Lancaster University

# Conference Scheduler User Guides

**Version 0.1**

## Authors

**Ahmed Kheiri** — a.kheiri@lancaster.ac.uk

**Peter Jacko** — p.jacko@lancaster.ac.uk

**Yaroslav Pylyavskyy** — y.pylyavskyy1@lancaster.ac.uk

# Contents

# 1. Project Description

## 1.1 Overview

The Conference Scheduler is an advanced tool designed to optimise the process of scheduling conferences in an autonomous, effortless and fully automated manner. This tool uses Excel, which follows a specific template, to store input data and Python for the implementation of optimisation algorithms, ensuring that conference schedules are created efficiently and effectively. The primary goal is to provide a complete solution to automated conference scheduling that is easily customised to fit the needs of different conferences.

## Key Features

1. **User-Friendly Data Input:** Users can easily input and manage their data using Excel. An Excel template along with data examples are provided to standardise the data entry process, minimise errors and offer flexibility.
2. **Automated Scheduling:** Automatically assigns tracks to sessions and rooms, and submissions to sessions, time slots and rooms based on the input data and optimisation criteria. It also detects and resolves potential conflicts, ensuring a smooth and conflict-free conference schedule.
3. **Constraints Management:** Contains a pool of constraints to select from and allows weight assignment for each constraint based on user's preferences.

4. **Advanced Optimisation Techniques:** Advanced optimisation algorithms are included in the scheduler to ensure a quick schedule generation, even for large-scale conferences with thousands submissions.

5. **Hybrid & Online Conferences:** Suitable for hybrid and online conferences where submissions need to be scheduled in appropriate sessions considering time-zone information.

6. **Output and Reporting:** Generates comprehensive optimised schedules in Excel format that can be easily reviewed and shared with stakeholders. The user is not only able to view a detailed report of violations for each constraint but also can manually edit the solution and observe the impact of their changes on solution quality.

## Benefits

- **Time-Saving:** Automates the arduous and time-consuming manual process of conference scheduling, saving significant time and effort for conference organisers.
- **Optimisation:** Uses advanced optimisation techniques to deliver high-quality conference schedules considering numerous preferences and constraints.
- **Flexibility:** Adaptable to different types of conferences and can handle a wide range of scheduling requirements.
- **Accuracy:** Reduces the likelihood of human error through automated checks and optimisations.
- **Scalability:** Suitable for managing both small and large conferences.

## How It Works

1. **Excel file configuration:** Users enter their data into the provided Excel template, set scheduling requirements and specify preferences (see Section 2).

2. **Optimisation Process:** The Excel data is imported into the system and the user selects their preferred algorithm for scheduling optimisation (see Section 3).

3. **Review and Adjustments:** The system generates the optimised schedule, which is then exported back into Excel. Users can then easily review the generated schedule, make any necessary adjustments and observe the impact on the schedule quality, and finalise the conference plan (see subsection 3.4.2).

## 1.2 Dependencies

- NumPy $\geq$ 1.24.2
- pandas $\geq$ 2.2.2
- PuLP $\geq$ 2.8.0
- Python $\geq$ 3.8.16
- pytz $\geq$ 2022.2

## 1.3 Terminology

- **Submission:** A formal event that requires scheduling at a conference (e.g., paper, presentation, tutorial, workshop, etc.).
- **Track:** A group of submissions with similar subject (e.g., stream, subject area, topic, etc.).
- **Time slot:** A fixed predefined amount of time available for presentation (e.g., 15 minutes, 20 minutes, etc.).
- **Session:** A certain time period of the conference that consists of a number of time slots (e.g., the duration of a session consisting of 4 time slots is 1 hour, assuming each time slot is 15 minutes).

## 1.4 Constraints Available

Users are able to select the constraints to include during the schedule optimisation by assigning a weight value to each constraint (see section 2.9). The following constraints are available to select from:

- **Presenters' conflicts:** In many conferences, authors are allowed to present more than one submission. This is resolved by either scheduling such submissions within the same room of a session or within different sessions (or time slots). Users can select between session or time slot level conflict resolution. (see section 2.1)
- **Presenters' preferences:** These are requests received from presenters in which they either express a preferred session to present their submission or declare their unavailability to present at specific sessions. (see section 2.1)
- **Presenters' time zones:** On the occasion of an online or hybrid conference, presenters may request the consideration of time zone differences upon scheduling. (see section 2.1 & section 2.3)
- **Rooms preferences:** Sometimes presenters may request to present their submission at a specific room for various reasons. Some examples are that a room may provide specific facilities which others do not provide, and some rooms may be easier to access in comparison to others. (see section 2.1)
- **Attendees' conflicts:** Some conferences collect preferences from attendees regarding which submissions they would prefer to attend. In such cases, an attendee conflict occurs when two preferred submissions of an attendee are scheduled in parallel. This is resolved in the same way as presenters' conflicts and users can select between session or time slot level conflict resolution. (see section 2.1)
- **Rooms capacities:** Users can express preferences regarding the scheduling of tracks into rooms by setting appropriate penalty values. (see section 2.6)
- **Similar tracks:** Sometimes, conferences have a number of tracks which are similar with the potential of attracting the interest of the same audience. Users can define which tracks are similar to avoid having them scheduled in parallel by setting appropriate penalty values. (see section 2.7)
- **Parallel tracks:** This constraint avoids scheduling the same track in parallel.
- **Session hopping:** Having a track scheduled in multiple rooms is inconvenient for the participants as they would have to switch rooms frequently. This constraint minimises the number of rooms that each track utilises.
- **Track chairs' conflicts:** Tracks are usually chaired by a person who might be also a presenter and/or an attendee at a conference. A track chair conflict occurs when either a track chair is responsible for two tracks which are scheduled in parallel

or a track chair is also a presenter or an attendee of a submission belonging to another track which is scheduled in parallel. (see section 2.2)

- **Tracks' scheduling preferences:** Users can express preferences regarding the scheduling of tracks into sessions by setting appropriate penalty values. (see section 2.5)
- **Rooms unavailability:** Sometimes, certain rooms might be unavailable for utilisation during certain sessions. Users can define which rooms are unavailable during certain sessions by setting appropriate penalty values. (see section 2.8)
- **Consecutive tracks:** This constraint aims to schedule tracks in a consecutive manner.
- **Submission's Order:** Users can define the scheduling order of submissions within their tracks.

## 1.5 Optimisation Methods Available

Users are able to select which optimisation method they prefer to optimise the conference schedule (see Section 3). Each optimisation method has its benefits and limitations which are summarised in Table 1.1 The following optimisation algorithms are available:

1. **Integer Programming:** Two mathematical models are available, an exact model including basic constraints and an extended model including additional constraints. For more information see `https://doi.org/10.1016/j.ejor.2024.04.001`.
2. **Matheuristic:** A decomposed robust matheuristic solution approach that consists of two phases. In phase one, an integer programming model is used to build the high-level schedule by assigning tracks into sessions and rooms. Based on this solution, the low-level schedule is created where submissions are allocated into sessions, rooms, and time slots. In phase two, a selection perturbative hyper-heuristic is used to further optimise both levels of the schedule.
3. **Hyper-heuristic:** A selection perturbative hyper-heuristic consisting of four low-level heuristics, specifically two swap heuristics, a reverse heuristic, and a ruin and recreate heuristic. Its framework involves a two-step iterative process during scheduling optimisation where, in the first step, a low-level heuristic is selected

randomly and is applied to the schedule. Then, in the second step, if the modified schedule is not worse than the previous, it is accepted. Otherwise, it is rejected and the previous schedule is restored.

TABLE 1.1: Benefits and Limitations of each Optimisation Method

| Method | Benefits | Drawbacks |
|---|---|---|
| Integer Programming | Optimal solutions.<br>Best for small to medium conferences with few constraints.<br>Best for instances where hard constraints can be satisfied. | May fail to return solution.<br>Unsuitable for time slot level constraints.<br>Unsuitable for large scale instances.<br>Commercial software licence required. |
| Matheuristic | Fast and Decent solutions.<br>Always finds solutions.<br>Handles numerous constraints.<br>Suitable for both session and time slot level constraints.<br>Suitable for conferences of any size including large scale instances. | Sub-optimal solutions.<br>Commercial software licence required. |
| Hyper-heuristic | Decent solutions.<br>Always finds solutions.<br>Does not require commercial software licence.<br>Handles numerous constraints.<br>Suitable for both session and time slot level constraints.<br>Suitable for conferences of any size including large scale instances. | Sub-optimal solutions.<br>Slower than Matheuristic. |

## 1.6 Citation

If you use the Conference Scheduler in your research or conference planning, please cite the relevant publication as follows:

*Ahmed Kheiri , Yaroslav Pylyavskyy, and Peter Jacko (2024) CSPLib – A Benchmark Library for Conference Scheduling Problems.*

*Yaroslav Pylyavskyy, Ahmed Kheiri, and Peter Jacko (2024) A Two-phase Matheuristic Approach to Conference Scheduling Problems.*

*Yaroslav Pylyavskyy, Peter Jacko, and Ahmed Kheiri. A Generic Approach to Conference Scheduling with Integer Programming. European Journal of Operational Research, 317(2):487-499, 2024. ISSN 0377-2217. doi:* `https://doi.org/10.1016/j.ejor.2024.04.001`.

## 1.7 Licensing

The Conference Scheduler code is open-source and is distributed under the MIT License. By using the Conference Scheduler, you agree to comply with the terms and conditions of the MIT License.

Please note that in order to use the GUROBI solver, a license is required.

## 1.8 Acknowledgements

The development of the Conference Scheduler has been supported by the UK Research and Innovation through the Programme Grant EP/V520214/1.

# 2. Data Format

The Excel file containing the input data needs to follow the specific format as described in the following sections. Many examples are available in the *Dataset* folder on github (see `https://github.com/ahmedkheiri/CSPLib/tree/main/Dataset`). The Excel file contains the necessary inputs for the Conference Scheduler and allows the user to make configurations. It consists of the following sheets: submissions, tracks, sessions, rooms, tracks_sessions penalty, tracks_rooms penalty, similar tracks, and sessions_rooms penalty, and parameters.

Note that all string type inputs are **case sensitive** and **must exactly match each other across all over the sheets**, otherwise an error is raised. **Users are strongly suggested to avoid using special characters such as -,*,etc. as this may cause errors.**

## 2.1 Submissions

The submissions sheet contains information and constraints for each submission. It consists of the following fields:

- **Reference:** Unique name or ID of submission. Each value is **case sensitive**, it must be **unique** and of **string** type. For example, NEW19A19, submission1, PaperOne, etc.
- **Track:** Name of track to which the submission belongs to. It refers to a group which contains similar submissions. Each value is **case sensitive** and must be a **string**. For instance, Analytics, Optimisation, Big Data and AI, etc.

- **Required Timeslots:** The number of time slots required for the submission. Each value must be an **integer**.
- **Order (optional):** The order in which submission should be scheduled within its respective track. Each value must be an **integer**. If irrelevant, use 0.
- **Time Zone:** The time zone of the main presenter's location. The range of GMT is between -12 to +12 (inclusive) and is used to determine the time zone. Half hour differences are not supported. For example, if a time zone is GMT+5:30, then a GMT+6 could be used instead. Each value is **case sensitive** and must be in the following **string** format: GMT+/-#. For instance, GMT+0, GMT+2, GMT-4, etc. If irrelevant, fill in the time zone of conference's location.
- **Presenters (optional):** The author or authors of the respective submission. Multiple authors can be used. Each value is **case sensitive** and must be a **string**. Note that if multiple authors are used, each author must be separated by a **comma** followed by <**space**>. For example, Author1, Author2, Author3. If irrelevant, leave empty.
- **Attendees (optional):** The attendee or attendees of the respective submission. Multiple attendees can be used. Each value is **case sensitive** and must be a **string**. Note that if multiple attendees are used, each attendee must be separated by a **comma** followed by <**space**>. For example, Attendee1, Attendee2, Attendee3. If irrelevant, leave empty.

In addition to these fields, separate columns must be used for each session followed by columns for each room as shown in Figure 2.1. The next number of columns is determined by the total number of available sessions, where each column corresponds to a session (from column H to column K in this example). Under these columns a penalty value may be set accordingly so as not to schedule the corresponding submission into the corresponding session. For instance, Submission_7 must be ideally scheduled in Session_1 or in Session_2 so we keep these values empty. Additionally, we do not want to schedule Submission_7 in Session_3 or Session_4, but if that cannot be fully satisfied then we prefer Session_3. To do so, we set a penalty value of 1 for Session_3 and a penalty value of 10 for Session_4.

Then, the number of the remaining columns is determined by the total number of available rooms, where each column corresponds to a room (from column L to column O in this example). Within these columns a penalty value may be set accordingly so as not

to schedule the corresponding submission into the corresponding room. For example, if we want Submission_9 scheduled in Room_2, we penalise all rooms except for Room_2.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Reference | Track | Required Timeslots | Order | Time Zone | Presenters | Attendees | Session_1 | Session_2 | Session_3 | Session_4 | Room_1 | Room_2 | Room_3 | Room_4 |
| 2 | Submission_1 | Track_1 | 1 | 0 | GMT+0 | Name_Surname_1 | Name_Surname_35 | | | | | | | | |
| 3 | Submission_2 | Track_1 | 2 | 0 | GMT+1 | Name_Surname_2 | Name_Surname_36 | | | | | | | | |
| 4 | Submission_3 | Track_1 | 3 | 0 | GMT+2 | Name_Surname_3 | Name_Surname_37 | | | | | | | | |
| 5 | Submission_4 | Track_1 | 4 | 0 | GMT+3 | Name_Surname_4 | Name_Surname_38 | | | | | | | | |
| 6 | Submission_5 | Track_1 | 1 | 0 | GMT+4 | Name_Surname_5 | Name_Surname_39 | | | | | | | | |
| 7 | Submission_6 | Track_2 | 1 | 0 | GMT+5 | Name_Surname_6 | Name_Surname_40 | | | | | | | | |
| 8 | Submission_7 | Track_2 | 1 | 0 | GMT+6 | Name_Surname_7 | Name_Surname_41 | | | 1 | 10 | | | | |
| 9 | Submission_8 | Track_2 | 1 | 0 | GMT+7 | Name_Surname_8 | Name_Surname_42 | | | | | | | | |
| 10 | Submission_9 | Track_2 | 1 | 0 | GMT+8 | Name_Surname_9 | Name_Surname_43 | | | | | 10 | | 10 | 10 |
| 11 | Submission_10 | Track_2 | 1 | 0 | GMT+9 | Name_Surname_10 | Name_Surname_44 | | | | | | | | |
| 12 | Submission_11 | Track_3 | 1 | 0 | GMT+10 | Name_Surname_11 | Name_Surname_45 | 1 | 10 | | | | | | |
| 13 | Submission_12 | Track_3 | 1 | 0 | GMT+11 | Name_Surname_12 | Name_Surname_46 | 1 | 10 | | | | | | |
| 14 | Submission_13 | Track_3 | 1 | 0 | GMT+12 | Name_Surname_13 | Name_Surname_47 | | | | | | | | |
| 15 | Submission_14 | Track_4 | 1 | 0 | GMT-12 | Name_Surname_14 | Name_Surname_48 | | | | | | | | |
| 16 | Submission_15 | Track_4 | 1 | 0 | GMT-11 | Name_Surname_15 | Name_Surname_49 | | | | | | | | |
| 17 | Submission_16 | Track_4 | 1 | 0 | GMT-10 | Name_Surname_16 | Name_Surname_50 | | | | | | | | |
| 18 | Submission_17 | Track_4 | 1 | 0 | GMT-9 | Name_Surname_17 | Name_Surname_51 | | | | | | | | |
| 19 | Submission_18 | Track_5 | 2 | 1 | GMT-8 | Name_Surname_18 | Name_Surname_52 | | | 1 | 10 | | | | |
| 20 | Submission_19 | Track_5 | 1 | 0 | GMT-7 | Name_Surname_19 | Name_Surname_53 | | | | | | | | |
| 21 | Submission_20 | Track_5 | 1 | 0 | GMT-6 | Name_Surname_20 | Name_Surname_54 | | | | | | | | |
| 22 | Submission_21 | Track_5 | 1 | 0 | GMT-5 | Name_Surname_21 | Name_Surname_55 | | | | | | | | |
| 23 | Submission_22 | Track_5 | 1 | 0 | GMT-4 | Name_Surname_22 | Name_Surname_56 | | | | | | | | |
| 24 | Submission_23 | Track_6 | 1 | 0 | GMT-3 | Name_Surname_23 | Name_Surname_57 | | | | | | | | |
| 25 | Submission_24 | Track_6 | 1 | 0 | GMT-2 | Name_Surname_24 | Name_Surname_58 | | | | | | | | |

FIGURE 2.1: Submissions sheet

## 2.2 Tracks

The tracks sheet contains information for each track and consists of the following two fields:

- **Tracks:** Unique name or ID of track which contains submissions of similar subject. Each value is **case sensitive**, it must be **unique** and of **string** type. For instance, Analytics, Optimisation, Big Data and AI, etc.
- **Chairs (optional):** The chair or chairs of the respective track. Multiple chairs can be used. Each value is **case sensitive** and must be a **string**. Note that if multiple chairs are used, each chair must be separated by a **comma** followed by <**space**>. For example, Chair1, Chair2, Chair3. If irrelevant, leave empty.

## 2.3 Sessions

The Sessions sheet contains all the necessary information regarding sessions and consists of the following fields:

- **Sessions:** Unique name or ID of session. Each value is **case sensitive**, it must be **unique** and of **string** type. For example, Wed1, MonMorning, Thursday2, etc.

- **Max Number of Timeslots:** The maximum number of time slots in the respective session. Each value must be an **integer**.
- **Date:** The date that each session corresponds to. Each value must follow the following format **MM/DD/YYYY**.
- **Start Time:** The time at which the session begins. Each value must be in the following time format **HH:MM**. For instance, 12:00, 16:30, etc.
- **End Time:** The time at which the session ends. Each value must be in the following time format **HH:MM**. For example, 12:00, 16:30, etc.

## 2.4 Rooms

The Rooms sheet contains the names of the rooms and consists of the following field:

- **Rooms:** Unique name or ID of room. Each value is **case sensitive**, it must be **unique** and of **string** type. For example, Room 1, RoomA, etc.

## 2.5 Tracks-Sessions Penalty

The Tracks-Sessions Penalty sheet is used to define penalty values to avoid scheduling a specified track into a specified session as presented in Figure 2.2. Column A includes all tracks, and the number of next columns is given by the total number of sessions available, where each column corresponds to a session (from column B to column E in this example). Different penalty values can be used to express preferences. Note that penalty values must be **integers**. For instance, Track_5 must be ideally scheduled in Session_3 and/or in Session_4 so we keep these values empty. Additionally, we do not want to schedule Track_5 in Session_1 or Session_2, but if that cannot be fully satisfied then we prefer Session_2. To do so, we just set a small penalty value for Session_2 and a high penalty value for Session_1.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | Session_1 | Session_2 | Session_3 | Session_4 |
| 2 | Track_1 | | | | |
| 3 | Track_2 | | | | |
| 4 | Track_3 | | | | |
| 5 | Track_4 | | | | |
| 6 | Track_5 | 10 | 1 | | |
| 7 | Track_6 | | | | |
| 8 | Track_7 | | | | |
| 9 | Track_8 | | | | |

FIGURE 2.2: Tracks-Sessions Penalty sheet

## 2.6 Tracks-Rooms Penalty

The Tracks-Rooms Penalty sheet is used to control the scheduling process of tracks into rooms as displayed in Figure 2.3. Column A contains all tracks, and the number of next columns is given by the total number of rooms available, where each column corresponds to a room (from column B to column E in this example). Different penalty values can be used to express preferences. Note that penalty values must be **integers**. For instance, if we want Track_1 and Track_2 scheduled in Room_4, then we set a penalty value for all rooms except for Room_4.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | Room_1 | Room_2 | Room_3 | Room_4 |
| 2 | Track_1 | 10 | 10 | 10 | |
| 3 | Track_2 | 10 | 10 | 10 | |
| 4 | Track_3 | | | | |
| 5 | Track_4 | | | | |
| 6 | Track_5 | | | | |
| 7 | Track_6 | | | | |
| 8 | Track_7 | | | | |
| 9 | Track_8 | | | | |

FIGURE 2.3: Tracks-Rooms Penalty sheet

## 2.7 Similar Tracks

The Similar Tracks sheet allows to define which pair of tracks should not be scheduled in parallel as shown in Figure 2.4. Column A includes all tracks, and the number of next columns is given by the total number of tracks, where each column corresponds to a track (from column B to column I in this example). Different penalty values can be used to express preferences. Note that penalty values must be **integers**. Suppose Track_3 is similar to Track_6 and Track_8 and we do not want to schedule Track_3 and Track_6 or Track_3 and Track_8 in parallel. We define this by simply setting a penalty value for that pairs of tracks.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | Track_1 | Track_2 | Track_3 | Track_4 | Track_5 | Track_6 | Track_7 | Track_8 |
| 2 | Track_1 | | | | | | | | |
| 3 | Track_2 | | | | | | | | |
| 4 | Track_3 | | | | | | 1 | | 1 |
| 5 | Track_4 | | | | | | | | |
| 6 | Track_5 | | | | | | | | |
| 7 | Track_6 | | | | | | | | |
| 8 | Track_7 | | | | | | | | |
| 9 | Track_8 | | | | | | | | |

FIGURE 2.4: Similar Tracks sheet

## 2.8 Sessions-Rooms Penalty

The Sessions-Rooms Penalty sheet is used to define unavailability of rooms for certain sessions as presented in Figure 2.5. Column A contains all sessions, and the number of next columns is given by the total number of available rooms, where each column corresponds to a room (from column B to column E in this example). Different penalty values can be used to express preferences. Note that penalty values must be **integers**. For instance, if Room_3 is unavailable during Session_4, then we add a penalty value for that session-room pair.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | Room_1 | Room_2 | Room_3 | Room_4 |
| 2 | Session_1 | | | | |
| 3 | Session_2 | | | | |
| 4 | Session_3 | | | | |
| 5 | Session_4 | | | 10 | |

FIGURE 2.5: Sessions-Rooms Penalty sheet

## 2.9 Parameters

The Parameters sheet includes settings for hybrid or online conferences, and allows to set weight values for penalties as shown in Figure 2.6. Columns A and B are associated with settings regarding hybrid or online conferences. The local timezone field refers to the timezone that applies at the location of the conference. Next, suitable scheduling times fields indicate the ideal scheduling time window for which submissions are not penalised. Less suitable scheduling times fields create a new time window for which submissions are slightly penalised, while unsuitable scheduling times are heavily penalised submissions. All times are converted into local times of online presenters. For instance, a submission would be penalised by 1 if the converted local time of the presenter is between 7:00 and 9:30 or between 21:30 and 23:00. If the converted local time is between 23:00 and 7:00 then a penalty of 10 will apply, otherwise if the converted local time is between 9:30 and 21:30 then no penalty applies. These settings along with defined session's start and end time are used to identify suitable sessions that are convenient for online presenters. Lastly, columns D and E are used to set the weight values. Setting different weight values allows the prioritisation of the listed types of penalties. Note that the time zone field is **case sensitive** and must be in the following **string** format: GMT+/-# (e.g., GMT+2, GMT-4, etc.), the time field must be in the following time format **HH:MM** (e.g., 10:30, 19:15, etc.), and penalty along with weight fields must be **integers**.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Sessions** | | | **Weights** | |
| 2 | Local time zone: | GMT+0 | | Tracks_Sessions\|Penalty: | 10 |
| 3 | **Suitable scheduling times** | | | Tracks_Rooms\|Penalty: | 10 |
| 4 | From: | 9:30 | | Sessions_Rooms\|Penalty: | 20 |
| 5 | To: | 21:30 | | Similar Tracks: | 10 |
| 6 | **Less suitable scheduling times** | | | Number of Rooms per Track: | 0 |
| 7 | From: | 7:00 | | Parallel Tracks: | 0 |
| 8 | To: | 23:00 | | Consecutive Tracks: | 10 |
| 9 | Penalty: | 1 | | Submissions_Timezones: | 5 |
| 10 | **Unsuitable scheduling times** | | | Submissions Order: | 0 |
| 11 | Penalty: | 10 | | Submissions_Sessions\|Penalty: | 5 |
| 12 | | | | Submissions_Rooms\|Penalty: | 5 |
| 13 | | | | Presenters Conflicts: | 0 |
| 14 | | | | Attendees Conflicts: | 0 |
| 15 | | | | Chairs Conflicts: | 0 |
| 16 | | | | Presenters Conflicts Timeslot Level: | 0 |
| 17 | | | | Attendees Conflicts Timeslot Level: | 0 |

FIGURE 2.6: Parameters sheet

# 3. Use Cases

The Conference Scheduler consists of the following modules: Main, Optimisation, Parameters, Problem, Room, Session, Solution, Solver_Checker_v1.1, Submission, and Track. To run the solver, the user should open and run the Main.py file. Some examples use cases are presented in the next sections.

## 3.1 Integer Programming

Two integer programming models are available: an exact model and an extended model. For best results, we recommend GUROBI solver which requires a licence. Alternatively, "GLPK_CMD" solver can be used which is free (see subsection 3.4.1). Note that both models only handle constraints on a session level and if any model is infeasible then either some constraints need to be relaxed or another method should be selected to schedule the conference.

The exact model handles the following constraints: presenters' conflicts, presenters' preferences, presenters' time zones, rooms preferences, rooms capacities, parallel tracks, session hopping, tracks' scheduling preferences, and rooms unavailability.

The extended model includes all the constraints of the exact model and the following additional constraints: attendees' conflicts, similar tracks, track chairs' conflicts, and consecutive tracks.

### 3.1.1 Schedule N2OR conference using the exact model and print solution's information

```
from Optimisation import *
instance = "N2OR"
f_name = "..\\Dataset\\"+str(instance)+".xlsx"
p = Problem(file_name = f_name)
parameters = p.ReadProblemInstance()
p.FindConflicts()
p.AssignTimezonesPenalties(parameters)
sol = Solution(p)
solver = ExactModel(p, sol)
solver.solve(timelimit = 3600)
print("Objective Value:", sol.EvaluateSolution())
print("All submissions scheduled?", sol.EvaluateAllSubmissionsScheduled())
sol.printViolations()
```

### 3.1.2 Schedule GECCO21 conference (online) using the extended model and save solution in Excel file

```
from Optimisation import *
instance = "GECCO21"
f_name = "..\\Dataset\\"+str(instance)+".xlsx"
p = Problem(file_name = f_name)
parameters = p.ReadProblemInstance()
p.FindConflicts()
p.AssignTimezonesPenalties(parameters)
sol = Solution(p)
solver = ExtendedModel(p, sol)
solver.solve(timelimit = 3600)
sol.toExcel(file_name = "Solution"+str(instance)+".xlsx")
```

## 3.2 Matheuristic

The matheuristic algorithm consists of two phases and handles all available constraints including time slot level. In phase one, an integer programming model is used to build the

high-level schedule by assigning tracks into sessions and rooms (either requires GUROBI license or see how to configure free solver in subsection 3.4.1). Based on this solution, the low-level schedule is created where submissions are allocated into sessions, rooms, and time slots. In phase two, a selection perturbative hyper-heuristic is used to further optimise both levels of the schedule.

### 3.2.1 Schedule ISF22 conference using the matheuristic with a 300 seconds time limit overall and save solution in Excel file

```
from Optimisation import *
instance = "ISF22"
f_name = "..\\Dataset\\"+str(instance)+".xlsx"
p = Problem(file_name = f_name)
parameters = p.ReadProblemInstance()
p.FindConflicts()
p.AssignTimezonesPenalties(parameters)
sol = Solution(p)
solver = Matheuristic(p, sol)
s_time = time()
solver.solve(s_time, run_time = 300)
sol.toExcel(file_name = "..\\Solution"+str(instance)+".xlsx")
```

### 3.2.2 Schedule OR60 conference using the matheuristic with a 90 seconds time limit for phase one and 500 seconds time limit overall

```
from Optimisation import *
instance = "OR60"
f_name = "..\\Dataset\\"+str(instance)+".xlsx"
p = Problem(file_name = f_name)
parameters = p.ReadProblemInstance()
p.FindConflicts()
p.AssignTimezonesPenalties(parameters)
sol = Solution(p)
solver = Matheuristic(p, sol)
s_time = time()
solver.solve(s_time, run_time = 500, timelimit = 90)
```

## 3.3 Hyper-heuristic

The hyper-heuristic algorithm does not require a software license and it handles all available constraints including time slot level. It consists of four low-level heuristics, specifically two swap heuristics, a reverse heuristic, and a ruin and recreate heuristic. Its framework involves a two-step iterative process during scheduling optimisation where, in the first step, a low-level heuristic is selected randomly and is applied to the schedule. Then, in the second step, if the modified schedule is not worse than the previous, it is accepted. Otherwise, it is rejected and the previous schedule is restored.

### 3.3.1 Schedule GECCO22 conference using the hyper-heuristic with a 3600 seconds time limit and apply ruin and recreate every 600 seconds

```
from Optimisation import *
instance = "GECCO22"
f_name = "..\\Dataset\\"+str(instance)+".xlsx"
p = Problem(file_name = f_name)
parameters = p.ReadProblemInstance()
p.FindConflicts()
p.AssignTimezonesPenalties(parameters)
sol = RandomInd(p)
solver = HyperHeuristic(p, sol)
s_time = time()
solver.solve(s_time, run_time = 3600, rr = 600)
print("Objective Value:", sol.EvaluateSolution())
print("All submissions scheduled?", sol.EvaluateAllSubmissionsScheduled())
sol.printViolations()
```

## 3.4 Additional Use Cases

### 3.4.1 How to configure the "GLPK_CMD" free solver in integer programming model

To configure the "GLPK_CMD" solver follow the next two steps:

*Step 1:* Open the *Optimisation.py* file.

*Step 2:* Go to line 470 and comment it out, then uncomment line 471 as shown next for the exact model. Similarly, comment out line 774 and uncomment line 775 for the extended model.

```
#model.solve(GUROBI(msg = 0, MIPGap = 0, timeLimit = timelimit))
model.solve(GLPK_CMD(msg = 0))
```

For more information visit `https://coin-or.github.io/pulp/guides/how_to_configure_solvers.html`

### 3.4.2 How to manually edit an obtained schedule and observe the impact on schedule's quality

Suppose the conference scheduler has generated the N2OR schedule, *SolutionN2OR.xlsx*, which is located in *Solutions* folder. To manually edit the schedule follow the next steps:

*Step 1:* Open the *SolutionN2OR.xlsx* file (e.g., Figure 3.1).

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | **Steelhouse LT** | **Stafford 1** | **Stafford 2** | **Steelhouse 1** |
| 2 | **Wed1** | Optimisation | Education | Big Data and AI | Metaheuristics |
| 3 | **Wed2** | Optimisation | Education | Modelling and Simulation | Metaheuristics |
| 4 | **Thu1** | Supply Chain & Transportation Management | Analytics | Modelling and Simulation | Metaheuristics |
| 5 | **Thu2** | Optimisation | Analytics | Big Data and AI | Consultancy |
| 6 | | | | | |
| 7 | **Wed1** | NEW19A3731 | NEW19A3758 | NEW19A3728 | NEW19A3750 |
| 8 | **Wed1** | NEW19A3747 | NEW19A3759 | NEW19A3735 | NEW19A3750 |
| 9 | **Wed2** | NEW19A3740 | NEW19A3722 | NEW19A20 | NEW19A22 |
| 10 | **Wed2** | NEW19A3745 | NEW19A3723 | NEW19A3734 | NEW19A3742 |
| 11 | **Thu1** | NEW19A3730 | NEW19A3738 | NEW19A3743 | NEW19A21 |
| 12 | **Thu1** | NEW19A3746 | NEW19A3754 | NEW19A3748 | NEW19A3756 |
| 13 | **Thu2** | NEW19A10 | NEW19A3729 | NEW19A19 | NEW19A11 |
| 14 | **Thu2** | NEW19A3721 | NEW19A3736 | NEW19A3737 | NEW19A3733 |
| 15 | **Thu2** | NEW19A3739 | NEW19A3755 | NEW19A3744 | NEW19A3749 |

FIGURE 3.1: N2OR initial schedule

*Step 2:* Proceed with editing and save the file. For example, swap the rooms of Education track and Big Data and AI track in session Wed1. Due to this change, the following submissions must also be swapped rooms: NEW19A3758 and NEW19A3759 with NEW19A3728 and NEW19A3735. This will result in a new schedule as shown in Figure 3.2.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | **Steelhouse LT** | **Stafford 1** | **Stafford 2** | **Steelhouse 1** |
| 2 | **Wed1** | Optimisation | Big Data and AI | Education | Metaheuristics |
| 3 | **Wed2** | Optimisation | Education | Modelling and Simulation | Metaheuristics |
| 4 | **Thu1** | Supply Chain & Transportation Management | Analytics | Modelling and Simulation | Metaheuristics |
| 5 | **Thu2** | Optimisation | Analytics | Big Data and AI | Consultancy |
| 6 | | | | | |
| 7 | **Wed1** | NEW19A3731 | NEW19A3728 | NEW19A3758 | NEW19A3750 |
| 8 | **Wed1** | NEW19A3747 | NEW19A3735 | NEW19A3759 | NEW19A3750 |
| 9 | **Wed2** | NEW19A3740 | NEW19A3722 | NEW19A20 | NEW19A22 |
| 10 | **Wed2** | NEW19A3745 | NEW19A3723 | NEW19A3734 | NEW19A3742 |
| 11 | **Thu1** | NEW19A3730 | NEW19A3738 | NEW19A3743 | NEW19A21 |
| 12 | **Thu1** | NEW19A3746 | NEW19A3754 | NEW19A3748 | NEW19A3756 |
| 13 | **Thu2** | NEW19A10 | NEW19A3729 | NEW19A19 | NEW19A11 |
| 14 | **Thu2** | NEW19A3721 | NEW19A3736 | NEW19A3737 | NEW19A3733 |
| 15 | **Thu2** | NEW19A3739 | NEW19A3755 | NEW19A3744 | NEW19A3749 |

FIGURE 3.2: N2OR edited schedule

*Step 3:* Open the *Solver_Checker_v1.1.py* file, edit as needed and run it to obtain the new schedule with the updated violations. It is possible to view the impact of the changes directly by printing on the console or by generating a new Excel file (violations sheet). Below is the code for the N2OR example.

```
from Solution import *
p = Problem(file_name = "..\\Dataset\\N2OR.xlsx")
parameters = p.ReadProblemInstance()
p.FindConflicts()
p.AssignTimezonesPenalties(parameters)
sol = Solution(p)
sol.ReadSolution(file_name = "..\\Solutions\\SolutionN2OR.xlsx")
print("Objective Value:", sol.EvaluateSolution())
print("All submissions scheduled?", sol.EvaluateAllSubmissionsScheduled())
print("Is Solution Valid?", sol.ValidateSolution())
sol.printViolations()
sol.toExcel(file_name = "New_Solution.xlsx")
```