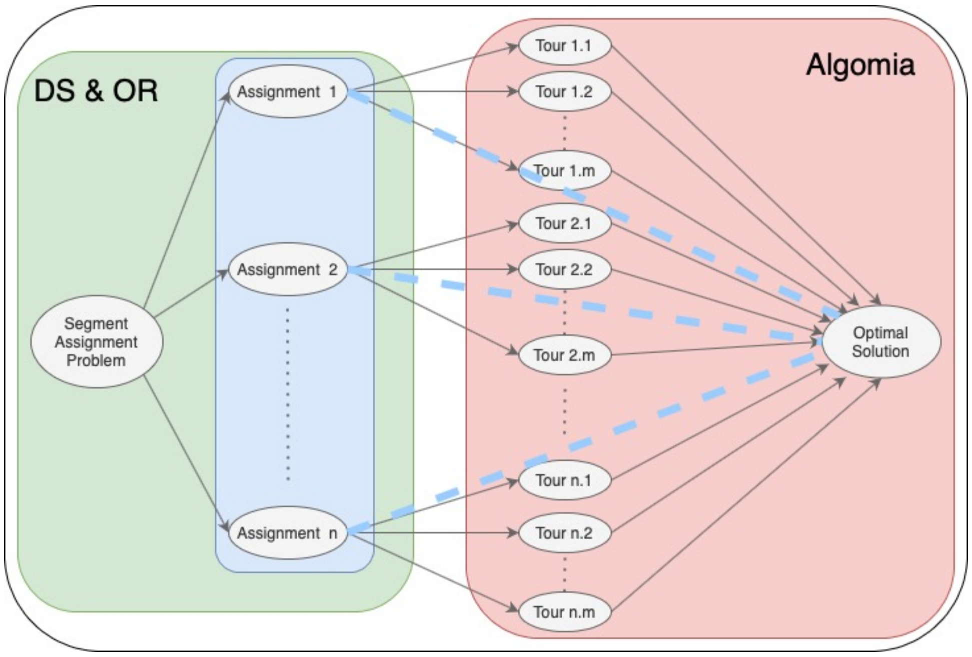

Figure 3: Coupled crew scheduling

First data delivery

The first data delivery represented a subset of all CRs and required significant work to understand its format and perform corrections. Several integrity issues were found, such as CRs requiring skills existing in no group or CRs for which a driver couldn't go back to his depot because no passride existed. With the SBB, we decided at the end of 05/22 to stop the data correction and delete all problematic CRs to focus on algorithmic developments. Section 3 provides more details on the data.

First implementations

We chose a Price-and-Branch method to solve the coupled approach treating the assignment and the tour generation as a single problem. This method minimizes the total work time needed to cover all CRs with feasible tours. The tours are generated by solving a resource constraints shortest path problem that integrates legal constraints. In the first tests performed, the tours were assigned to depots and not to groups. The first tests showed this method could not solve the problem to optimality but could find a good quality solution when considering the entire Swiss network. Section 5 provides more details on the algorithms.

Second meeting in Bern

The issues with data were presented, as well as the Price-and-Branch method. No numerical results could be provided due to the delay imposed by the data correction. The SBB insisted on designing methods that could provide information for management decisions. This led to a discussion over the different case studies that could be considered with our model. Regarding the data correction, a second data delivery was scheduled for 09/22. This data would consider all CRs of the SBB network.

Case study design

Because of a lack of indicators to measure the quality of an assignment or a set of tours, the focus was put on minimizing the total work time. One question of the SBB was to know what level of detail should be considered in our model. To this end, several levels of aggregation of CRs and groups were considered. Models have been adapted to study the limitation of group size, identify skills to develop in specific groups, and analyze the impact of using more extensively taxi and bus passrides. Section 5.4 provides more details on the case studies.

Second data delivery

The second data delivery was in the same format as the first one. Fewer errors were detected, but the correction required more time as we decided to avoid deleting CRs as in the first data delivery. The data size was increased by 50%, asking for computational improvement of our algorithmic methods. The instance size is close to the largest instances considered in the state-of-the-art literature. The output files of the Phönix optimizer containing the tours for the input files we received were also provided.

There has been a confusion regarding the files sent to us that were only detected in 12/22, and certain groups were missing from the data delivered in 9/22. When this error was detected, it was unfortunately too late to reimport the data and go through a correction procedure. The missing data did not impact the algorithmic interpretation of our results but can affect the results of the case studies. Section 3 provides more details on the data.

Third meeting in Bern

Results of the different case studies on the first data delivery were presented. We did not manage by that time to decode the input format needed for Phönix and validate our results. We estimated that the total work time was reduced by 10% compared to the assignment received in the first data delivery. The decision was taken during the meeting to focus on the correction data of the second delivery as much as possible and validate our results with Phönix.

Algorithmic improvements

The number of CRs increased by 50% on the second delivery, and the decision was taken to generate tours for groups rather than depots, making the number of graphs used go from 37 to 88. The Price-and-Branch algorithm could not fit the size of the problem. We developed an Iterative Price-and-Branch algorithm that significantly improved the solutions found with the Price-and-Branch algorithm.

Validation with Phönix

Significant effort has been put into implementing scripts that write input files based on our solutions for Phönix and repeatedly obtained output files with failed crew diagrams. After investigating the output files provided by the SBB on the second data delivery, we observed over 8000 CRs out of the 18000 contained in the SBB input files were in a failed crew diagram. We did not manage to understand what was the reason for these failures. We also detected in 12/22 assignments of CRs to depots where valid tours exist, but Phönix cannot find them. This is because Phönix uses a partitioning approach to generate tours which is algorithmically faster but can eliminate feasible tours. We use a set covering approach that does not eliminate the tours eliminated in a partitioning approach. We were thus facing the risk of generating assignments for which feasible tours exist but that Phönix could not find. For all these reasons, the validation of our results with Phönix was abandoned in 12/22. We developed an alternative method to compare the quality of our assignment to the one proposed in the second data delivery. The level of legal constraints considered in our models was considered sufficient by the SBB to consider the tours in our solutions feasible.

Final tests

We performed a final fine-tuning of our algorithms, and all case studies were tested on the second data set. The instance size required to perform tests that run between 4h and 74h. Over 800h of tests were performed. All results are available in Section 6.

Final meeting in Bern

A project summary was performed with the main elements the SBB could benefit from. The decoupled approach is limiting. Considering the assignment of CRs and the generation of tours has proven to be computationally feasible, significantly reducing the total work time. Models are there for various case studies, but the lack of accurate data limits the interpretation of numerical results. Several software limitations have been pointed out, especially regarding acquiring data and interpreting solutions. This report contains all points covered during the final meeting with more detailed information.

2.3 Research direction

A coupled approach was chosen to assign CRs to depots. The assignment of CRs and the tour generation are considered a single problem, as illustrated in Figure 3. The tours respect the legal constraints provided by the SBB and which are detailed in Section 3.3.

A set covering approach is considered in which we generate a set of tours for each depot before attempting to cover all CRs with a subset of tours minimizing some quality criteria. As no specific quantifiable indicators were provided, we considered minimizing the total work time. Based on a set of tours covering all CRs, the CRs assigned to depots are those of the tours starting at this depot. More details on the modelization of CRs in Section 4.1 and on the set covering approach in Section 4.2.

3 Data

3.1 Required data

We detail in this section all the data that is required in our models. For each element, the attributes needed are indicated as sub-bullets.

- Depots: all depots from which drivers start and end their tours
  - Name
- Locations: all stations where CRs start or end
  - Name
  - Depot: is the location a depot?
  - Break: does the location have a break room?
- Tracks: all tracks where CRs start or end
  - Name
  - Location: location of the track
  - Break: is the track a break room?
- Groups: all groups in all depots
  - Name
  - Depot: associated depot of the group
  - Skill: list of all the skills of the group
  - Maximum number of tours per day
- CRs: all CRs
  - Start/end location and track
  - Start/end time
  - CR type (train, shunting, setup-up,....)
  - Takeover and handover time
  - Rolling stock
- Rolling stocks: lists of all rolling stocks

    - Name
    - CRs: list of CRs on the same rolling stock

- Walks: all walking possibilities between tracks
    - Start/end location and track
    - Walk duration

- Passrides: all passrides drivers can take between two CRs
    - Start/end location and track
    - Start/end time

3.2 Format, corrections, and hypotheses

The SBB provided two sets of data. The first in 04/22 and the second in 09/22. On both deliveries, a JSON file was provided for each SBB depot representing an assignment of CRs to depots. Each JSON file was extracted for the software Sopre and is also used as input files for the Phönix optimizer and contained information over walks and passrides possibilities to generate tours. The format of these files has been quite complex to understand. The data described in the last section was scattered in all the JSON files, representing over 10 Gb of data on the second delivery. A significant amount of effort was put into extracting the data from all files delivered and rebuilding some missing information, such as the rolling stock.

3.2.1 First delivery

We received 36 JSON files corresponding to SBB depots in 04/22. The data delivered represented an assignment of CRs to the depots with a total of 12623 CRs. The first data delivery did not contain all CRs of the SBB as they were not available in Sopre at that time. After extracting and processing the raw data from the JSON files, we experienced several issues with our implemented algorithms. We observed several integrity issues with the data in the debugging procedure. Here is a non-exhaustive list of the issues found:

- Dead-end CRs: some early and late CRs could not be connected to their home depot as no passride exists allowing them to go to or to come to these CRs.
- Teleportations: rolling stocks had missing CRs, ending a CR in a certain train station and starting the next one in another.
- Overlap: CRs on the same rolling stock were overlapping the next one, meaning a CR was supposed to start when the previous one on the same rolling stock was not over.
- Missing skills: some CRs could not be assigned to any group of their depot as no group had the required skills.

During 05/22, we tried to correct the data with the SBB during our regular meetings by correcting CRs and adding missing CRs or passrides. In 06/22, we decided to stop the corrections to focus on algorithmic developments. The following hypotheses were made:

- Each CR without any passride allowing to get to that CR from a depot or go back to a depot from that CR is deleted.
- Rolling stocks with teleportations are corrected by adding virtual CRs.
- For each overlapping CR, one of them was deleted from the data.
- The skills necessary to cover a CR but that are missing from all groups of the depot of this CR were added to each group.

A total of 583 CRs of the 12623 were deleted from the data in the correction procedure (4.62% of the CRs). We did not identify with the SBB the reason for these integrity issues, but Phönix initially had the same problems. The corrected data was used to develop our first algorithms and present preliminary results in an intermediate report delivered for the third meeting in Bern on 5/10/22.

3.2.2 Second delivery

A second data set was delivered in 10/22. The delivery time was chosen based on when the Sopre software generating the JSON files had the most complete picture of the SBB network. The data contained 18553 CRs. The same integrity issues as in the first data delivery were observed. The format of the input files was also slightly changed without asking for some adaptation for our importation scripts. More effort was put into correcting the data than in the first data delivery to eliminate as few CRs as possible. Still, we could only correct some of the data and had to remove 233 CRs (1.26% of the CRs). The output files of Phönix for the inputs we received were also delivered. We observed that 240 CRs of the input files disappeared from the output of Phönix, and 8234 CRs were in a tour with a fail status. The 233 CRs we deleted from the data are part of the 240 CRs deleted by Phönix. Together with the SBB, we did not identify the reason for the fail status of the tours. The fact of being unable to determine why the input files of the SBB did not work for Phönix is one of the reasons why we abandoned the perspective of validating our results with Phönix.

A significant difference in the data between the first and second delivery is the number of groups in each depot (88 vs. 52). In the first data delivery, most depots contained a single group; we thus decided to perform the assignments of tours to depots rather than groups. In the second data delivery, almost all depots had several groups. This led us to redesign our algorithms so tours would be generated for groups and not depots. We received in 12/22 the number of tours of each group in the current SBB planning to consider limiting the number of tours generated for each group to their number of employees. There was a total of 1403 tours. We observed at that moment that some groups had tours assigned but did not appear in the data delivered in 10/22. The following groups covering CRs of the second data delivery were missing in our data (the number of tours in the current planning is indicated in brackets):

- BI - Gruppe 51 KUPL (2)
- BI - Gruppe BI_ZBS (5)
- BN - Gruppe BN_ZBS (3)
- BR - Gruppe BR_ZBS (3)
- BS - Gruppe 31 RES (11)
- BS - Gruppe 32 RES (7)
- BS - Gruppe BS_ZBS (8)
- CH - Gruppe CH_ZBS (1)
- CHI - Gruppe CHI_ZBS (1)
- ER - SOB-Leistung. (28)
- GE - Gruppe 33 RES (3)
- GE - Gruppe 51 KUPL (2)
- LS - Gruppe 51 KUPL (2)
- LS - Gruppe LS_ZBS (6)
- RH - Gruppe RH_ZBS (1)
- SG - Gruppe 31 RES (1)
- ZG - Gruppe 31 RES (1)
- ZUE - Gruppe 11 ETR (14)

- ZUE - Gruppe 12 TGV (18)
- ZUE - Gruppe 13 ICE (17)
- ZUE - Gruppe 15 Rail (17)
- ZUE - Gruppe 50 RES (12)
- ZUE - Gruppe 51 KUPL (8)
- ZUE - Gruppe 52 MIN (2)
- ZUE - Gruppe SUR SURV (2)
- ZUE - Gruppe ZUE_ZBS (21)

The missing groups cover 196 tours in the SBB planning of September, representing 14% of all tours. The depot in which most groups are missing is the one of Zurich and are those with specific train skills. As we did not have these groups in our data but did have the CRs they were covering, this can lead to misinterpretation of numerical results. There could exist some CRs that some missing group should cover in Zurich, but that will be covered by another group having the appropriate skills coming from a further depot, potentially increasing the total number of tours. As this error was detected only in 12/22, and there was no straightforward way to correct out data by adding the missing groups, we decided to focus on final algorithmic developments. We had to increase the size of all groups we received by 40% to find a feasible assignment.

3.2.3 Impact of corrections

The complicated format of the data and the correction of the errors took at least four months of the research project. We summarize here the main issues encountered and their consequences.

- Documentation of the format of data files:
  - No documentation of the input files asked for regular discussions with the SBB to decode their format. A lot of time had to be spent on the format rather than focusing on research.
  - The input format needed to visualize the tours of our solutions in Sopre was also unclear, which led to difficult interpretations of our solutions by SBB members who did not regularly collaborate on the project.
  - The modification in the file format we received also took some time to detect after the first tests were performed, leading to inconsistent results. Several parsing scripts had to be written to try and identify all the changes.
- Data integrity:
  - The data integrity significantly solved the implementation and debugging of our algorithms. When observing an error in our results, we naturally search for errors in our implementations. Most often, this time spent debugging our algorithms resulted in lost time because the source of the error came from the data.
  - Deciding which action to take when integrity issues were found was a time-consuming procedure as these were mainly discussed during the online meetings every 15 days.
- Instance sizes to solve:
  - The size of the instances to solve remained unknown until 06/22. Considering the difficulty of scheduling problems, the method chosen most often depends on the instance size to solve. The direction to choose for our first implementations was thus delayed.
  - The size of the complete SBB network was unknown until 10/22 on the second data delivery. Hopefully, the data size did not significantly increase, and the first implementations only

needed minor improvements. Still, if the data size was twice the size of the first delivery, the algorithms implemented during the first part of the project could have become inefficient.

- Validation method:
  - It was decided from the start of the project that Phönix would be used to confirm the quality of the tours we would find with our methodology. Unfortunately, the input format was not clear to us. We have tried several times to make the Phönix optimizer work on our results or toy inputs. We constantly obtained tours with a fail status without any further explanation.
  - Phönix uses heuristic methods to generate tours. Thus, it may not find a good solution, although it exists. We had the confirmation in 11/12 Phönix was using a partitioning approach, which could potentially be incompatible and more limiting than the set covering approach we chose. Additional information on Phönix regarding its format and computation method from the beginning would have been most helpful in avoiding compatibility issues.
  - The output files of Phönix we received on the second data delivery have feasible tours for only 55% of all CRs without any information on the failed tours. This has been misleading on how the SBB uses the Phönix optimizer and what was expected in the validation procedure of our results with Phönix.

3.3 Legal constraints, hypotheses, and objective function

Legal constraints

We considered the following set of legal constraints for a set of tours covering all CRs:

- Maximum paid time per tour: 10h
- Minimum paid time per tour: 6h
- Maximum continuous work time between breaks: 4h30
- A break lasts at least 30 minutes
- Only the first hour of break is unpaid
- Break needs to be at break room; walk duration does not count in a break
- Average paid time per group: 8h12
- All transition from a CR to another must include
  - walking times and passride times
  - handover and takeover times if the two CRs are on the same rolling stock

This represents all the legal constraints we received from the SBB, which were considered sufficient to consider the tours generated realistic.

Hypotheses

The following hypotheses are considered:

- The maximum standby time for a driver (waiting time between two CRs to cover) is set to 180 minutes (same time as Phönix)
- Train CRs that are preceded by an attendance CR on the same rolling stock are merged into a single CR
- Walking times between two tracks in the same location that are missing in our data are set to 4 minutes
- The use of taxi and bus passrides must be avoided if possible

Objective function

Several indicators to measure the quality of the assignments and the underlying tours found with our algorithms were considered:

- Minimize the total paid time
- Minimize the number of tours
- Minimize the maximum transition time of a driver between two CRs
- Minimize the number of cut-offs in tours (there is a cut-off when a driver changes rolling stock)

As an average paid time of 8h12 must be respected for all tours, minimizing the total paid time is equivalent to minimizing the number of tours (a tour over 8h12 being compensated by a tour under 8h12). For convenience reasons, we decided to minimize the total number of tours. The two other indicators were not considered further in our models. We observed that minimizing the maximum transition time can lead to solutions with a higher total paid time. This can be explained by the fact this approach tends to avoid potential transitions for drivers and reduce the possibilities in terms of tours. Minimizing the number of cut-offs was first considered to reduce the propagation of delay of a rolling stock to other rolling stocks. We did not obtain any quantifiable information on the cost of a delay or the number of delays observed on the network to consider numerical experiments. All methods presented in the following sections can be adapted to minimize the number of cut-offs.

4 Modelization of the problem

We present in this section how the collected data was modeled on graphs and the set covering that was used to minimize the total number of tours covering all CRs. Table 2 provides the notations used in our models and algorithms and are defined more in detail in the following sections.

4.1 Graph modelization

The CRs are modeled on graphs where tours satisfying legal constraints will be searched. Several types of graphs used are detailed in this section.

T	Set of all tasks
D	Set of all depots
G	Set of all groups
P^d	Set of all tours in the depot graph $d \in D$
$\tilde{P}^d$	Set of tours considered in the RMP for depot graph $d \in D$
c_t	Cost of task t
c_{uv}	Cost of arc uv
t_t	Time of task t
t_{uv}	Time of arc uv
t_p^d	Time of tour $p \in P^d$
T_{max}^d	Maximum number of tours in depot $d \in D$
t^{ave}	Average work time (8h12)

Table 2: Notations

4.1.1 Crew requirements graphs

All CRs are modeled in a crew requirement graph illustrated in Figure 4. Each node of this graph

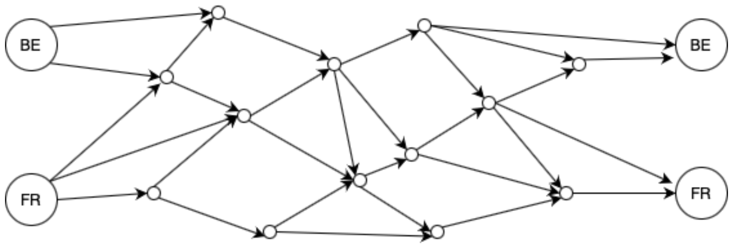

Figure 4: Crew requirement graph

represents a CR, also called a task in the following. The set of all CRs (of all tasks) is denoted T. All the information on CR in Section 3.1 are associated with nodes (start and end location and track, start and end time,...). The main information used in our algorithms for a task $t \in T$ is its total time t_t. Each arc represents a possible transition between two CRs. There exists an arc uv only if task u ends before the start time of v. The time of a transition t_{uv} is the difference between the start time of v and the end time of u is at most 3h. A driver must also be able to get from task u to task v by one of the transition possibilities illustrated in Figure 5. A driver can stay on the same rolling stock if the next CR is in less than 3h. A driver can also go to another CR starting in the same location by waiting before or after walking to the right track. Finally, a driver can take a passride to another location to cover his next CR and have a waiting period before and after his passride. During each waiting period, a driver can take a 30-minute break if he is in a location with a break room and enough time to walk to it. For each arc, we consider as break possibilities the earliest and latest options as illustrated in Figure 6. If a driver takes a break during a transition and has several break options, the latest one respecting the maximum continuous work time of 4h30 is always chosen.

In Figure 4, additional nodes were added before and after the graph and labeled BE and FR. These nodes represent the departure of a driver from a depot on the left-hand side and the return to a depot

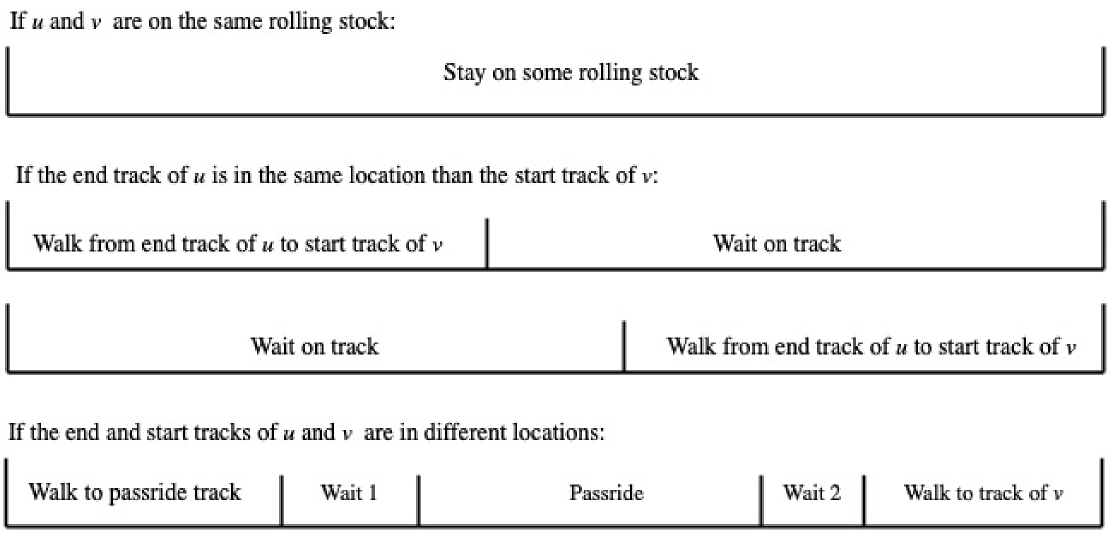
If u and v are on the same rolling stock:
Stay on some rolling stock
If the end track of u is in the same location than the start track of v:
Walk from end track of u to start track of v
Wait on track
Wait on track
Walk from end track of u to start track of v
If the end and start tracks of u and v are in different locations:
Walk to passride track
Wait 1
Passride
Wait 2
Walk to track of v

Figure 5: Transition possibilities

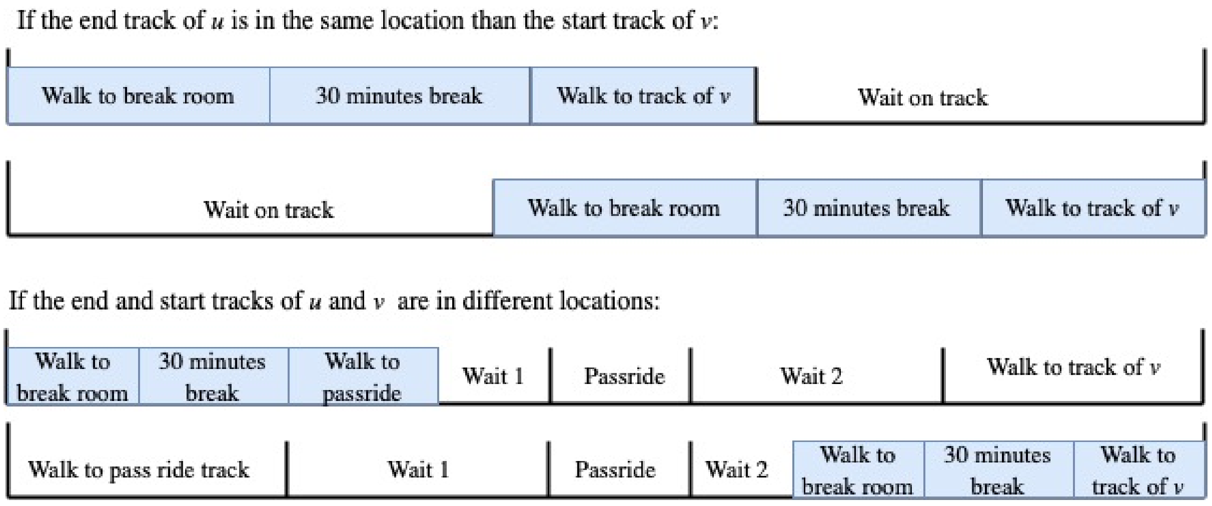

Figure 6: Transition possibilities

on the right-hand side. The nodes representing the start from a depot are connected to each task t

- either staring at the same depot, in which case the cost of the arc of the cost of walking from the main track to the track of t,
- or for which there exists a passride from the depot to t. The cost of the arc is then the cost of the passride, the walk to the track of t, and the waiting time before t starts.

Arcs connected to the nodes representing the return to a depot have symmetrical properties.

Different levels of aggregation of tasks have been tested. Consider two consecutive CRs u and v on the same rolling stock. Two configurations were mainly tested in which u and v are merged as follow:

- *full*: if u is an attendance task, v is a train task, and $c_{uv} = 0$.
- *aggregated* : if c_{uv} was greater than the handover time of u or the takeover of v.

The full and aggregated configurations reduce the number of tasks by respectively 25% and 40% averaging on both data deliveries.

4.1.2 Depot and group graphs

The tours searched must start and end in the same depot. To this purpose, we considered a depot graph for each depot $d \in D$ as illustrated in Figure 7. A depot graph contains only the tasks for which there

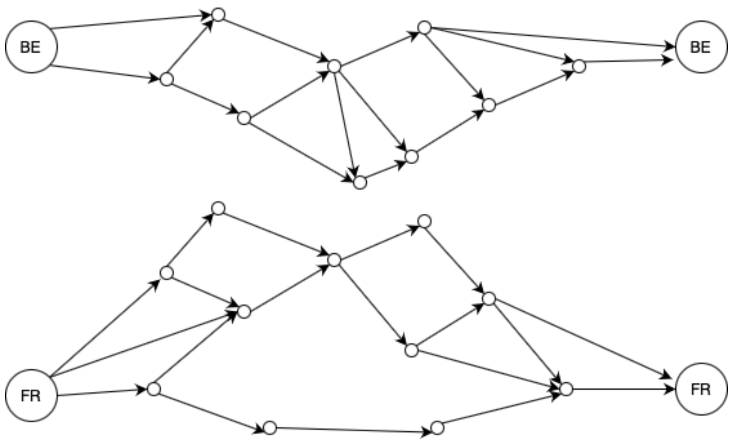

Figure 7: Depot graphs

exists one group with all the skills to cover this task and if there exists a tour covering it respecting legal constraints. Each task appears in all depot graphs it can be assigned to. Their size is, on average, 80% smaller than the crew requirement graph. The tours covering all tasks of the crew requirement graph are searched in depot graphs. In a depot containing several groups, there is a risk that a path from the start to the end depot might jump from one group to another. To consider a higher level of precision, we also considered a group graph for each group $g \in G$ alternatively. These are similar to depot graphs but are specific to groups. Nodes in group graphs correspond to tasks for which the group has the required skills. Their sizes are slightly smaller than for depot graphs, but the number of graphs is larger. Details on the graphs' size can be found in Section 6.1.

4.1.3 Original graphs

For comparison purposes, we will also consider the original graphs corresponding to the assignment of CRs to depot the SBB provided in the data deliveries. An original graph is defined for each group and contains all the tasks assigned to its depot for which the group has the required skills. These graphs contain, on average, only 3% of tasks of the CR graph as each task appears only in the original graphs of the depot it has been assigned to. We run our algorithms on these graphs to compare the number of tours generated with our method compared to the number that would be generated on the original assignment.

4.2 Set covering problem

We aim to cover the crew requirement graph by tours generated in the depot graphs. Figure 8 illustrates a covering of the CRs of Figure 4 with two tours in the depot of Bern and one in the depot of Fribourg. The tours in a depot graph $d \in D$ are denoted P^d. Throughout all following sections, depot graphs

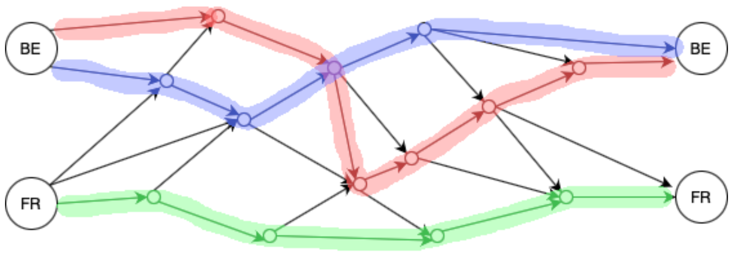

Figure 8: Cover of the crew requirement graph

can be interchanged with group graphs. All notations are identical for depot graphs and are indexed by g instead of d. Notice this covering approach allows over coverage of tasks as illustrated in Figure 8 in which a CR is covered by the blue and red tours. If several tours cover a task, the task can be assigned to one of the drivers of the tours covering this task at random, and other train drivers will use this task as a passride. The covering approach can thus allow multiple consecutive passrides in the case of over coverage. As explained in Section 3.3, we aim at minimizing the total number of tours, which is equivalent to minimizing the total work time. Allowing over coverage cannot degrade the quality of the best solution found and is thus not restrictive.

The Set Covering Problem (SCP) is a classical combinatorial optimization problem that minimizes the total cost of sets covering a graph $G = (V, A)$. The sets are subsets of vertices to choose from in a set S; each set s has an associated cost c_s.

$$\begin{aligned} \min \quad & \sum_{s \in S} c_s x_s && && (1) \\ \text{s.t.} \quad & \sum_{s \in S : v \in s} x_s \geq 1 && v \in V && (2) \\ & x_s \in \{0, 1\} && s \in S && \end{aligned}$$

The graph to cover is the crew requirement graph and all its tasks T. The sets that cover the graphs are tours $p \in P^d$ in each depot graph $d \in D$. The cost of each set is equal to 1 as we minimize the number of tours. We are facing a constrained version of the SCP as there are additional constraints to consider to enforce the average work time t^{ave} (8h12) and the maximum number of tours per depot T^d_{max} for each depot $d \in D$. This results in the following Constrained SCP formulation:

$$
\begin{aligned}
\text{(CSCP)} \quad \min \quad & \sum_{d\in D}\sum_{p\in P^d} x^d_p && && (3)\\
\text{s.t.} \quad & \sum_{d\in D}\sum_{p\in P^d : t\in p} x^d_p \geq 1 && t \in T && (4)\\
& \sum_{p\in P^d} x^d_p \leq T^d_{max} && d \in D && (5)\\
& \frac{\sum_{p\in P^d} t^d_p x^d_p}{\sum_{p\in P^d} x^d_p} \leq t^{ave} && d \in D && (6)\\
& x^d_p \in \{0,1\} && d \in D, p \in P^d &&
\end{aligned}
$$

Constraint (4) enforces all tasks are covered, constraint (5) limits the maximum number of tours in each depot, and constraint (6) limits the average work time in each depot (t^d_p is the total time of $p \in P^d$). Notice our CSCP formulation does not force the average time to be equal to 8h12 but only enforces an upper bound on it. In numerical experiments, we observe the average work time is naturally pushed to its maximum value of 8h12 to reduce the number of tours.

Formulation CSCP contains two main difficulties. First, the number of variables is not scalable because it should consider all possible tours in all depot graphs. We will thus only consider a subset of all tours based on their quality with methods detailed in Section 5.1. Secondly, because of constraints (5) and (6), no classical heuristic solving the SCP can be applied. Section 5.3 proposes a heuristic method for large instances.

Another classical approach to crew scheduling problems is a partitioning approach:

$$
\begin{aligned}
\min \quad & \sum_{s\in S} c_s x_s && && (7)\\
\text{s.t.} \quad & \sum_{s\in S : v\in s} x_s = 1 && v \in V && (8)\\
& x_s \in \{0,1\} && s \in S &&
\end{aligned}
$$

This approach is similar to a set covering approach, except each task must be covered exactly once. Constraints (8) is very convenient for solvers in their preprocessing techniques (e.i. equalities equal to 1) but is more restrictive than for the SCP and reduces the solution space, potentially eliminating some optimal solutions. For instance, the crew requirement graph in Figure 9 admits a feasible solution under a set covering approach but not under a partitioning approach, as the node in the node in the middle of the graph has to be over-covered to have a covering of all tasks. Partitioning methods need a high number of arcs to compensate for its drawbacks on the set covering approach. The passrides we received in the data deliveries do not represent all possible passrides but were fine-tuned for the assignment of the input files. A lot of possible passrides are thus missing, reason for which the set covering approach was chosen. Still, the algorithms presented in the following section can be adapted if a partitioning formulation is used.

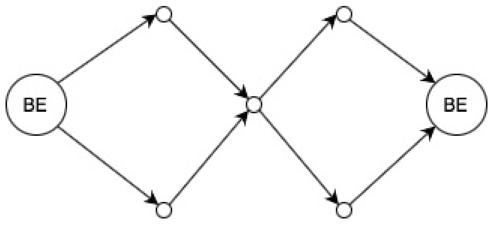

Figure 9: Crew requirement graph with no solution under a partitioning approach

5 Algorithms

We present in this section the different algorithms developed to solve the CSCP modeling the crew scheduling problem.

5.1 Price-and-Branch

As previously mentioned, a difficulty of the CSCP is its large number of variables corresponding to all possible paths in the SBB network. Such problems can be tackled with a Price-and-Branch method. The full CSCP is, in this case, called the master problem, a problem that contains too many variables. We work on a reduced version of the CSCP called the Restricted Master Problem (RMP), containing only a subset of the variables of the master problem. Variables are added to the LP relaxation of the RMP by solving a pricing problem that identifies variables that can improve the current optimal value of the LP relaxation of the RMP. This corresponds to a column generation procedure in which columns correspond to the variables added to the RMP. Once no new column is found, the optimal value of the LP relaxation of the master problem and RMP is identical. The integer version of the RMP is then solved through a Branch & Bound procedure. A flow chart of a general Price-and-Branch method is given in Figure 10.

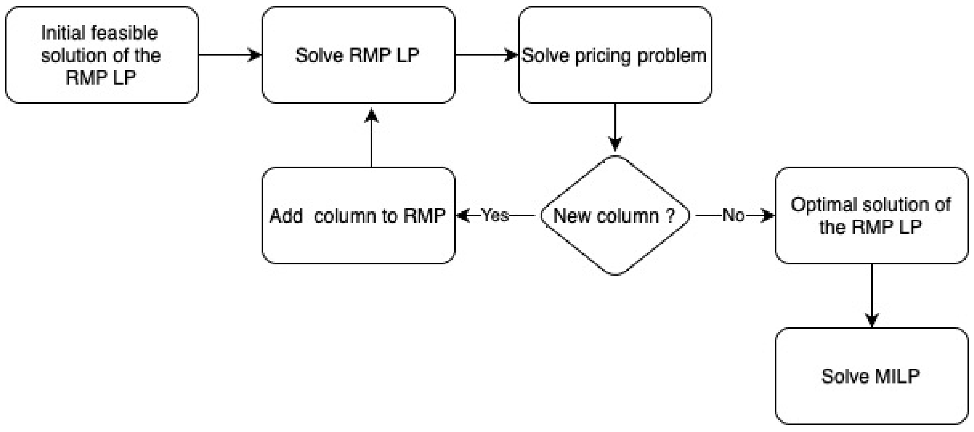

Figure 10: Generic Price-and-Branch flowchart

Consider subsets of tours $\tilde{P}^d$ of P^d for each depot for which a solution exists to the LP relaxation of the RMP. The RMP can be written as follow (same formulation as the CSCP written in standard form):

$$
\begin{aligned}
\text{(RMP)}\quad \min\quad & \sum_{d\in D}\sum_{p\in \tilde{P}^d} x_p^d \\
\text{s.t.}\quad & \sum_{d\in D}\sum_{p\in \tilde{P}^d : t\in p} x_p^d \geq 1 && t\in T \quad (u_t)\\
& -\sum_{p\in \tilde{P}^d} x_p^d \geq -T_{max}^d && d\in D \quad (v_d)\\
& \sum_{p\in \tilde{P}^d} (t^{ave} - t_p^d) x_p^d \geq 0 && d\in D \quad (t_d^{max})\\
& x_p^d \in \{0,1\} && d\in D, p\in \tilde{P}^d
\end{aligned}
$$

The dual variable of each constraint is indicated in brackets next to it. Variables x_p^d that are not in any set $\tilde{P}^d$ but can improve the LP relaxation of the RMP can be identified based on their reduced cost. The reduced cost of a variable indicates the impact on the objective function of the variable was increased. We are thus searching for variables x_p^d with a negative reduced cost. These reduced costs can be obtained based on the dual constraint of variable x_p^d and the value of optimal dual variables. The dual constraint of x_p^d is:

$$\left(\sum_{t\in p} u_t\right) - v_d + (t^{ave} - t_p^d) t_d^{max} \leq 1$$

The reduced cost of x_p^d given an optimal solution of the LP relaxation of the RMP is obtained by placing each term of the dual constraint of x_p^d on the right-hand side:

$$t_d^{max} t_p^d - \left(\sum_{t\in p} u_t\right) + v_d - t^{ave} t_d^{max} + 1 \tag{9}$$

The current optimal solution of the LP relaxation of the RMP can be improved if we can find a tour for which (9) is negative. Searching for these tours consists of the pricing problem and is performed by solving a Ressource Constrained Shortest Path Problem (RCSPP). Tours with a negative reduced cost are searched in each depot graph. This task can be parallelized as the RCSPP is independent for each depot graph. The RCSPP problem and the algorithm used to solve it are detailed in Section 5.2. If new tours are found, they are added to the RMP before iterating the solving of its relaxation and price problem. If no new tour is found, there does not exist any tour that can improve the LP relaxation of the RMP, and the integer RMP is solved by Branch & Bound. Note that a Price-and-Branch approach does not guarantee to find an optimal solution to the integer problem. To such end, a Branch-and-Price approach could be considered, but the size of the instances considered is much too large to consider such a method.

The Price-and-Branch procedure illustrated in Figure 10 needs to start with a set of tours containing a feasible solution of the LP relaxation of the RMP. To find this feasible set of tours, we consider the following Soft Master Problem (SMP) that can start with an empty tour set for each depot ($\tilde{P}^d = \{\}$ for $d \in D$) and no variables x_p^d:

$$
\begin{aligned}
\text{(SMP)} \quad \max \quad & \sum_{t \in T} y^t & & & \\
\text{s.t.} \quad & \sum_{d \in D} \sum_{p \in \tilde{P}^d : t \in p} x_p^d \geq y_t & & t \in T \quad (u_t) & (10)\\
& - \sum_{p \in \tilde{P}^d} x_p^d \geq -T_{max}^d & & d \in D \quad (v_d) & \\
& \sum_{p \in \tilde{P}^d} (t^{ave} - t_p^d) x_p^d \geq 0 & & d \in D \quad (t_d^{max}) & \\
& x_p^d \in \{0,1\} & & d \in D, p \in \tilde{P}^d & \\
& 0 \leq y_t \leq 1 & & t \in T &
\end{aligned}
$$

The SMP is similar to the RMP with an additional set of variables y_t for each task $t \in T$ representing the coverage of a task. In right handside of constraints (10), we replaced 1 by y_t. This allows having solutions of the SMP that do not cover all tasks. To fall back on the RMP, all variables y_t should be set to 1. To this end, we maximize the sum of variables y_t and aim to find a solution with an objective equal to $|T|$ in which all tasks are covered. The pricing problem generates tours in the same way as for the RMP based on the reduced cost of variables x_p^d, which for the SMP is equal to

$$
t_d^{max} t_p^d - \left(\sum_{t \in p} u_t \right) + v_d - t^{ave} t^{max}.
$$

The full Price-and-Branch algorithm for the CSCP is illustrated in Figure 11. We first generate tours for the LP relaxation of the SMP to find a feasible solution for the LP relaxation of the RMP. Tours are then generated to find the optimal solution of the LP relaxation of the RMP. Once all tours are generated, the integer version of the RMP is solved by Branch & Bound.

5.2 Ressource Constrained Shortest Path Problem

5.2.1 General algorithm

The Ressource Constrained Shortest Path Problem (RSCPP) is a Shortest Path Problem (SPP) in which a limited amount of resources are available to perform a path. A simple example of resource for a SPP would be to limit the number of nodes visited during a path. While the SPP is polynomial, the RCSPP is a NP-hard problem. In the SPP, each vertex has a single label (containing the distance to the node and its predecessor). In the RCSPP, each node can have an exponential number of labels representing all the states in which a node can be reached. The basic mechanisms of the SPP and the RCSPP are similar. Each time a node v is visited, all labels of its predecessors are examined, and only the best possibilities to reach v are used as new labels for v.

The RCSPP is solved on depot graphs in which tours are paths going from the depot start node to the depot return node. As shown in Figure 7, depot graphs are acyclic graphs in which the nodes can be ordered by start time. A state s of a node v represents a partial path and is defined by a vector of resources $(c_s, t_s, t_s^b, b_s, pred_s)$ in which

- c_s is the cost of the partial tour

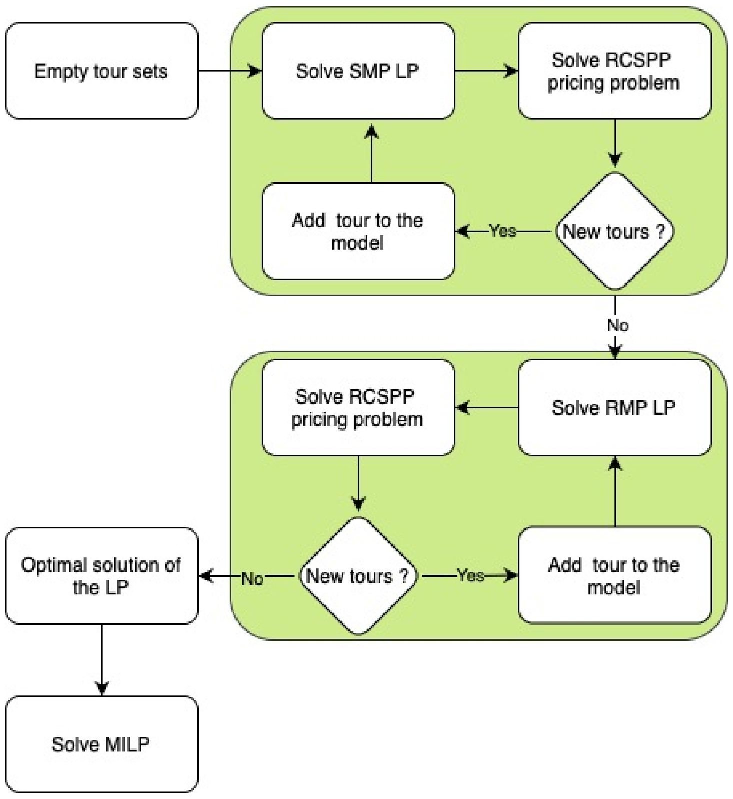

Figure 11: CSCP Price-and-Branch flowchart

- t_s is the total paid time of the partial tour and cannot exceed 10h
- t_s^b is the time spent working since the last break and cannot exceed 4h30
- b_s is the number of breaks already taken in the partial tour
- $pred_s$ is the predecessor state of s

The state of the first node of the path is set to $(0, 0, 0, 0, -)$. Nodes are parsed by starting time. Consider a node v and one of its predecessors u. For each state of u, a new state is defined for v by extending the partial path to u to v if the resources are not exceeded. Another state can be defined if arc uv has a break possibility. If a break is taken on an arc, the latest possibility is always chosen (see Section 4.1.1). A state s dominates a state s' ($s \leq s'$) if all resources of s are smaller or equal to those of s'. When generating states, only non-dominated states are added to nodes. We consider for now the cost c_t of tasks and of arc c_{uv} are equal to their time. For each arc, if the driver can take a break (there is a break room on the arc, and it can be reached before the maximum working time of 4h30), we denote by t_{uv}^- the time needed to reach the breakroom and by t_{uv}^+ the time needed to reach the next CR after the break (so $t_{uv} = t_{uv}^-$ + 30 minutes + t_{uv}^+). The set of all states of a node v is denoted $\mathcal{S}_v$. Furthermore let, $\delta^-(v)$ be the set of arcs entering v. Algorithm 1 gives a pseudo-code of the RCSPP.

Algorithm 1 returns a single path, but $\mathcal{S}_t$ contains several states and the algorithm can return the n first shortest paths $\mathcal{S}_t$. Returning more than one path allows to generate more tours in a single run of the RCSPP, but all are not necessarily needed to find an optimal solution of the LP relaxation of the

Algorithm 1: RCSPP for depot graphs

```
Data: Depot graph $G = (V, A)$, start node $s$, end node $t$
Result: Minimum cost path considering breaks
Initialize state set for $v \in V$, $\mathcal{S}_v \leftarrow \{\}$;
Initialize states of $s$, $\mathcal{S}_s \leftarrow \{(0, 0, 0, 0, -)\}$;
$t \leftarrow$ last vertex of $V$;
for $v \in V \backslash \{s\}$ ordered by start time do
    for $uv \in \delta^-(v)$ do
        for $s \in \mathcal{S}_u$ do
            $c_{s'} \leftarrow c_s + c_u + c_{uv}$;
            $t_{s'} \leftarrow t_s + t_u + t_{uv}$;
            $t^b_{s'} \leftarrow t^b_s + t_u + t_{uv}$;
            if $t^B_{s'} \leq 4h30$ (create new state without break) then
                $s' \leftarrow (c_{s'}, t_{s'}, t^B_s, B_s, s)$;
                if $T_{s'} \leq T^{max}_v$ and $\nexists s'' \in \mathcal{S}_v : s'' < s'$ then
                    remove from $\mathcal{S}_v$ states dominated by $s'$;
                    $\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{s'\}$;
            if uv has a break possibility (create new state with break) then
                if $t^B_s + t^{B-}_{uv} \leq 4h30$ then
                    $t^B_{s'} \leftarrow t^{B+}_{uv}$;
                    if $B_s < 2$ then
                        $c_{s'} \leftarrow c_{s'} - 30\ min$;
                        $t_{s'} \leftarrow c_{s'} - 30 min$;
                    $s' \leftarrow (c_{s'}, t_{s'}, t^b_{s'}, B_s + 1, s)$;
                    if $T_{s'} \leq T^{max}_v$ and $\nexists s'' \in \mathcal{S}_v : s'' < s'$ then
                        remove from $\mathcal{S}_v$ states dominated by $s'$;
                        $\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{s'\}$;
return $S \in \mathcal{S}_t$ with minimum $c_S$
```

RMP. When solving the RCSPP in the pricing problem of the Price-and-Branch, we decided to retrieve the 5 shortest paths found.

5.2.2 Pricing problem

In the previous section, the cost and time of nodes and RCSPP were identical. This will no longer be the case when solving the pricing problem generating new tours for the RMP. Recall the reduced cost of a tour x_p^d in the RMP:

$$t_d^{max} t_p^d - \left(\sum_{t \in p} u_t \right) + v_d - t^{ave} t_d^{max} + 1$$

The pricing problems aim to find tours in each depot graph with the most negative reduced cost. If costs are equal to times, t_p^d represents the total time of the shortest tour obtained when solving the RCSPP. By setting the cost to time and multiplying these costs by t_d^{max} and removing from the cost of each node its dual variable u_t, the RCSPP will find the tour minimizing $t_d^{max} t_p^d - \left(\sum_{t \in p} u_t \right)$. As $v_d - t^{ave} t_d^{max} + 1$ is independent from the shortest path found, these values can be added to the solutions of the RCSPP. If the value obtained is negative, a path with a negative reduced cost has been found and is added to the RMP.

Note that the RCSPP can return tours shorter than the minimum paid time of 6h, but such tours are generally not considered in the pricing problem. As the value of variables u_t are removed from the cost of each node, their cost can become negative. The RCSPP will, at some point, aim at finding long tours that pass by many nodes to have the smallest cost. Practically, once the LP relaxation of the SMP is solved to optimality, we remove the tours with a total paid time under 6h from the SMP. If the solution is no longer feasible, we perform a new iteration of the pricing problem on the SMP until all tours in a feasible solution are at least 6h long.

Note also that we impose a break time of 30 minutes, whereas, in reality, a break can last longer. Still, the RCSPP will generally return tours with 2 breaks of 30 minutes, reaching the maximum unpaid time for breaks.

5.3 Iterative Price-and-Branch

We will see in numerical results that the number of tours generated in the column generation is too large for a MILP solver to handle. As no classical heuristic for the SCP from the literature can be applied because of the additional constraints in the CSCP, we developed an Iterative Price-and-Branch (IPB) that aims at finding an integer solution by working on a subset of columns.

The flowchart of the generic version of the IPB is provided in Figure 12. The IPB needs a feasible solution and a tour pool that can be found by solving the column generation of the RMP. The start solution can be found by solving the integer version of the RMP until a feasible solution is found (generally a couple of seconds). The IPB will then work on a RMP of limited size. The initial columns of the RMP are those of the initial solution. This guarantees that the RMP always has a feasible integer solution. We then run several iterations of the pricing problems until 2000 tours are found. The pricing problem searches for tours in the initial tour pool, so only the reduced cost of all existing tours is inspected, and no RCSPP is solved at this stage. Once 2000 tours are added, we solve the integer version of the RMP until an exit criterion is met. The RMP is solved for at least 5 minutes and exits if a new solution is

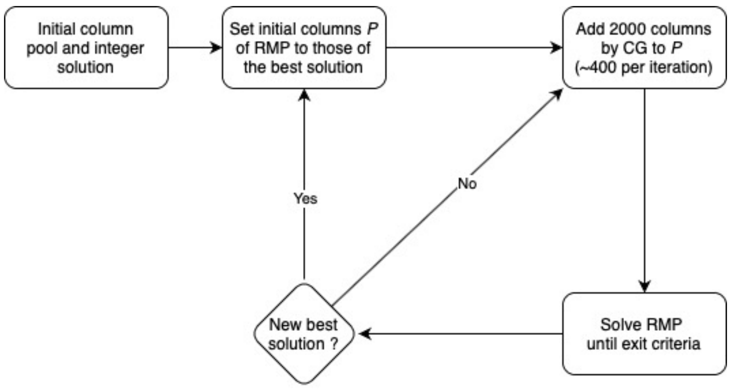
Initial column pool and integer solution
Set initial columns P of RMP to those of the best solution
Add 2000 columns by CG to P (~400 per iteration)
Yes
No
New best solution ?
Solve RMP until exit criteria

Figure 12: Generic Iterative Price-and-Branch flowchart

found or if the gap in the Branch & Bound tree falls under 10%. If no new best solution is found, we return to the pricing problem and add another 2000 tours. If a new best solution is found, the RMP is reset. The initial tours of the RMP are set to those of the new best solution, the old tours go back into the tour pool, and we iterate the procedure. The main idea behind the IPB is that the pricing problem finds the best columns based on the current solution. We thus try to iteratively improve our solution based on a limited number of paths in a close neighborhood to the current best solution.

The IPB can be applied to any problem solved by Price-and-Branch. We propose two variants of the IPB that are specific to the CSCP we are solving. Figure 13 illustrates the first variant of fixing tours. In this variant, the exit criteria of the RMP is a maximum solving time of 5 minutes. If no new best solution is found, some tours of the current best solution are fixed. The columns are fixed based on their time lost (time lost = total paid time - time waiting between two CRs) from the smallest time lost to the largest if they have no overlapping CRs. The tour with the smallest time lost is always fixed when fixing tours. The second tour that is fixed is the second having the smallest time lost and having no overlapping task with the other tours fixed during the current iteration. Tours are fixed only if they have a time lost under 2h30. Once a tour is fixed, it stays fixed until the end of the IPB.

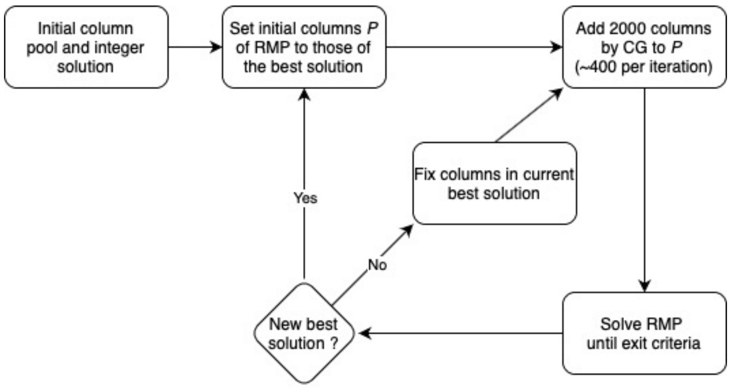
Initial column pool and integer solution
Set initial columns P of RMP to those of the best solution
Add 2000 columns by CG to P (~400 per iteration)
Fix columns in current best solution
Yes
No
New best solution ?
Solve RMP until exit criteria

Figure 13: Iterative Price-and-Branch with column fixing flowchart

The second variant of the IPB is identical to the first one except that when a new best solution is found, all tours previously fixed are unfixed. This variant will be more time-consuming than the first one but will enable a larger solution space search. The flowchart is provided in Figure 14.

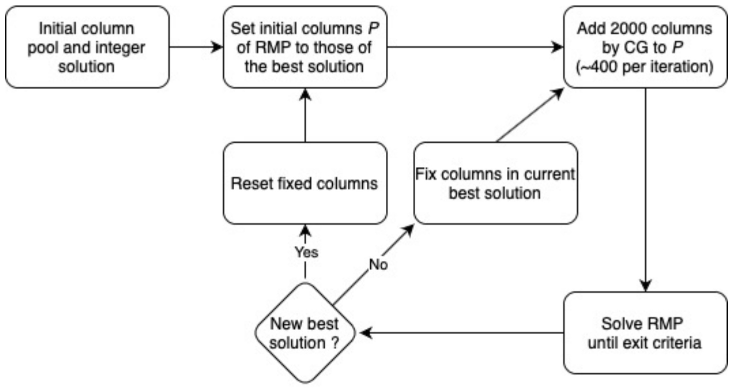
Initial column pool and integer solution
Set initial columns P of RMP to those of the best solution
Add 2000 columns by CG to P (~400 per iteration)
Reset fixed columns
Fix columns in current best solution
Yes
No
New best solution ?
Solve RMP until exit criteria

Figure 14: Iterative Price-and-Branch with column fixing and resetting flowchart

5.4 Case studies

Several case studies for the crew scheduling problem are detailed in this section.

1. Analyse the level of details needed in a model to obtain reliable solutions
2. Minimizing the total number of tours based on existing skills, groups, and passrides as described up to now.
3. Analyze which group size should be increased or decreased to reduce the total working time.
4. Skill development
    - The development of routing skills is tested by considering depot graphs containing all task for which the depot has the required routing skills. The tours generated can, in this case, contain missing routing skills that can be potentially interesting to develop.
    - An alternative model has been developed to identify specific skills that would be interesting to develop in specific groups. Let S^g be the set of skills of group $g \in G$ and λ_s^g the cost of developing skill s in group g ($\lambda_s^g = 0$ if $s \in S^g$). Consider an additional decision variable z_s^g representing whether skill s should be developed in group g. We obtain the following formulation:

$$
\begin{aligned}
\min \quad & \sum_{g\in G}\left(\sum_{p\in P^g} x_p^g + \sum_{s\in S}\lambda_s^g z_s^g\right) && && (11)\\
\text{s.t.} \quad & \sum_{g\in G}\sum_{p\in P^g : t\in p} x_p^g \geq 1 && t\in T \quad (u_t) && (12)\\
& -\sum_{p\in P^g} x_p^g \geq -T^g_{max} && g\in G \quad (v_g) && (13)\\
& \sum_{p\in P^g}(t^{ave} - t_p^g)x_p^g \geq 0 && g\in G \quad (t_g^{max}) && (14)\\
& \sum_{p\in P^G : \exists t\in p, s\in S_t} x_p^g \leq |P^g| z_s^g && g\in G, s\in S \quad (k_s^g) && (15)\\
& x_p^g \in \{0,1\} && g\in G, p\in P^g && (16)\\
& z_s^g \in \{0,1\} && g\in G, s\in S &&
\end{aligned}
$$

Constraints (15) forces $z_s^g = 1$ if group g is assigned a tour in which skill s is required. The reduced cost of a tour x_p^d can be computed similarly than in Section 5.1. We could not test this model for all groups and skills at once considering the size of the formulation (each group graph could potentially contain all tasks of the CR graph) and because the cost of developing a skill λ_s^g could not be quantified. To test this model, specific groups and potential skills interesting to develop should be identified.

5. Evaluate the efficiency of opening new groups G'. The objective function of the RMP can be replaced by $\sum_{g\in G}\sum_{p\in P^g} x_p^g + \sum_{g\in G'}\mu_g\sum_{p\in P^g} x_p^g$, putting a penalty μ_g on tours assigned to a new group $g\in G'$.

6. Determine whether using additional bus and taxi passrides can reduce the total cost. This can again be performed by changing the objective function to $\sum_{g\in G}\sum_{p\in P^g}(1+\alpha t_p + \beta b_p)x_p^g$ in which t_p and b_p is the number of taxi and bus passrides on tour p and α and β are penalties for each taxi and bus passride.

6 Numerical results

We present in this section the algorithmic efficiency of our methods and the results of the case studies. All numerical results reported in this section have been performed on the data provided in the second data delivery. Numerical experiments were run on a 24-core Inter I9 processor 3.2 GHz with 128 Gb of RAM. All the code has been implemented in Julia 1.7.3, mathematical models are solved with Gurobi 9.5.2.

6.1 Instances

Several different configurations of the data are tested. The parameters of each instance are as follow:

- Aggregation of CRs: full (F) or aggregated (A) as described in Section 4.1.1
- Type of graph: original graph (O), depot graph (D), or group graph(G)

- Type of skill: all (A) or train (T). This represents the skills required for a CR to be assigned to a depot or group. In the case of the train skills, all routing skills are ignored.

A three-letter code identifies each instance based on its parameters. For example, *FGA* is the instance considering Full CRs (attendance aggregated to the following train CR), Group graphs, and All train skills. The size of the CR graph and the subgraphs (depot graphs, group graphs, or original graphs) of the instances used are given in Table 3.

Instance	Ave. CR graph size		Nb. of	Ave. subgraph size	
	Nodes	Arcs	subgraphs	Nodes	Arcs
FOA	12217	10583546	88	613	95407
FDA	12217	10583546	37	2730	612391
FGA	12217	10583546	88	2406	461242
ADA	9438	6555032	37	2103	316419
ADT	9438	6555032	37	3202	437473
AGA	9438	6555032	88	1965	307692
AGT	9438	6555032	88	3184	439319

Table 3: Instance sizes

The are 18553 CRs in the initial data. By merging the attendances with the following train CR, this number falls to 12217 CRs. When aggregating all CRs separated by less than their takeover or handover time, the number of CRs falls to 9438. The density of the CR graph is very high, with, on average, 866 exiting arcs per node in the full configuration and 695 in the aggregated configuration.

The sizes of the depot and group graphs are similar. Group graphs are slightly smaller as they contain only tasks assignable to the corresponding group and not those assignable to other groups in the same depot. Considering only train skills are required increases the size of subgraphs by 50%.

Instance FOA represents the original assignment of the data delivered. Each task only appears in the graphs of the depot to which it has been assigned. The number of nodes in each subgraph is significantly smaller than for other instances in which the same task can appear in several different subgraphs corresponding to different depots. The average number of nodes exiting each node in subgraphs is at most 224, for instance, FDA.

6.2 Algorithmic performance

In this section, we present the column generation performance and the MILP of the Price-and-Brand method in Sections 6.2.1 and 6.2.2. The results of the IPB are presented in Section 6.2.3. The performance of all algorithms is similar for each type of instance considered (except FOA which is much smaller); we present in this section only the results of instance AGA (Aggregated tasks, Group graphs, All skills must be respected), instance for which we obtained the best results.

6.2.1 Tour generation

Figure 15 illustrated the evolution of the number of tours generated together with the objective value of the LP relaxation of the RMP, representing a lower bound on the number of tours necessary based

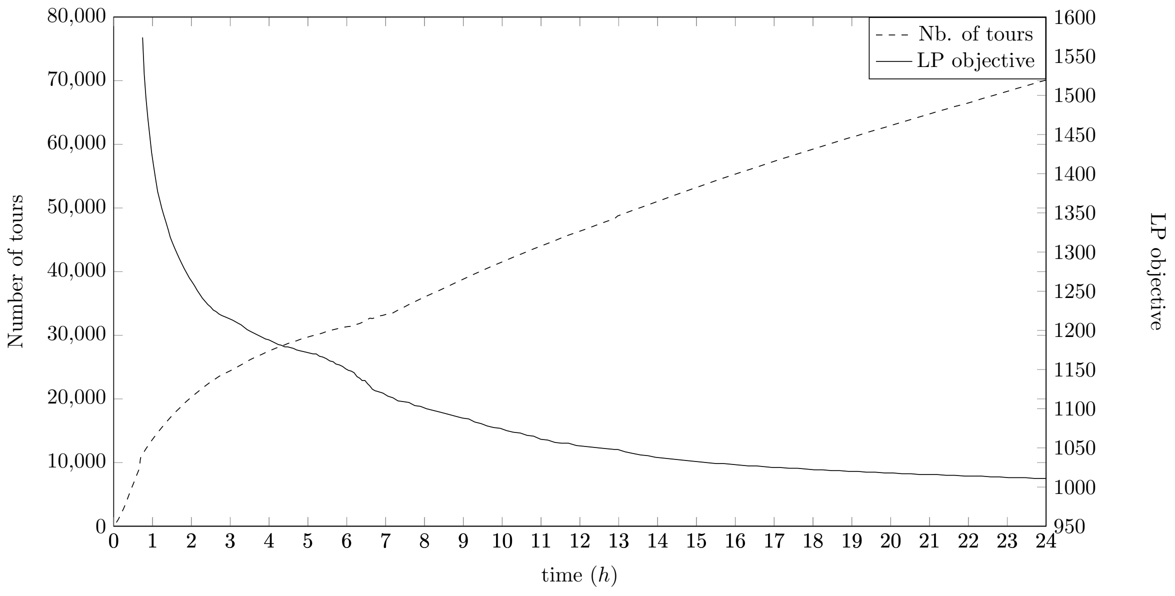

Figure 15: Nombre of tours and LP objective value during the CG on AGA

on the current columns. At each iteration of the price problem, at most 5 tours are generated for each of the 88 group graphs, leading to at most 440 new tours. The SMP is solved in 45 minutes. The LP objective is indicated when the generation of tours for the RMP starts. Throughout the procedure, the pricing problem takes between 3 and 4 minutes to be solved, whereas the solving time of the LP relaxation of the RMP strongly increases during the procedure. It is solved in a couple of seconds at the first iterations and up to 5 minutes after 24h (and 194 iterations). After 24h, the generation of tours is not over, tours with a negative reduced cost are still found, but the improvement of the LP objective at each iteration is under 0.01%. In other results in this section, we use only the tours generated in the first 24h (representing 76122 tours with an LP objective of 1009.31).

When letting the column generation run until no new tour is found, the procedure terminates after 74h and generates a total of 198461 tours with an LP objective of 942.6. In contrast, the column generation ends after only 42 minutes and 124 iterations on the original graphs, which are much smaller (instance FOA).

6.2.2 Set covering problem

We present in this section the results of the Branch & Bound of the RMP after the tours have been generated. Figure 16 illustrates the results on the original graphs (instance FOA) when using all tours generated by the column generation. The upper curve is the best solution found, the lower curve is the lower bound value. Solving the CSCP for the original graphs is much simpler than for other subgraphs, as the problem can be decomposed by depot as each CR appears in a single depot. The best solution found contains 1427 tours, and the lower bound is equal to 1368 tours (4.13% gap). A total of 1763245 nodes are explored in 4h in the Branch & Bound tree.

The number of tours found seems consistent with reality. In the current SBB planning, there are 1403 tours, and since we have missing groups, more tours are likely needed with our data. For comparability purposes, we compare the number of tours found for other instances with our methods to the 1427 tours

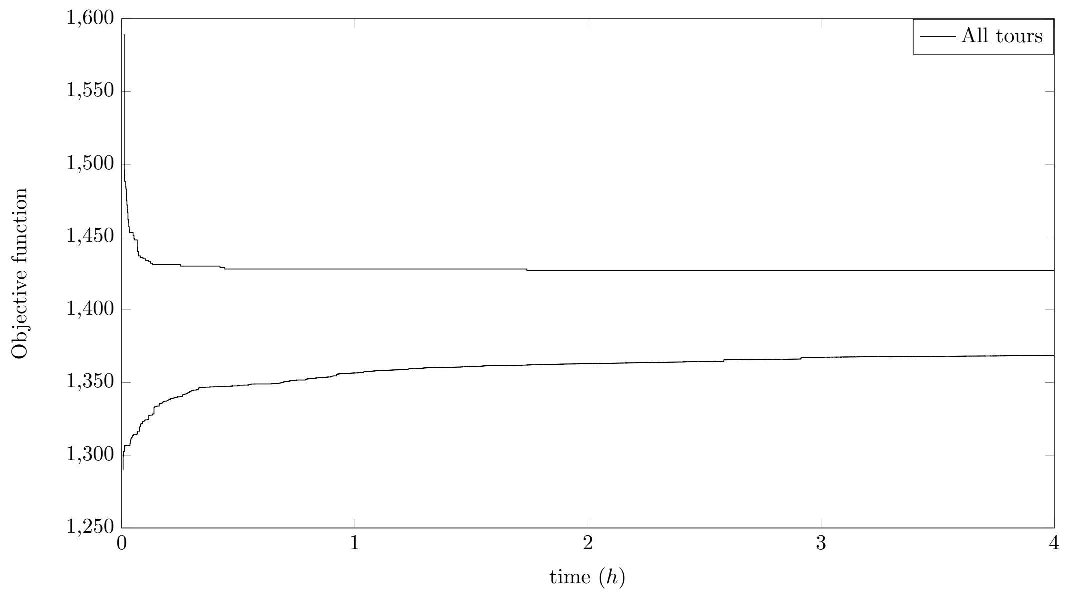

Figure 16: Feasible solution and lower bound evolution of the RMP on FOA

we found for the original assignment to evaluate the improvement.

On the AGA instance, the performance of the Branch & Bound strongly decreases. Although theoretically, the more tours are generated in the column generation, the better the integer solution should be, practically, the MILP solver cannot keep up with the size of the RMP as illustrated in Figure 17. The evolution of the best solution and the lower bound are plotted using tours generated in 6, 12, or 24 hours in the RMP. The best solution is obtained by considering only the tours generated in 6h. The solution quality degrades when considering more tours. We can also observe that the lower bound almost doesn't improve. Table 4 provides the best solution found, the gap to the lower bound, and the number of nodes explored in the Branch & Bound tree. We can see the number of nodes explored is significantly smaller

Tours	Best sol.	Gap	Nb. nodes
6h	1349	15.5%	1577
12h	1360	22.6%	30
24h	1374	27.7%	0

Table 4: Results of the RMP MILP

compared to instance FOA, reason for which the lower bound does not improve. The gaps obtained are also very high.

The tendency to have worse solution quality when considering more tours and the very high gap has been observed for all instances and was the main motivation for the IPB.

6.2.3 Iterative Price-and-Branch

We first present the results for the generic IPB illustrated in Figure 12, which does not consider column fixing. Figure 18 illustrates the execution of the IPB when using tours generated in 24h. This figure plots

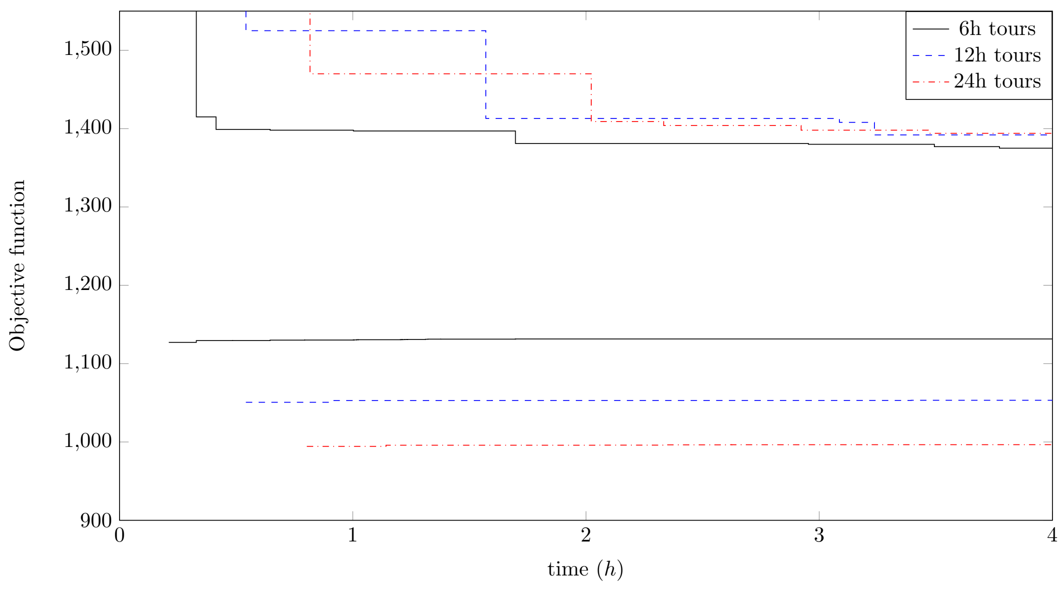

Figure 17: Feasible solution and lower bound evolution of the RMP on AGA

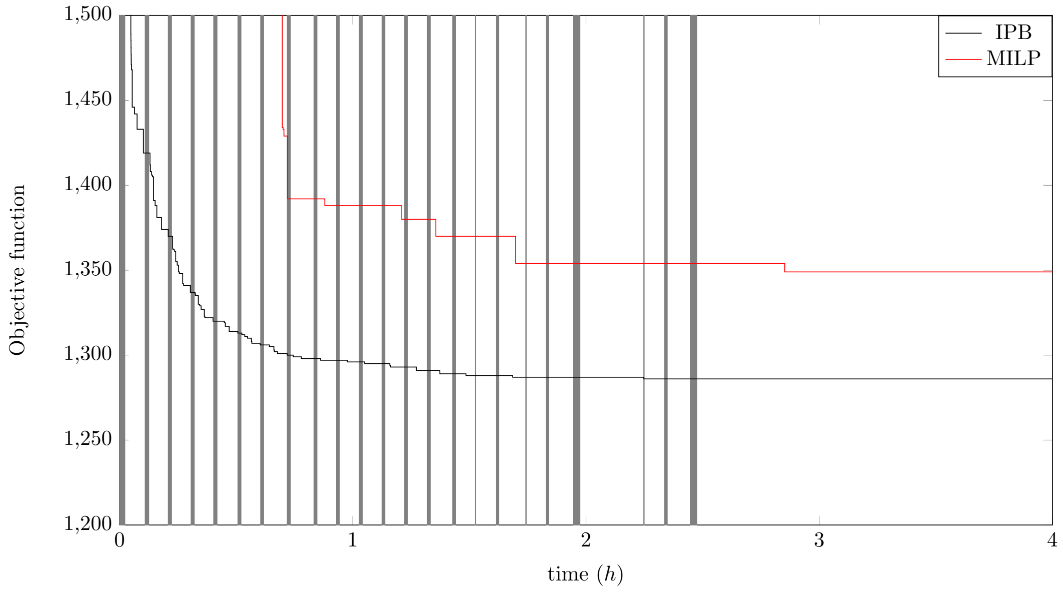

Figure 18: IPB starting with CSCP solution found after 10s in full MILP

the evolution of the best solution of the full MILP formulation and the IPB for comparison purposes. The IPB uses the first solution found by the full MILP (after a couple of seconds) as a start solution. The gray vertical lines are the periods during which the IPB adds new tours to the RMP in the pricing problem. In the other parts, the IPB is solving the integer version of the RMP. The solution quality improves faster with the IPB than with the full MILP formulation. Figure 19 is as Figure 18 except that the start solution is the one found after 4h with the full MILP (1349 tours). While the full MILP

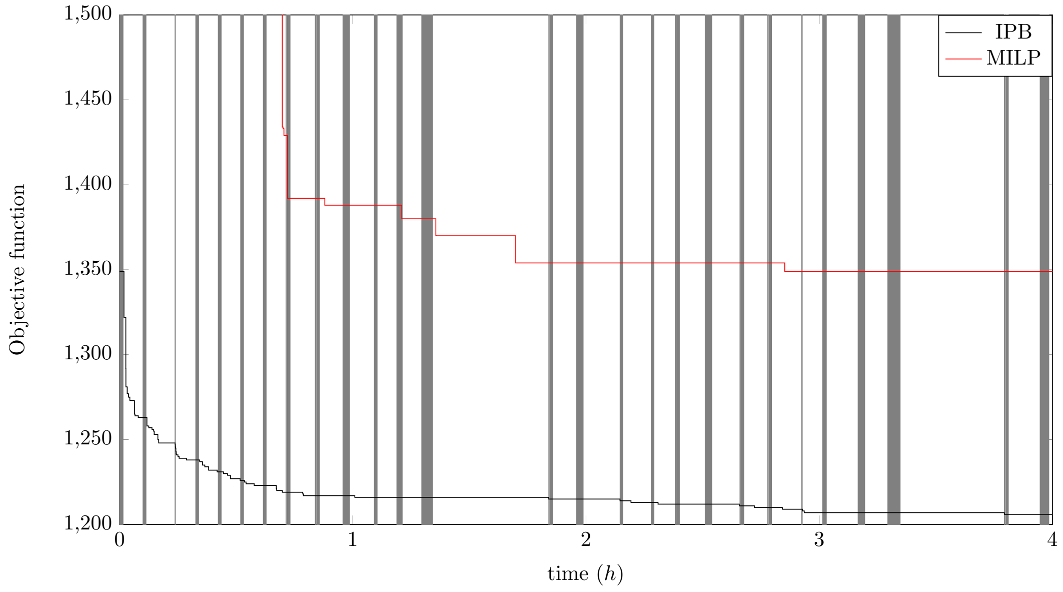

Figure 19: IPB starting with CSCP solution found after 4h in full MILP

could not improve the solution quality in the last hour of execution, the IPB immediately improved the solution. After a 4h runtime, the IPB finds a solution with 1222 tours (9.5% less than the start solution). No matter the start solution, the IPB immediately significantly improves it, as illustrated in Figure 20. Different start solutions are obtained from the full MILP after a runtime of 0h, 1h, 4h, and 8h. Overall, the better the start solution is, the better the solution of the IPB will be.

The IPB is tested in Figure 21 with tours generated after 6, 12, or 24 hours. The full MILP formulation tends to find worse solutions when more tours are considered, but we can see the IPB does not have this issue. The size of the MILP in the IPB is independent of the size of the tour pool. Only the solving time of the pricing problem increases when more tours are considered, but as the tours are precomputed, the pricing problem is solved quickly, as seen in Figure 18.

We now compare the results of the generic IPB to those considering column fixing and resetting illustrated in Figures 13 and 14. The results are illustrated in Figure 22. All three methods use the tours generated in 24h and the same start solution generated by the full MILP in 30s. Both variants fixing tours provide better solutions than the generic IPB. The IPB considering only tour fixing is the fastest as once a tour is fixed, it stays fixed until the end of the execution. This variant terminates after 1h19 when all tours in a solution are fixed. The variant considering column resetting is slower but finds a better final solution as the procedure offers a larger exploration of the solution space.

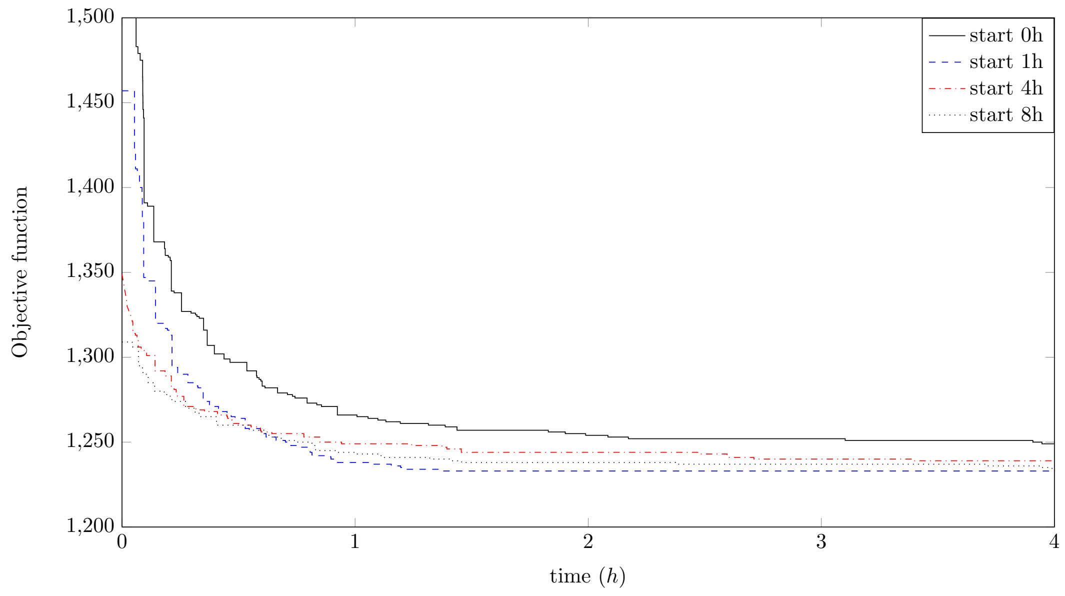

Figure 20: IPB with different start solutions

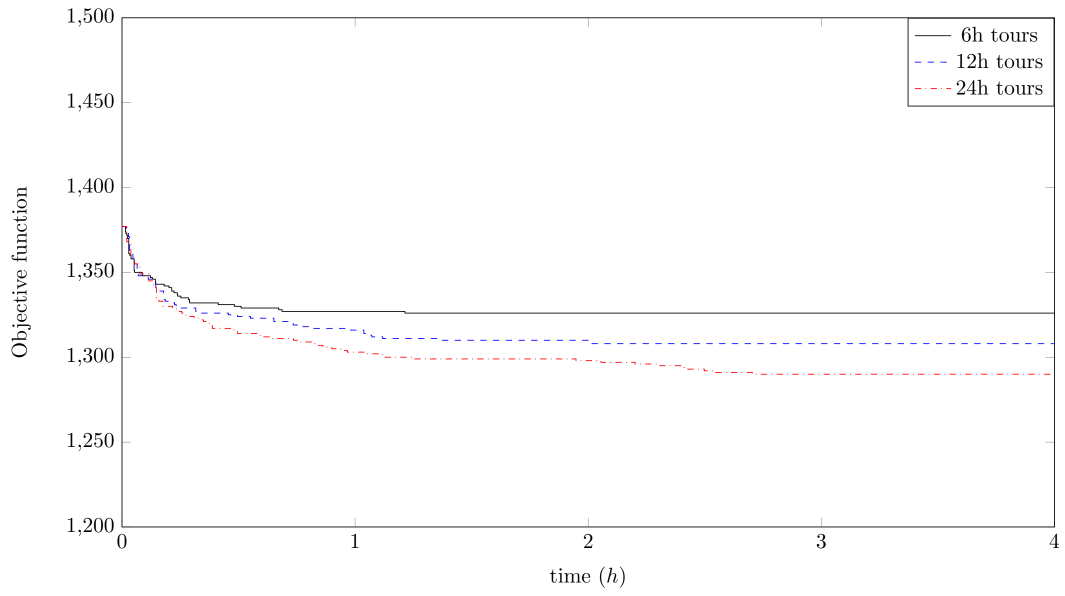

Figure 21: IPB with different number of tours

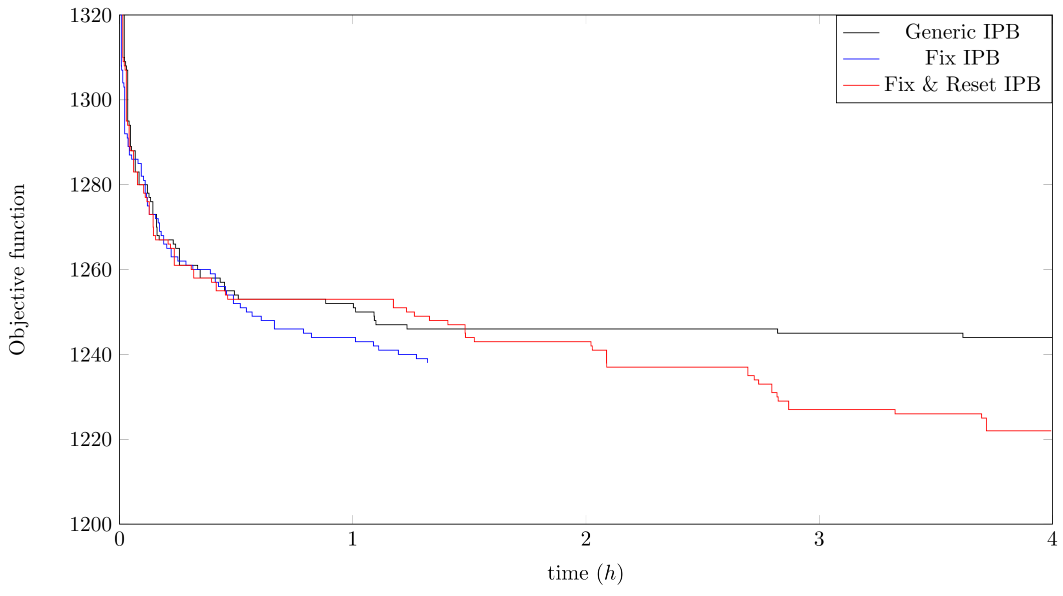

Figure 22: IPB variants comparison

6.3 Case studies

The results of the case studies are presented in this section. We report the best solutions found over all tests performed. These tests include all tours generated after the column generation terminates, solving the full MILP for up to 8h to find the best starting solution possible for the IPB and let the IPB run for 8h. The best solutions were always found with the IPB with tour fixing and resetting. The most valuable solutions are provided in addition to this report in JSON files.

6.3.1 Level of aggregation

Different levels of aggregation of CRs and groups were considered. We first tested the aggregation of CRs that are on the small rolling stock as described in Section 4.1.1. The efficiency of CRs was mainly tested on the first data set. The tests stopped the tour generation after 24h before solving the MILP for 4h. Results are summarized when all skills are required or only train skills in Table 5. The number of tours

	FDA	FDT	ADA	ADT
CG iterations	148	99	322	237
Nb. of tours generated	45009	44462	72209	54285
MILP solution	1139	1142	944	942

Table 5: Preliminary results

cannot be compared to those obtained on the second data set as the number of CRs was smaller. We can see that aggregating CRs produces significantly better results (FDA vs. ADA and FDT vs. ADT). The column generation is faster, and the solution obtained in the MILP is almost 20% better. This is mainly due to the size of the formulations, which are significantly smaller with the aggregations.

We can also observe that relaxing the routing skills (FDA vs. FDT and ADA vs. FDT) does provide any noticeable improvement in the solution quality. When discussing this result of the SBB, we conclude this is probably the consequence of the high correlation between train skills and routing skills.

The efficiency of using depot graphs was convincing in the first data set, in which most depots had a single group. Most of the tours generated could be covered by a group even though the groups were not explicitly considered. In the second data set, most depots had several groups. We observed that over 30% of the tours generated could not be assigned to any group as the skills required for some tours were jumping from one group to the other. Thus, we decided to generate tours on group graphs in which no correction would be needed. Although the number of subgraphs would go from 37 to 88 when considering group graphs, the impact on the solving time was limited. Considering group graphs does not significantly increase the size of the RMP, as there are limited constraints associated with subgraphs. The solving time of the pricing problems increases linearly in the number of subgraphs as the pricing problems are solved independently for each subgraph.

We thus focused the numerical experiments on group graphs with aggregated CRs and did not consider train skills further.

6.3.2 Minimization of tours

The minimization of tours was pushed to its maximum on AGA by letting the column generation run until it terminates (in 74h after generating 198461 tours), finding a start solution for the IPB with the full MILP formulation in 8h, and running the IPB with column fixing and resetting for 8h. The solution obtained contained 1168 tours, representing a reduction of 18.15% on the current SBB assignment containing 1427 tours. The best solution found is provided next to this report in file `AGA-sizeLimit.JSON`.

6.3.3 Maximum number of tours per group

We analyzed the group size on instance AGA to see which groups should be extended and which are too large. The results in this section are to interpret cautiously as we had missing groups in our data and had to increase the size of the groups we had by 50% to obtain a feasible solution. Figure 23 illustrates the number of tours per group.

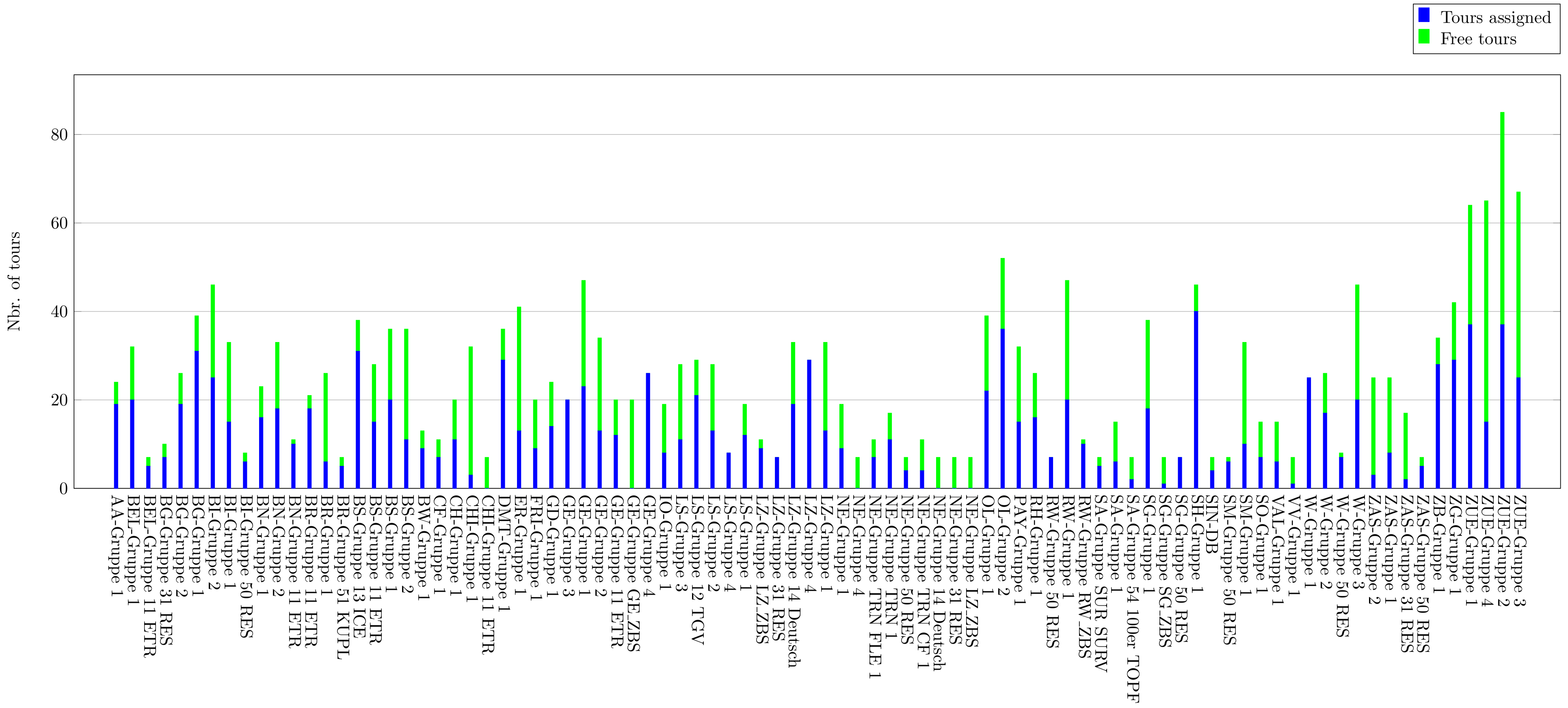

Figure 23: Number of tours on the best solution of AGA

This solution is the one minimizing the total number of tours detailed at the previous point. The bars in blue represent the number of tours assigned to the group, and the number of tours still available is represented in green. The following groups reach their maximum number of tours:

- GE - Gruppe 4
- LS - Gruppe 4
- LZ - Gruppe 31 RES
- LZ - Gruppe 4
- RW - Gruppe 50 RES
- SG - Gruppe 50 RES
- W - Gruppe 1

We can also observe some groups with no tours or a very limited number compared to their size, especially in the NE depot:

- CHI - Gruppe 1
- CHI - Gruppe 11 ETR
- GE - Gruppe GE_ZBS
- NE - Gruppe 4
- NE - Gruppe 14 Deutsch
- NE - Gruppe 31 RES
- NE - Gruppe NE_ZBS
- SG - Gruppe SG_ZBS
- VV - Gruppe 1

We performed the same tests without limiting the size of the groups. The solution obtained contained 1144 tours, providing a reduction of 24 tours. The best solution found is provided next to this report in file `AGA-noSizeLimit.JSON`. Figure 24 illustrates the number of tours per group.

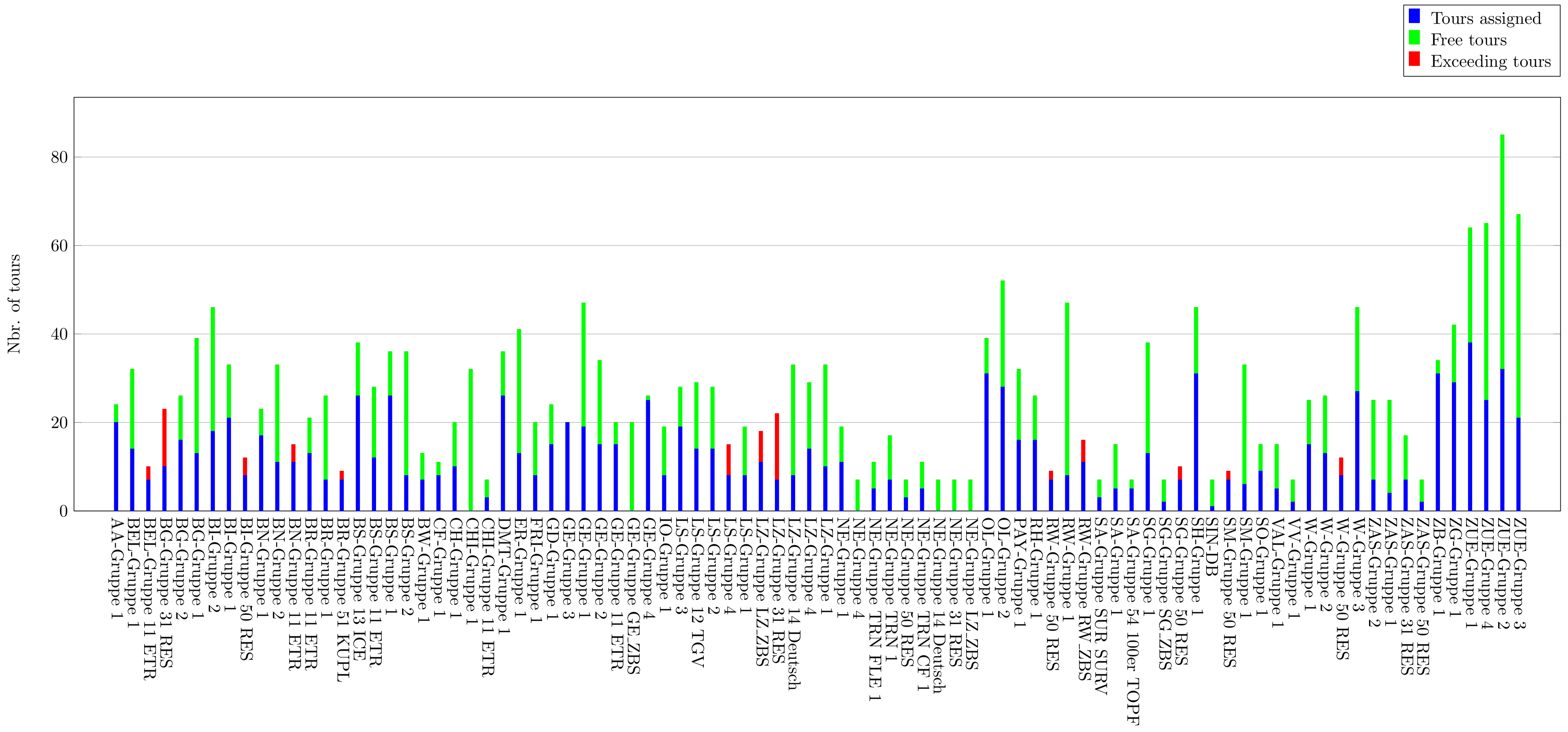

Figure 24: Number of tours on the best solution of AGA without limiting group sizes

The bars in red are the number of tours exceeding the group capacity. The following groups reach their maximum number of tours or exceed it:

- BEL-Gruppe 11 ETR
- BG - Gruppe 31 RES
- BI - Gruppe 50 RES
- BN - Gruppe 11 ETR
- BR - Gruppe 51 KUPL
- GE - Gruppe 3
- LS - Gruppe 4
- LZ - Gruppe LZ_ZBS
- LZ - Gruppe 31 RES
- RW - Gruppe 50 RES
- RW - Gruppe RW_ZBS
- SG - Gruppe 50 RES
- SM - Gruppe 50 RES
- W - Gruppe 50 RES

We can again observe some groups with no tours or a very limited number compared to their size, especially in the NE depot:

- CHI - Gruppe 1
- GE - Gruppe GE_ZBS
- NE - Gruppe 4
- NE - Gruppe 14 Deutsch
- NE - Gruppe 31 RES
- NE - Gruppe NE_ZBS
- SG - Gruppe SG_ZBS
- SIN - DB
- VV - Gruppe 1
- ZAS - Gruppe 50 RES

6.3.4 Potential new groups

The SBB suggested opening two new groups, one in BS and another in ZUE, both with all skills of all groups in BS and ZUE. This was motivated by the fact that some groups in both depots have the skills to go from one depot to another but do not have the skills to return during specific periods of the day. We considered a penalty of $\frac{4}{3}$ for tours in the new groups, meaning tours are affected to the new groups only if 4 tours of the old groups can be replaced by 3 tours in the new groups. The best solution found had an objective of 1139.66 and is available in file `AGA-sizeLimit-OptionalGroups`. Two tours are assigned to the new group in ZUE, and no tour to the new group in BS. In other solutions found that are worse, we always observed at least one tour in the new group in ZUE and no tours in the new group in BS.

As for the analysis of the maximum number of tours in the previous section, the results are to be interpreted cautiously because of the missing groups in our data.

7 Conclusion

The research direction of this project has strongly evolved from the initial demand of the SBB to assign CRs to depots without generating tours to the development of an algorithm generating tours for drivers. This choice was made as the quality of an assignment cannot be evaluated without tracking the resulting total paid time defined by the tours. The level of detail to consider has also strongly evolved from a low level of detail to a high level of detail. At first, tours were assigned to depots, and limited legal and technical constraints were considered. Still, throughout the project, we observed it was algorithmically feasible to integrate a higher level of detail to have more realistic solutions. The assignment of tours to groups does not complexify the problem, and all the legal constraints considered could easily be integrated into the tour generation.

At the beginning of the project, there was a doubt about whether it was possible to generate tours for the entire Swiss network, but this is feasible when considering numerical results, which reduce the number of tours by almost 20% compared to the current SBB planning. The Iterative Price-and-Branch approach is an efficient approach for the crew scheduling problem. This approach also allows considering several case studies by considering different objective functions or sizes for groups. Still, it could be interesting to narrow the information that is searched for specific case studies. For instance, regarding the size of groups, it could be interesting to fix a part of the assignment for several depots and only make the size of some specific groups vary. Another example is the model for skill development that could not be tested to identify which groups' skills could be developed and consider skills in all other groups as fixed.

Because of the integrity issues in the data delivered, the results of the case studies are to interpret cautiously. The Iterative Price-and-Branch methods produce a schedule with fewer tours than the current SBB methods. Decomposing the generation of tours in two stages (assigning CRs to depots before generating tours in each depot for their groups) is not the most efficient way. Regarding the case studies, further tests should be performed on accurate data to make strategic decisions to increase or reduce the size of groups or to develop skills in specific groups.

The main challenge for the SBB with our approach (or any algorithm that would be developed for crew scheduling) is the interfacing with the current SBB softwares. The input data needed for crew scheduling is currently available from Sopre with a very heavy format and integrity issues that seem to originate from bugs in Sopre, which the SBB and the developer of the Phönix program confirmed. The interpretation of the results of a crew scheduling algorithm also needs to be imported in Sopre at the moment for visualization. Again, the format required by Sopre to plot tours in its graphical interface is unclear. Some thinking on how to deal with this problem is very important if the SBB aims to integrate specific algorithms in their crew scheduling procedure. We see two options. Either the new algorithms are used with existing SBB softwares, in which case, the input and output for the algorithm must be clearly defined from the start of the project. Or, the SBB decides to implement a part of the software they are using to adapt its code based on the specific needs of the new algorithm. The second option is more costly in the short run but could be worth considering in the long run.

The project definition also took a lot of time during this project, mainly because of a need for more information. For instance, regarding long-term strategic decisions, there was no quantifiable information on the cost of increasing the size of a group or the cost of developing specific skills in groups. Furthermore, we did not know which elements are possible to change. The motivation of the SBB for this approach was to avoid influencing our research direction so we could propose novel ideas. Practically, this produced a lot of back-and-forth discussions during our regular meeting with the SBB. Our interpretation of the data or legal constraints to use for research directions between the meetings would be invalidated at a future meeting, delaying the development of algorithms. With a more precise definition of the research project and data to work on, we could have pushed the case studies further during this one-year research project and potentially provide reliable information for long-term strategic decisions.

The Iterative Price-and-Branch also has a strong potential for further development. The implementation of resource-constrained shortest path problem generating tours is quite simple, and it is possible to improve its computation time based on the literature. There is also the possibility to consider additional legal constraints as resources in the tour generation to improve the level of detail. Furthermore, the Iterative Price-and-Branch was developed in a limited time at the end of the project. It is probably possible to improve this method by testing other exiting criteria when solving the integer RMP or other fixing and resetting criteria for tour fixing. Also, we considered a set covering modelization for the problem, but all our methods can be adapted for a partitioning modelization. Suppose a new crew scheduling algorithms that would provide an input for Phönix was to be developed. In that case, a partitioning approach could be more adapted as Phönix works with a partitioning approach.