Lancaster University

An Investigation into OR techniques for Conference Scheduling Problems

Yaroslav Pylyavskyy, BSc, MSc

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Lancaster University Management School
Department of Management Science

July 2025

Abstract

Academic conferences provide great benefits to their participants and stimulate the advancement of knowledge. In the hope of exploiting fully a conference though, an effective schedule is required. Given that many conferences have different constraints and objectives, different mathematical models and heuristic methods have been designed to address rather specific requirements of the conferences being studied per se. The aim of this thesis is the investigation of different operations research tools for the creation of a generic conference scheduler applicable to many conferences. In chapter 3, a penalty system is presented that allows organisers to set up scheduling preferences for tracks and submissions. A generic scheduling tool based on two integer programming models is presented which schedules tracks into sessions and rooms, and submissions into sessions by minimising the penalties subject to certain hard constraints. Then, in chapter 4, a decomposed two-phase matheuristic solution approach is presented as an alternative approach to mathematical models that struggle for some conference scheduling problems. The results showed that the matheuristic finds near-optimal solutions and finds solutions for instances where the mathematical model fails to provide solutions within the one hour time limit. Next, in chapter 5, we make benchmark data publicly available to facilitate the comparison and evaluation of different developed methods for conference scheduling problems. In addition, we present a selection hyper-heuristic algorithm to solve the benchmark instances and provide computational results. The aim is to encourage researchers to contribute to the benchmark dataset with new instances, constraints, and solving methods. In chapter 6, we present extended formulations of mathematical models to handle constraints that need to be resolved on time slot level. Lastly, we compare the performance of all developed methods by solving all available instances and highlight the benefits and limitations of each method.

Acknowledgements

I would like to express my gratitude to several individuals and organisations whose support and guidance have been vital throughout my PhD journey.

First and foremost, I am grateful to my supervisors, Dr. Ahmed Kheiri and Dr. Peter Jacko. Their support, insightful advice, and constant encouragement have been pivotal in the successful completion of this research. Their expertise and dedication have not only guided my work but have also inspired me to strive for excellence. I appreciate their patience and the countless hours they have invested in reviewing drafts and providing constructive feedback.

I would also like to extend my sincere thanks to the UK Research and Innovation (UKRI) for their generous funding and support. This research would not have been possible without the financial assistance provided by the UKRI, which helped me to focus on my research.

Additionally, I am thankful to Lancaster University for providing an excellent, inspiring and resourceful environment that made this research possible. Special thanks to Dr. Ivan Svetunkov and Prof. Adam Letchford for their valuable advises and support.

Lastly, I would like to acknowledge my family and friends for their support and encouragement. Their belief in my abilities and their emotional support have been a constant source of strength throughout this journey.

Thank you all for your extremely helpful contributions and support.

Statement of Authorship

Chapter 3

Conceptualisation and study design: I had the idea of formulating the integer program models and designed the experimental framework.

The co-authors contributed by suggesting the format of the input file, the penalty system, the weighted sum method and provided critical feedback for the improvement of integer program models.

Data collection and analysis: I developed the whole codebase and mathematical models, run 100% of the experiments and validated the results.

The co-authors contributed by providing the data, reviewing the codebase and the mathematical formulations, validating the results, and suggesting improvements on models' formulations.

Writing and editing: I wrote the initial draft of the manuscript and coordinated feedback from co-authors and reviewers to finalise the manuscript.

The co-authors contributed by reviewing the manuscript and suggesting improvements.

Visualisation and presentation: I created all the tables and figures, as well as the appendix.

The co-authors contributed by suggesting improvements on the tables.

The co-authors contributed as follows: Dr. Peter Jacko provided critical feedback for the improvement of integer program models, reviewed the manuscript and mathematical formulations, validated the results and suggested improvements on tables. Dr. Ahmed Kheiri secured the funding, suggested the format of the input file, the penalty system, the weighted sum method and improvements on tables, provided the data, reviewed the codebase and the manuscript, and validated the results.

Chapter 4

Conceptualisation and study design: I had the idea of the matheuristic method and designed the experimental framework.

The co-authors contributed by providing feedback.

Data collection and analysis: I developed the whole codebase, run 100% of the experiments and validated the results.

The co-authors contributed by providing the data, reviewing the codebase and the matheuristic method, providing feedback for the improvement of the matheuristic method, and validating the results.

Writing and editing: I wrote the initial draft of the manuscript and coordinated feedback from co-authors to finalise the manuscript.

The co-authors contributed by reviewing the manuscript and providing feedback.

Visualisation and presentation: I created all the tables.

The co-authors contributed by suggesting improvements on the tables.

The co-authors contributed as follows: Dr. Ahmed Kheiri provided the data, reviewed the codebase and the hyper-heuristic algorithm (phase 2), validated the results, and provided feedback. Dr. Peter Jacko reviewed the integer program formulation (phase 1), validated the results, and provided feedback on the manuscript as well as improvements on the integer program formulation.

Chapter 5

Conceptualisation and study design: I designed the initial experimental framework.

The co-authors contributed by suggesting the benchmarking idea, suggesting improvements on the experimental framework and providing feedback.

Data collection and analysis: I developed the whole codebase, run 100% of the experiments and validated the results.

The co-authors contributed by providing the data, reviewing the codebase and validating the results.

Writing and editing: I wrote the initial draft of the manuscript and coordinated feedback from co-authors to finalise the manuscript.

The co-authors contributed by reviewing the manuscript and suggesting improvements.

Visualisation and presentation: I initially created all the tables.

The co-authors contributed by improving the tables.

The co-authors contributed as follows: Dr. Ahmed Kheiri suggested the benchmarking idea, suggested improvements on the experimental framework, provided the data, provided feedback, reviewed the codebase, validated the results, reviewed the manuscript and improved the tables. Dr. Peter Jacko provided feedback, validated the results and reviewed the manuscript.

Chapter 6

Conceptualisation and study design: I designed the experimental framework.

The co-authors contributed by suggesting the idea and providing feedback.

Data collection and analysis: I run 100% of the experiments and validated the results.

The co-authors contributed by providing the data, reviewing the codebase, providing feedback, and validating the results.

Writing and editing: I wrote the initial draft of the manuscript and coordinated feedback from co-authors and reviewers to finalise the manuscript.

The co-authors contributed by reviewing the manuscript and providing feedback.

The co-authors contributed as follows: Dr. Ahmed Kheiri suggested the idea, provided the data, reviewed the codebase and manuscript, provided feedback and validated the results. Dr. Peter Jacko suggested the idea, provided feedback, reviewed the manuscript and validated the results.

Yaroslav Pylyavskyy

July 2025

Statement of Originality

This thesis has not been submitted in support of an application for another degree at this or any other university. It is the result of my own work and includes nothing that is the outcome of work done in collaboration except where specifically indicated. Many of the ideas in this thesis were the product of discussions with my doctoral supervisors Dr. Ahmed Kheiri and Dr. Peter Jacko.

Chapter 3 of this thesis has been published as: 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.

Chapter 4 of this thesis has been submitted to *Journal Of Scheduling*: Yaroslav Pylyavskyy, Ahmed Kheiri, and Peter Jacko (2025) A Two-phase Matheuristic Approach to Conference Scheduling Problems.

An abridged version of Chapter 5 of this thesis has been accepted at *GECCO 2025* Conference: Ahmed Kheiri , Yaroslav Pylyavskyy, and Peter Jacko (2025) CoSPLib – A Benchmark Library for Conference Scheduling Problems.

Chapter 6 of this thesis has been published as: Y. Pylyavskyy, A. Kheiri and P. Jacko, Exact and Hyper-heuristic Methods for Solving the Conference Scheduling Problem, *2024 International Conference on Decision Aid Sciences and Applications (DASA)*, Manama, Bahrain, 2024, pp. 1-5, doi: 10.1109/DASA63652.2024.10836575.

Appendix B of this thesis has been published as: Ahmed Kheiri, Yaroslav Pylyavskyy, and Peter Jacko. 2025. Automated Scheduling of GECCO 2023. SIGEVOlution 17, 3, Article 2 (September 2024), 7 pages. https://doi.org/10.1145/3717408.3717410

Yaroslav Pylyavskyy

July 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conferences are formal events of great importance to academic communities as they provide numerous benefits to participants and stimulate the advancement of knowledge. They provide the opportunity to academics and researchers to share their latest findings, exchange ideas, receive critical feedback, and network with other peers across different institutions and backgrounds. In the hope of exploiting fully a conference though, an effective schedule is required. However, the development of an effective schedule is not trivial and usually poses a challenging task for conference organisers due to multiple preferences and constraints involved. Conferences are usually scheduled by a group of organisers manually, which is an arduous and often error-prone process done under time pressure. Additionally, the schedule usually requires last-minute changes, after being already published, resulting in an overwhelming experience overall. In the past, organisers were happy to achieve any feasible solution and would stop the scheduling process at that point without considering optimisation of the conference (Sampson, 2004). However, nowadays, the complexity of conference scheduling has shifted towards the quality of the solution.

1.1 Literature Review on Conference Scheduling Problems

As noted by Thompson (2002), conference scheduling problems can be tackled through a Presenter-Based Perspective (PBP) or an Attender-Based Perspective (ABP). The PBP focuses on a schedule that prioritises presenters' preferences to enhance their satisfaction, whereas the ABP concentrates on maximising attendee satisfaction. Some

research, however, has embraced a mixed approach, considering the optimisation of both presenters' and attendees' preferences.

1.1.1 Presenter-Based Perspective

Potthoff and Munger (2003) studied the conference scheduling problem (CoSP) of 2001 San Antonio meetings of the Public Choice Society as a test case. The conference was attended by 333 participants and included in total 14 subjects of areas, 96 panels, and 10 time periods that for each one a maximum number of 10 time slots were available. The authors collected data from the conference booklet that was distributed to attendees. The panels were not assigned to the subject areas in the booklet, thus Potthoff and Munger (2003) did the assignment themselves, though they do not provide details about this process. In contrast to the studies presented so far, they did not consider attendees' preferences collection in their work. They justify this decision by explaining that this would have significantly increased the problem complexity and this is the reason why previous studies implemented heuristics rather than exact methods. Therefore, the authors solved the problem of assigning sessions in time slots using integer programming. The objective was to minimise the conflicts of attendees for competing to attend sessions of the same time slot subject to the following constraints; 1) each time slot has no more than 10 sessions scheduled (less is allowed), 2) each panel is scheduled only once, 3) presenters should not be scheduled to present more than one session of the same time slot, and 4) the number of sessions from a subject area should be within the ceiling and floor limits in each time slot. The authors stated that their implemented method reached the optimal solution with all constraints satisfied.

Potthoff and Brams (2007) extended the previous work by applying the proposed IP method to the annual meetings of the Public Choice Society in New Orleans for the years 2005 and 2006. In addition, they presented some issues encountered during the scheduling process and suggested ideas on how to overcome them in the future. The 2005 annual meeting required the assignment of 76 sessions belonging to 13 subject areas into 9 time slots, whereas the 2006 annual meeting included 45 sessions within 6 subject areas, and again 9 time slots were available. In contrast to previous annual meetings, panel chairs or discussants were not required in 2005 and 2006. Similarly to Potthoff and Munger (2003), the objective was to produce a schedule which minimises the conflicts of attendees competing to attend sessions of the same time slot. An extra constraint was added to the existing formulation which indicated all panels along with

the respective time slots for which a presenter is unavailable to attend. Both solutions to the annual meetings were feasible having all constraints satisfied and successfully accomplished the objective. One of the issues that the authors encountered during the scheduling process were late cancellations of papers. This was particularly problematic when the presenter had to deliver more than one paper. Another issue was with late accepted papers for which the best panel matches were already full and thus readjustments on the schedule were required. Additional readjustments were required because some participants declared unavailability at late notice for already scheduled time slots. To overcome these issues, Potthoff and Brams (2007) suggested some ideas such as; i) introducing a non-refundable registration fee as a requirement for abstract acceptance, ii) rejecting late submission abstracts or accept these but with a higher registration fee and assign these to panels with no common topic, iii) not allowing one person to submit multiple abstracts or allow but collect extra fees which would be returned after successful completion of the conference, and iv) collect extra fees for presenters who require a specific time slot excluding those of exceptional reasons.

Edis and Edis (2013) considered an artificial conference including 10 subject areas and 170 presentations with a 3 days time span. Each day had 4 time periods with approximately three parallel sessions available for which up to 5 presentations could be scheduled. The authors tackled the problem by implementing an integer programming model and an extended version of it which addressed a secondary objective. While the goal of the primary objective was to minimise the concurrent occurrence of same or similar subject areas within the same time period in each day, the aim of the secondary objective was to distribute the number of presentations into time periods in a balanced manner. In the primary integer programming model, the authors included the following constraints; 1) determination of time periods with the same subject area within parallel sessions in a day, 2) assignment of each presentation into a session only once, 3) each session should have at most one subject area assigned, 4) only a limited number of presentations, from the same subject area, is assigned to each session, 5) the difference between the number of presentations from the same subject area but in different sessions should be equal to or less than one, 6) allowance of at most one "problematic" presentation to be scheduled into any session in a day. In the extended integer programming version, Edis and Edis (2013) considered the participants time preferences of attending or not a presentation and the time preferences of the Program Chair for presentations assignment. Thus, the following additional constraints were included in their model; 7) each presentation associated with a time preference should be scheduled exactly on the time period and day of preference, 8) each presentation associated with a time preference

should not be scheduled on non-preferred time period and day, and 9) identification of time periods for which the number of presentations in all parallel sessions is unequal. The results of this study showed that the primary objective was successfully obtained by the primary integer programming model. However, the secondary objective was not achieved by the primary model and the authors tackled this problem by implementing the extended integer programming version. They claimed that the extended version managed to achieve the optimal solution by reducing the violations number from eight to one.

1.1.2 Attender-Based Perspective

Eglese and Rand (1987) worked on a case study about the Tear Fund conference which occurred in July 1985, officially introducing for the first time the CoSP. The duration of the conference was 4 days in total, each day had 1 session available and 7 rooms were available. In June (one month before the conference), attendees of the conference were sent a detailed description list of the 15 seminars. They were prompted to declare four preferred seminars to attend and a reserve choice. Due to insufficient technology means of that time (eg. submitting preferences manually), only the first 50 preferences forms out of the 265 forms were used in scheduling the conference. The conference was scheduled manually and the organiser estimated that 10% of the participants would attend their reserve choice. The estimation was quite accurate as in fact around 8% of participants did so (one of the authors was among those participants). Eglese and Rand (1987) mentioned that the forms were problematic in two ways. Firstly, the forms did not capture the level of preference strength between choices and secondly, it was not explicitly mentioned if a statement of preference was implied among the first four choices. The authors estimated that around 20% of the returned forms were inaccurate because the first four preferences were in an alphabetical order. They argued that these drawbacks were significant and played a vital role in timetable evaluation. Therefore, they evaluated the timetables according to a weight assignment system. A score, which was a sum of weights, was assigned to each participant based on the seminars that they were scheduled to attend. This score was calculated by considering the order of the preferred seminars in the forms, while the weight for a participant being assigned a non preferred seminar was slightly worse than being assigned their reserve choice. To this end, the objective of the problem was to minimise the total sum of weights. In addition, a secondary objective was to equalise the number of participants attending a replicated seminar. The authors implemented a heuristic solution rather than an exact method due

to the size of the problem. Constraints included in the formulation were; availability of rooms (7 in total), seminars replication was allowed for as many times as needed and avoiding very large or small seminar groups. The constraints of blackout facilities availability and the fact that one seminar leader was responsible for two seminars were not included in the formulation. The authors stated that these constraints were not problematic, but could be easily included if needed, and they checked the solutions to ensure they were not violated. Their developed heuristic involved two stages; At the first stage a feasible solution was generated, while at the second stage the solution was improved through an annealing algorithm. The researchers accomplished to reduce the number of participants who attended a non-preferred or reserve choice to zero with their proposed solution.

Sampson and Weiss (1995) were the next who tackled the CoSP, in 1995. In their work, they generated their own conference data randomly and used these for comparison purposes between the developed heuristic solutions and exact solutions. Although the process of problems generation was not presented in this study, it was discussed in detail in their extended work, Sampson and Weiss (1996). They also included attendees' preferences collection and defined a problem size parameter for the number of requests per participant. Preferences were handled based on a priority order in which requests of high-priority were considered first. The authors argued that priority order is more advantageous than participant order for several reasons. The main drawback of participant order is that attendees reviewed in the early phase had a greater probability of being allocated to a non-full session than attendees reviewed later. Priority order, on the other hand, provides better enrolment participant satisfaction and enhanced equity between attendees. The focus of their study involved the enrolment process of attendees based on priority order, and the scheduling of sessions into time slots and rooms. Replication of sessions was allowed in their problem for the sake of increasing availability to attendees. Unlike Eglese and Rand (1987), the authors of this study included finite capacity of rooms. The objective of their study was to maximise the satisfaction of attendees in terms of attending preferred sessions. They tackled the problem by implementing a local search heuristic which simultaneously solved both enrolment and scheduling phases. The following constraints were considered; i) every participant is not assigned to more than one sessions in each time period, ii) participants are not allowed to attend more than one replicated session of their preference, iii) attendees are not assigned to sessions of unspecified time periods, iv) assignment of attendees does not exceed specified room capacities, v) the defined number of each session offerings are scheduled, vi) moderators (who are involved in more than one session) are not assigned into two sessions of

the same time slot, vii) restriction to certain rooms and/or time periods is allowed for certain sessions, and viii) schedule is verified in terms of rooms capacity. The authors compared the solutions of the heuristic to exact solutions for which optimal solutions were known. The results revealed that their heuristic method yielded near optimal solutions in a few seconds, whereas exact methods required a few hours, especially for large problems. In addition, they highlighted that simultaneous approaches are more efficient than sequential or traditional ones as far as enrolment and scheduling phases are concerned.

Sampson and Weiss (1996), extended their previous work by adding sensitivity analysis to the problem and described the process of the conference data generation. For the process of data generation, the authors considered the following parameters; number of sessions, conference participants, number of time periods, and average number of offerings per session. Requests per participant, which represents the collection of attendees preferences, were selected and sorted in a random manner. The selection of sessions by participants was determined by a probability, which was proportional to the average number of offerings per session. The authors determined room utilisation and capacity utilisation parameters, instead of directly assigning random values to rooms and capacity. The former parameter defines the average occupied percentage of rooms for each conference period, while the latter defines the estimation of the occupied total seating in percentages. Note that all rooms were considered to have equal capacities. The focus of their study was to provide useful insights regarding the solution sensitivity when problem parameters change by analysing four key subjects of the problem. Firstly, they explored the impacts of the length of the conference, the number of offerings per session and attendees' satisfaction on the solution sensitivity. Secondly, they measured the sensitivity of participants satisfaction to rooms availability. Thirdly, they investigated whether some conference time periods (T) could be reduced given that the demand of attendees for sessions (R) is smaller ($R < T$). Finally, they conducted an analysis to determine the level of seating capacity utilisation that could be considered sufficient. The authors used simulation to obtain the results of their sensitivity analysis. Overall, the results revealed that offering some sessions more than once significantly increases the attendees satisfaction with minimum costs. They mentioned that up to that period, rarely had popular conferences followed the specific strategy. Additionally, they highlighted that conference organisers should focus on selecting rooms based on the size parameter rather than deciding the number of offered rooms. Another key finding was that flexibility is essentially increased when more time periods are added in comparison to more rooms being added. In further analysis, they stated that the determination of

an 'optimal' conference duration strongly depends on the collection of attendees' preferences. Lastly, they mentioned that a 'safety stock' of seating capacity is beneficial for ease of enrolment, while over-utilised capacities limit attendees' enrolment but at the same time might have a side benefit of reducing potential scheduling conflicts.

Le Page (1996) tackled the CoSP of the American Crystallographic Association 1995 Annual Meeting, which was attended by over 1100 participants. The conference involved 26 subject areas including 35 half-day sessions scheduled in 5 rooms of different sizes. Those sessions were scheduled in parallel and some of them required to be scheduled in specific order. Le Page (1996) was the first to introduce parallel sessions and sessions ordering in a CoSP. Prior to the scheduling process, the author collected session preferences from 102 attendees via e-mail. The e-mail requested attendees to submit seven sessions which they would mostly desire to attend. The objective of the problem was to assign the sessions into days and rooms by minimising the number of attendees competing to attend sessions of their desire. A novel semi-automated heuristic algorithm was developed to accomplish the objective. Although the algorithm was fast and generated a much better solution than a manual schedule, its main drawback was that it could not handle all the constraints, which means that a manual intervention was required during the scheduling process. Specifically, the problematic constraint was the ordering of particular sessions and the author claimed that softening this constraint should be ultimately decided by humans.

Sampson (2004) presented the scheduling process of the 2001 Annual Meeting of the Decision Sciences Institute Conference. This conference included 213 sessions with 10 time periods and 37 rooms available for sessions assignment and was attended by 1086 participants. The author collected attendees' preferences through the web for the conference scheduling process. In total, 520 participants submitted their preferences in a ranked order with a maximum allowed limit of 36 preferred discussions per participant. These submissions were used to create a weighted parameter matrix. The objective of the problem was to assign all sessions into time periods and rooms in such way that the participants' satisfaction is maximised (the author defined this problem as *Preference-Based Conference Scheduling*). Although Sampson (2004) provided a formulation of the problem, he implemented a simulated annealing heuristic to solve it due to the NP-Complete nature of the formulation. The formulation involved the following constraints; 1) participants are assigned only in their preferred sessions, 2) participants are not allowed to be in more than one place at the same time, 3) participants are only assigned to scheduled sessions and do not exceed the capacity limits, 4) each session is scheduled

in only one place, 5) presenters are not allowed to be in more than one place at the same time, 6) available time periods and rooms for each session are defined, and 7) only one session is assigned in each room at each given time periods. The quality of the generated schedule was evaluated by a survey which was distributed to all participants and by follow-up emails sent to participants who submitted preferences. In all, 230 participants completed the survey from which 211 participants had attended the same meeting of the previous year. Results indicated that the quality of the schedule was satisfying and better than the previous year meeting.

Zulkipli et al. (2013) tackled the Capacity Planing CoSP which involved 3 subject areas, 5 sessions, 60 papers, 3 rooms, and 3 parallel time periods with 4 time slots each. They asked participants to submit their preferences of attending paper presentations in a ranking order, starting from 1 to 10. Then, they assigned a weight to each paper in a respective manner. These weights were then used to form the objective function, which aimed to assign the papers into rooms and time slots in such way that each time period achieves a balanced number of papers with respect to the weights. The authors implemented a goal programming method to eliminate the deviations of each time period subject to the following constraints; i) maximise weight under-achievement deviations of all time slots, ii) minimise weight over-achievement deviations of all time slots, iii) each paper is presented only once, and iv) each time period should have a minimum number of papers presented. In the final analysis, the authors compared their generated schedule to the actual schedule of the conference. Results indicated that the proposed schedule was at 93% optimal by achieving only a weight over-achievement deviation equal to 3 at a specific time slot, whereas the actual schedule had a score of 374 for weight over-achievement deviations and a score of 234 for weight under-achievement deviations.

Quesnelle and Steffy (2015) conducted a case study by using real data from an older conference, namely The 2013 PenguiCon Conference. This conference involved 253 presentations, 195 presenters, 14 rooms, and it was attended by around 1000 participants. In addition to this, the authors included the factor of participants' preferences in their work by generating these based on the actual attendance recorded for each talk of the conference. In this study, they provided problem definitions and showed that the scheduling problem under study along with some variants are all NP-Hard. They specifically defined and focused on the *Extended Conference Timetable Decision Problem (ECTTD)* and the *Preference Conference Optimization Problem (PCO)* which are both variants of the typical CoSP. The former defined problem is a variant of *Conference Timetable*

Decision Problem (CTTD) (as described in Quesnelle and Steffy (2015)) in which additional constraints regarding room availability and compatibility are considered. The latter defined problem is a variation that includes an additional set of constraints that considers the participants' preferences. It is identical to the *ABP CoSP* as introduced and described by Thompson (2002) in 2002. Quesnelle and Steffy (2015) presented an integer programming formulation for each problem and achieved optimal solutions for both problems. It should be noted that the authors dramatically decreased the required number of variables in their integer programming models from 27,421,440 variables to only 91,514 variables. They achieved this by manually excluding variables during the modelling phase which were known to get zero value and would ,thus, lead to unnecessary computations. The objective of the ECTTD problem was the assignment of presenters into presentations and time slots, as well as the allocation of presentations into rooms based on their availability and compatibility. The following constraints were included in the formulation; 1) each speaker must be scheduled for all of their presentations, 2) all co-speakers must have the same schedule for their common presentations, 3) if a speaker offers multiple presentations, then these presentations are scheduled in different time slots, 4) the schedule of the rooms is complete, and 5) no room is multi-scheduled. The PCO problem included the objective of minimising participants' preferences conflicts by assigning presenters into presentations and time slots, as well as assigning presentations into appropriate rooms. In the formulation of PCO the authors used the same constraints as in ECTTD and added the following to capture participants' preferences; 6) boolean variables were used to indicate the time slots of each presentation, 7) validate that the previously mentioned variables are correct indicators, and 8) boolean variables were used to indicate concurrent presentations based on boolean variables mentioned in constraint (6). Both models yielded conference schedules that satisfied all the constraints and the PCO problem resulted in none participants' conflicts. As mentioned before, the authors generated artificial participants' preferences. They did so by using a uniform distribution and extended the particular experimentation into generating different participants' preferences following a normal distribution but with different variances. Results of the particular experimentation indicated that the computational time increased in an exponential manner as the variance values were decreased. Quesnelle and Steffy (2015) reported that this issue was due to the symmetric nature of their integer programming model. Hence, they decided to investigate and overcome this by implementing some enhancements in their formulation. Firstly, they grouped all rooms into three room classes and replaced those in the formulation which speed up significantly the solving time. Secondly, they decided to dualise the constraint (3) which reduced the solving

time by 75%. Finally, they compared the Standard PCO model, the Symmetry model, and the Dualised model (Dualised model includes the symmetry reformulation) in terms of computational time and variance. This lead to the confirmation of the previous observation about the relationship of those variables. That is, solving time exponentially increases when variance decreases.

Manda et al. (2019) used the dataset from Ecology 2013 for testing purposes and delivered a schedule for the Evolution 2014 conference. While the former unconstrained conference spanned 5 days including 324 talks, 8 time slots, and 5 parallel sessions, the latter constrained conference spanned 4 days including 1014 talks, 16 time slots, and 14 parallel sessions. The objective of this study was to schedule all talks into time slots and parallel sessions with the purpose of maximising the coherence within sessions and minimising similarity between parallel sessions. To do so, the researchers followed a novel approach by extracting keywords from the title and abstract of each talk and implemented the Latent Dirichlet Allocation (LDA) algorithm to generate topic models. Then, they developed their own metric, called Match Percentage, to evaluate the fitness of the generated topic models. In addition, the authors created another metric, namely the Discrimination Ratio, that captures the session coherence and the dissimilarity between parallel sessions and was used as the objective function for maximisation. For the initialisation process, three different approaches were implemented; Random, Greedy, and Integer Linear Programming (ILP). These initial solutions were then further optimised and compared by two different heuristic methods and two different optimisation methods. For the first optimisation method defined as stochastic optimisation, the authors implemented a Hill Climbing algorithm (HC) and a Simulated Annealing algorithm (SA) with a swap operator. In the second optimisation method defined as sequential optimisation, they implemented variants of HC and SA which split the optimisation process into two parts. Firstly, the algorithms maximise similarity within sessions and secondly, the algorithms minimise similarity between sessions. Both heuristic methods and optimisation methods were tested on the Ecology 2013 dataset to determine their efficiency. The researchers reported that ILP generated the best initial solution and SA yielded the best final solution on average within 50 runs but, ultimately, irrespective of the initialisation approach taken, the final schedules had similar quality. With regard to the optimisation methods, the authors reported that stochastic optimisation outperformed the sequential optimisation. After this experimentation, they decided to proceed with only a random initialisation approach and implemented both optimisation methods including both heuristic methods to generate the schedule for both constrained and unconstrained Evolution 2014 conferences. The results were similar to the Ecology

2013 and surprisingly revealed that the best solution obtained included the constraints. In the final analysis, the final schedule delivered was significantly altered by the program committee and the authors decided to receive a further evaluation by 29 volunteers who were experts in evolutionary biology. The results were statistically tested and showed no significant differences between the delivered schedule and the altered schedule. These results might imply that there is space for further research regarding the topic modelling automation and whether it is possible to perform this in an objective manner.

1.1.3 Mixed Approach

Thompson (2002), conducted two experiments with the aim of comparing the solution quality of both manual and computer-based ABP conference scheduling approach to PBP conference scheduling approach. In both of his experiments he considered rooms of different capacities, rooms availability, and presenters' requirement to offer replicated sessions if necessary. The author used data from a real conference to generate three artificial conferences with similar characteristics for the first experiment. The problem involved parallel scheduling of 12 sessions into 3 time slots and 4 different rooms. The number of attendees was 100 in total and the number of presenters was 10, from which two were assigned to present 2 sessions each. The range of preferences collected was between 3 and 6 for each participant and were weighted. The weight assignment followed a descending system with participants attending their first choices being assigned a high score and a low score for their last choices. This weight system was used to evaluate the solutions quality. The three artificial conferences were scheduled manually (ABP), randomly (PBP), and by the proposed method, a simulated annealing heuristic (ABP). The second experiment involved data from another real conference including 64 sessions, 8 time slots, and 8 different rooms. In this conference, some sessions were allowed to be repeated, while others were not. A total number of 175 participants, including presenters, attended the conference and submitted preferences within a range of 0 to 8 sessions. Presenters, when not presenting a session, also attended sessions and submitted preferences. In this experiment, the weight system was different because attendees did not specify preference order of sessions and were thus assigned a fixed score for every desired session they were able to attend. In contrast to the first experiment, the manual scheduling was skipped for this experiment due to complexity level. The results of the two experiments were statistically compared and revealed the following key findings; i) no statistical significance was found between manual and random scheduling methods, ii) a statistical significance at 0.001 level existed between random (or manual) and heuristic

methods, and iii) while the heuristic method produced only slightly better solutions (3 - 4.4%) than the other methods for the first experiment, the solutions were much better (15.9%) for the second experiment. In other words, the author stated that his proposed method contributes more to the quality of the solution as the problem size and complexity increase.

Nicholls (2007) developed a simple heuristic algorithm to schedule the 2003 Western Decision Science Institute Annual Meeting. This conference involved 330 registered attendees, 295 papers, 73 regular sessions including four papers in average, 11 special sessions (a whole session is required for a paper), and 7 rooms of different size. Its duration was 2 full days and three sessions were held during the third day. Although the conference practice did not include a formal attendees' preferences collection, the ease of stream access was considered during the scheduling process along with a small number of presenters' preferences. Nicholls (2007) explicitly stated that his developed method was used to assist the Program Chair during the scheduling process rather than autonomously produce the conference schedule. In other words, it was a tool that assisted the scheduling process but did not provide a complete solution. Additionally, the author stated that the proposed method did not have an objective function *per se*, but its main purpose was to resolve conflicts by utilising a set of rules which considers preferences from both presenters and attendees. Thus, the heuristic aimed to generate a feasible solution and did not optimise the solution further. The constraints considered in this work were; i) meet the preferences of Program Chair who decides which attendees' preferences to consider, ii) ensure that no author conflicts exist, iii) ensure that time slots have not been exceeded, and iv) meet the presenters' preferences. Nicholls (2007) stated that the generated schedule satisfied all the constraints according to feedback collected by participants. Finally, he highlighted that more advanced methods are required to solve larger conferences with more attendees' and presenters' preferences included.

Stidsen et al. (2018) tackled the CoSP of the EURO2016 Conference, which is considered one of the largest OR conferences globally. This particular conference included 25 areas of subject, 124 streams, 463 sessions, 11 time periods (for each typically 4 time slots were available), 54 rooms, 1600 presenters, and attracted around 2000 participants. The researchers did not consider the attendees' preferences due to unavailability of such data. They stated that because of the design of the registration system it was not possible to obtain such information and redesigning the system was impractical during the scheduling process. In addition to this, another data issue was the unavailability of data regarding the expected attendance for each talk. While historical attendance data

could be used to solve this problem, this information was considered as unreliable by the authors. Therefore, a rule of thumb was used which required that the room size of each stream and the number of talks in the whole respective stream should be proportional. The research objective of this study was to generate a schedule which follows the hierarchical structure of the conference by the implementation of a multi-objective mixed integer programming model. This model had in total 5 objectives which were ranked based on their significance and were sequentially solved following a lexicographic optimisation approach. These objectives were ranked in the following order; 1) minimisation of the number of areas assigned to different buildings, 2) maximisation of the number of related areas assigned to the same building, 3) minimisation of the number of different rooms allocated for each stream, 4) minimisation of the number of time gaps within streams and, 5) maximisation of the residual room capacity. These objectives were subject to constraints based on the hierarchical structure of the conference which were; 1) rooms are only assigned sessions of a stream for which the capacity is satisfied, 2) all sessions belonging to the same stream must be assigned to the same building, 3) no more than one session is allocated to each room in each building in each time slot each day, 4) all sessions of each stream must be assigned and, 5) no parallel sessions are allowed for each stream. The proposed model successfully achieved optimal solutions for two objectives, specifically (1) and (3), and found near optimal solutions for the remainder objectives. In order to evaluate the quality of the schedule, the researchers asked the delegates to answer a questionnaire. The results revealed that the generated schedule was coherent and the areas of subject were satisfyingly assigned to appropriate buildings so as to ease the process of switching buildings. Despite the level of complexity involved in the proposed model, there are some worth mentioning notes that the authors acknowledge. First of all, the author conflicts, which would make the problem much more complex, are circumvented due to the policy of the conference to not allow the same author presenting multiple papers. In addition to this, any potential author conflicts in terms of presenting a paper and chairing another session were addressed at the early stage of the papers submission by the submission system. Secondly, the participants' preferences were not considered and the room utilisation was partially considered. Finally, it is clarified by the authors that the proposed model generates only a high-level schedule, leaving intentionally the low-level schedule to the stream organisers for two reasons. Firstly, they claimed that stream organisers should schedule sessions and talks so as to maintain a logical order and, secondly, there was not sufficient data to generate the low-level schedule. Overall, the proposed method was successful and the same method was used to schedule the IFORS2017 conference and was used for the

scheduling of EURO2018 and IFORS2020.

Vangerven et al. (2018) addressed the CoSP of four conferences, namely the MathSport 2013, MAPSP 2015 & 2017, and ORBEL 2017. Their work was primarily focused on maximising the satisfaction of attendees in terms of attending their preferred talks. Therefore, the authors created a profile for each participant by collecting their preferences via e-mail. Apart from accomplishing their main purpose, profiles were also used by the authors in the process of session chairs selection. A secondary goal of the researchers was the minimisation of session hopping. Vangerven et al. (2018) were the first to introduce the aspect of session hopping in a CoSP. As defined by the authors, session hopping occurs when participants miss parts of their preferred talks because these talks are scheduled in different sessions or rooms and due to presenters not beginning their talks in the scheduled time. A third objective of this work was to satisfy the preferences of the presenters. The authors proposed a hierarchical three-phased approach by implementing integer programming formulations, a dynamic programming approach and a heuristic approach to accomplish the three previously mentioned objectives. In the first phase, they minimise the total missed attendance and by maintaining the total attendance at the maximum level, they minimise the session hopping in the second phase. Notice that in the second phase the authors implemented an integer programming model along with either dynamic programming approach or heuristic approach. They decided to develop a fast heuristic approach because in some cases dynamic programming required several hours to yield a solution. In the third phase, they maintained the levels of the total attendance and the session hopping while they minimised the number of presenters' preferences violations. The authors applied their proposed method to

Table 1.1: Characteristics of conferences from Vangerven et al. (2018)

Name	Parallel Sessions	Talks	Rooms	Registered Attendees	Profiles
MathSport 2013	2	76	2	97	68
MAPSP 2015	3	90	3	-	78
MAPSP 2017	3	87	-	-	58
ORBEL 2017	4	80	-	140	101

four medium size conferences in total, which details are shown in Table 1.1. In the MathSport 2013 conference, the proposed method achieved an optimal solution with 42 scheduling conflicts, zero presenters' preferences violations, and a 96.7% attendees' preferences satisfaction on average. For MAPSP 2015, the optimal objective value was 155 in the first phase (90.17% attendees' preferences satisfaction), 120 in the second phase, and zero presenters' preferences violations. Note that for MAPSP 2015 the authors did

minor modifications to their proposed model in order to satisfy the room capacities constraints. The results for MAPSP 2017 revealed an optimal objective value of 478 in the first phase, 145 in the second phase, and none presenters' preferences were violated. In ORBEL 2017, the authors achieved an optimal objective value of 100 (91.7% attendees' preferences satisfaction), 281 in the second phase, and zero presenters' preferences violations. Overall, the proposed method was proved to be highly efficient by generating optimal solutions for all four conferences. In the final analysis, Vangerven et al. (2018) showed that the CoSP with n parallel sessions is NP-Hard when $n \geq 3$, which adds to the result that the preference conference optimisation problem (PCOP) is NP-Hard as showed by Quesnelle and Steffy (2015).

Patro et al. (2022) defined the *Virtual Conference Scheduling* (VCS) problem in which the goal was to schedule submissions into time slots by maximising the efficiency and fairness objectives. While the former objective maximises the total attendance in the conference, the latter objective maximises the attendees' satisfaction, which depends on their interest in a specific submission and their availability to attend it, and the speakers' satisfaction which depends on whether their submission is scheduled in a time slot that provides high attending availability for the interested attendees. The authors consider the preferences of attendees to attend certain submissions as well as their availability based on time zone information. They presented an integer programming formulation suitable for small conferences and a rounding heuristic along with a clustering approach for larger conferences. Their methods successfully generated balanced conference schedules in terms of efficiency and fairness when tested on real and artificial datasets. However, it should be noted that the VCS problem is restricted to a single track conference scenario without parallel sessions, and the speakers' satisfaction metric does not consider the availability of the speaker to present based on time zone information.

A case study regarding the scheduling of GECCO 2019 was presented by Riquelme et al. (2022). The authors defined the problem as a *Track-Based CoSP* which requires the assignment of tracks into sessions and rooms, and the assignment of submissions into time slots of sessions in such way that the number of missing seats is minimised and certain hard constraints are satisfied. They also presented an instance generator which they used to generate 45 artificial instances with similar characteristics to GECCO 2019. In their study, the authors developed an integer programming model and a simulated annealing heuristic, and evaluated the performance of the two methods by solving the artificial instances. Computational results showed that the integer programming model managed to optimally solve most instances but failed to solve several instances within

the one hour time limit. On the other hand, the simulated annealing replicated the results of the integer programming model and found decent solutions for the unsolved instances within a short amount of time.

Rezaeinia et al. (2024) tackled the CoSP as described in Vangerven et al. (2018) and proposed three optimisation approaches to support the scheduling of LOGMS, INFORMS TSL Workshop and ICSP. Attendees' preferences were collected through an online survey which the authors used to form the objective function, a utility function, of the integer programming formulations. The authors developed two optimisation approaches assuming that submissions have been already grouped into tracks, namely a single integrated model and a two-model decomposition approach. The former approach maximises the utility function by optimising both high and low level schedules simultaneously, whereas the latter approach initially generates a high level schedule by maximising the utility function on a session level and, then, it generates the low level schedule, based on the previously obtained solution, by maximising the utility function on a time slot level. In the third optimisation approach, which is a relaxed model, they allow submissions to be freely scheduled regardless of their assigned track. Following different experimentation scenarios, the results showed that the single integrated model is impractical in terms of computational time and the relaxed model produced schedules where some tracks contained submissions which were too disconnected from each other based on feedback from the organising committee. The two-model decomposition approach proved to be the most effective approach, finding high-quality solutions within a short amount of time and resulting in its adoption to schedule the three conferences reported.

1.1.4 Software for Conference Scheduling Problems

In addition to the CoSP studies, a few software tools exist which aid the scheduling process of conferences. However, such software do not construct the conference schedule per se, but only provide additional information and recommendations to the organisers that are involved in the conference scheduling. One of these software tools is *Cobi* (Kim et al., 2013), which is a scheduling tool that uses community sourcing applications for the collection of preferences and constraints from community members. The collected data is used to resolve the conflicts within a scheduling interface and those conflicts are highlighted so as to visually help organizers identify and resolve them as shown in Figure 1.1. Additionally, the software helps organisers to create cohesive sessions based on the knowledge of paper authors who suggest which of the other papers fits

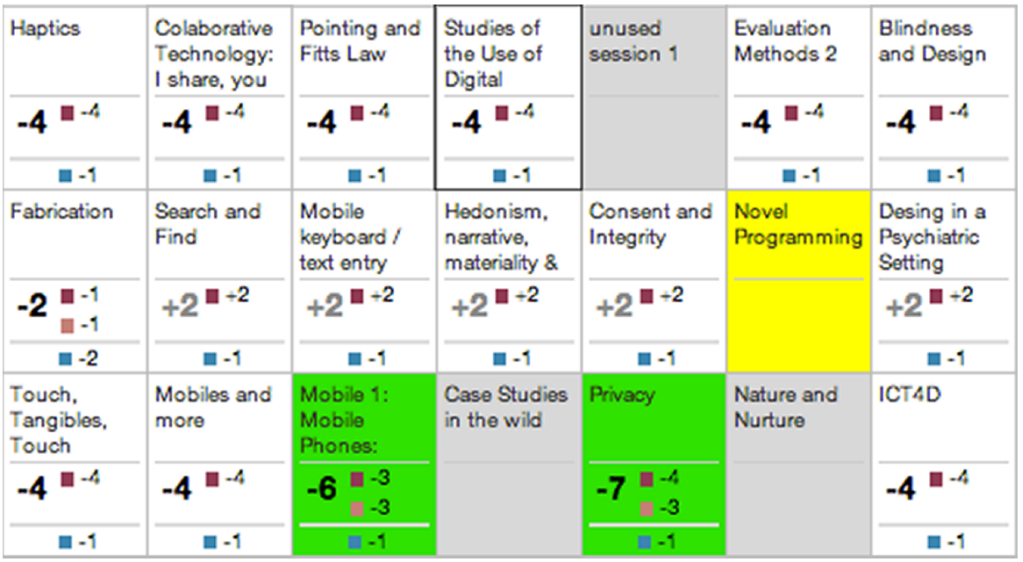


FIGURE 1.1: Scheduling interface of *Cobi* (adapted from Kim et al. (2013))

well with their own. The interface of *Cobi* also aids the organisers by suggesting edits which improve the quality of the schedule, and visually presents the consequences of their potential edits. All edits and final decisions are made by the organisers who are the drivers of the system. *Cobi* considers preferences and constraints related to sessions, papers, and chairs. The system categorises the constraints (or preferences) into; *system-defined* and *community-defined*. The former category ensures that presenters and chairs are at one place at a time, all authors can attend their paper presentation, similar sessions are not scheduled concurrently, and chairs are only allowed to chair sessions in which they do not have a paper. The latter category is associated with presenters' preferences and with the identification of papers that fit well within a session. *Cobi* was used by the organisers of CHI2013 conference to demonstrate its effectiveness, where they successfully managed to resolve 168 out of 238 conflicts.

Confer is another software presented by Bhardwaj et al. (2014), which is a paper-recommendation tool that collects attendees' preferences and constraints. It is designed so as to overcome limitations of *Cobi* and complement each other. In contrast to *Cobi*, *Confer* provides an interface where attendees can explore the exhaustive list of accepted papers and mark all the papers that they wish to attend. Attendees can either directly search for a paper by keyword, author name, affiliation, etc. or choose from the recommendations of the system as shown in Figure 1.2. The data collected within this software is then used to create coherent sessions and aids the scheduling process. Bhardwaj et al. (2014) argue that more coherent sessions can be created by assigning a pair of papers

into the same session given that a large number of attendees wish to attend both of them. However, this could be quite inaccurate as high attendance for two papers does not necessarily mean that the two papers are related. In addition, *Confer* helps organisers to achieve an improved schedule by considering attendees' conflicts reduction. Organisers avoid scheduling papers preferred by many attendees concurrently. Lastly, *Confer* is useful for determining popularity of papers so as to schedule those accordingly in rooms of appropriate size.

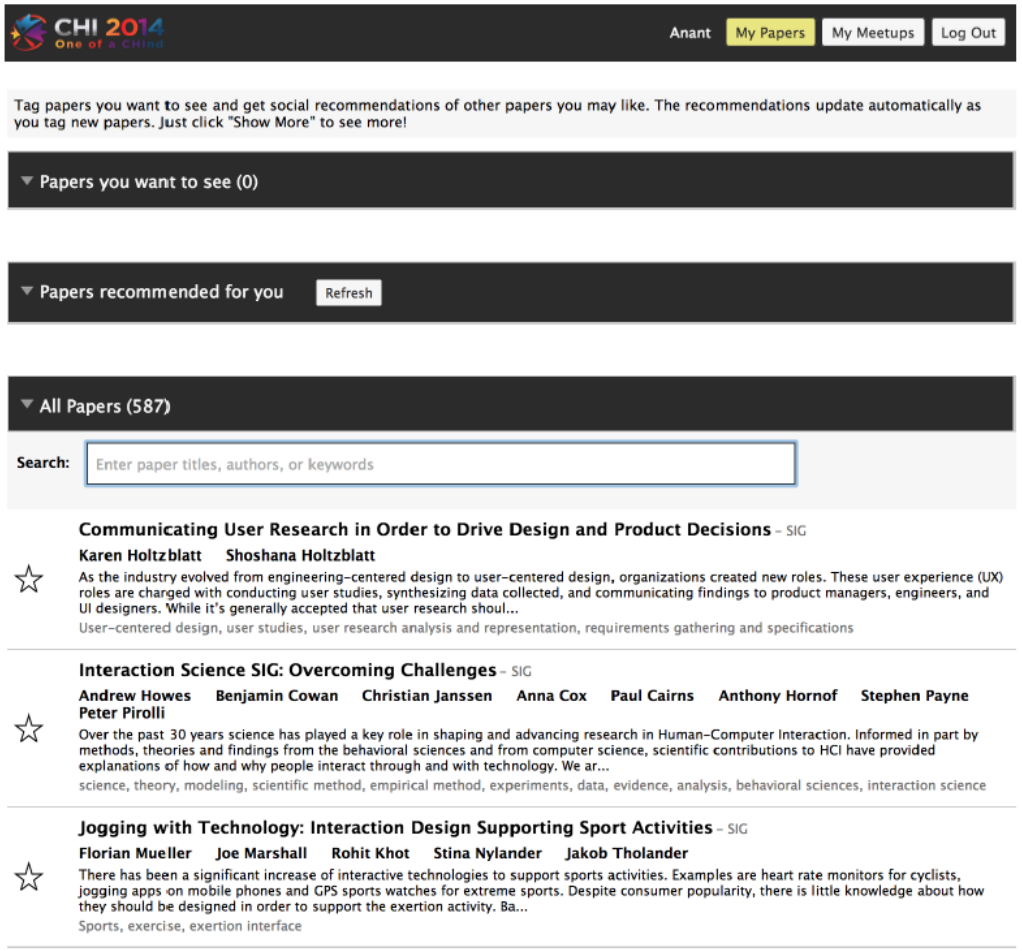


FIGURE 1.2: *Confer's* interface for CHI2014. (adapted from Bhardwaj et al. (2014))

Some additional software tools are discussed in this paragraph. *Confex* (Exchange, 1996) is a commercial software that offers conference planning tools for conference management including tools for scheduling. Organisers use its interface to assign papers into sessions and schedule sessions into time slots and rooms. Presenters' conflicts and double-bookings are automatically detected by the system and are reported to the user.

In addition, the organisers may set room capacities and requirements to receive warnings when sessions are scheduled in inappropriate rooms. *Conference Navigator 2.0* (Wongchokprasitti et al., 2010) and *Conferator* (Macek et al., 2012) are software tools similar to *Confer*. Attendees use these software to mark papers they are interested in and receive personalised paper recommendations to create their own personal schedules. However, in contrast to *Confer*, these software tools have not considered the exploitation of their data which could be used to resolve attendees' conflicts during the scheduling process.

1.2 Research Gaps

The CoSP literature is presented in Table 1.2, where each study focuses on specific conferences, addressing only conference specific requirements. The Table 1.2 shows that there is no study up to date that describes a generic conference scheduler. In addition, the COVID-19 pandemic has resulted in many conferences switching to online and hybrid conferences. Consequently, conference organisers are now encountered with completely new parameters to consider during the scheduling process (e.g., time zone differences). With this in mind, the following research questions are addressed in this thesis;

1. What a generic conference scheduler would look like in terms of objectives, constraints and modelling?
2. What Operations Research techniques are efficient for optimal (or near-optimal) scheduling of conferences, including online and hybrid conferences?
3. Given that there is no open-source code base for conference scheduling available, could this research result in an open-sourced solution for conference scheduling?
4. What is the potential practical impact of this research?

A clarification of the conference terminology as used in this thesis is now presented due to the diverse conference terminology that has been used in the CoSP literature. While various terms such as paper, presentation, talk, discussion, and panel are used in the literature, we use the term "submission" to refer to a formal event that requires scheduling at a conference. The term "track" is used to refer to a group of submissions with similar subject, whereas terms such as stream, subject area, and topic are used in the literature. We use the term "time slot" to refer to a fixed predefined amount of

TABLE 1.2: Requirements of conferences considered in the literature

Requirement	Eglese and Rand (1987)	Sampson and Weiss (1995)	Thompson (2002)	Potthoff and Munger (2003)	Sampson (2004)	Potthoff and Brams (2007)	Nicholls (2007)	Zulkipli et al. (2013)	Edis and Edis (2013)	Quesnelle and Steffy (2015)	Stidsen et al. (2018)	Vangerven et al. (2018)	Manda et al. (2019)	Patro et al. (2022)	Riquelme et al. (2022)	Rezaeinia et al. (2024)
Speakers' conflicts			✓	✓	✓	✓	✓		✓	✓			✓		✓	
Speakers' preferences						✓	✓		✓	✓		✓	✓		✓	
Rooms preferences	✓									✓						
Attendees' conflicts	✓	✓	✓	✓	✓	✓	✓			✓		✓				✓
Rooms capacities		✓	✓		✓		✓	✓		✓	✓				✓	
Similar tracks													✓			
Parallel tracks				✓		✓			✓		✓				✓	
Session hopping											✓	✓				
Track chairs' conflicts	✓	✓		✓		✓	✓								✓	
Tracks' scheduling preferences		✓							✓							
Rooms unavailability	✓		✓							✓						
Consecutive tracks											✓					
Speakers' time zones																

time available for presentation, and the term "session" is used to refer to a certain time period of the conference that consists of a number of time slots.

In general, a CoSP requires the scheduling of tracks into sessions and rooms to create a high-level schedule, and the scheduling of submissions into sessions, rooms and time slots to obtain a low-level schedule considering multiple soft and hard constraints. While some studies in the literature generate both high and low level schedules, others only generate a high level schedule leaving the scheduling of the low-level schedule to the organisers. Due to many conferences having different constraints and objectives, there are various problem descriptions, objective functions, and developed methods in the literature which depend on the need of the particular conference. As a result, different mathematical models and heuristic methods have been designed to address rather specific requirements of the conferences being studied per se. Consequently, a method that works well for a conference could be unsuitable for another conference.

The main goal of this thesis is the investigation of different operations research tools for the creation of a generic conference scheduler applicable to many conferences. This

scheduler is freely available at `https://github.com/ahmedkheiri/CoSPLib` and can be used to generate both high and low level optimised conference schedules in an autonomous and fully automated manner. A generic solution approach has been designed to allow the customisation of our scheduler to fit the needs of different conferences. A spreadsheet file is used to store input data, which follows a specific template with the purpose of providing a generic approach suitable for many conference scheduling problems. Our scheduler contains a pool of constraints to select from and allows weight assignment for each constraint based on their subjective significance. In addition, the scheduler is also suitable for hybrid and online conferences where submissions need to be scheduled in appropriate sessions considering timezone information. When a CoSP is solved using the scheduler, an informative solution file is generated which provides insights regarding the solution quality. The decision maker 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.

1.3 Dataset

The dataset used throughout this thesis consists of sixteen instances from four conferences, namely the Genetic and Evolutionary Computation Conference (GECCO), the OR Society's 60th Annual Conference (OR60), the New to OR Conference (N2OR), and the International Symposium on Forecasting (ISF). Their characteristics are presented in Table 1.3, while Table 1.4 shows the instances used per chapter. Note that OR60F, OR60F2 and OR60F3 are artificially generated instances derived from OR60. Apart from those, the remaining instances are all real data. All instances are available at `https://github.com/ahmedkheiri/CoSPLib`.

1.4 Thesis Structure

This thesis follows a paper-based thesis structure and there may be some overlap of material between chapters. The remainder of the thesis is structured as follows:

In chapter 2, the methodology chapter is presented, which provides definitions of technical terms and explores the theoretical foundations of each methodology used in this thesis, along with their relevance and applicability to the research.

TABLE 1.3: Instances considered in this thesis

Instance	Submissions	Tracks	Sessions	Rooms	Timeslots
GECCO2019	202	29	13	10	45
GECCO2020	158	24	7	8	28
GECCO2020 Poster	131	1	2	1	132
GECCO2020 Workshop	131	26	8	10	40
GECCO2021	138	27	6	8	24
GECCO2021 Workshop	203	28	8	10	56
GECCO2022	179	39	7	8	56
GECCO2022 Workshop	138	59	8	10	80
GECCO2023	207	26	6	9	60
GECCO2023 Workshop	233	55	8	9	80
ISF2022	311	49	11	10	36
N2OR	35	8	4	4	9
OR60	329	45	8	23	24
OR60F	279	45	8	23	24
OR60F2	556	72	16	23	49
OR60F3	1112	72	32	23	105

In chapter 3, we present a penalty system that allows organisers to set up scheduling preferences for tracks and submissions regarding sessions and rooms, and regarding the utilisation of rooms within sessions. In addition, we also consider hybrid and online conferences where submissions need to be scheduled in appropriate sessions based on timezone information. A generic scheduling tool is presented that schedules tracks into sessions and rooms, and submissions into sessions by minimising the penalties subject to certain hard constraints. Two integer programming models are presented: an exact model and an extended model. Both models were tested on five real instances and on two artificial instances which required the scheduling of several hundreds of time slots. The results showed that the exact model achieved optimal solutions for all instances except for one instance which resulted in 0.001% optimality gap, and the extended model handles more complex and additional constraints for some instances. This chapter demonstrates the suitability of the proposed generic approach to optimise schedules for in-person, hybrid, and online conferences.

Next, in chapter 4, we present a decomposed robust matheuristic solution approach that consists of two phases. In phase one, we use an integer programming model to build the high-level schedule by assigning tracks into sessions and rooms. Based on this solution, we create the low-level schedule where submissions are allocated into sessions, rooms, and time slots. In phase two, we make use of a selection perturbative hyper-heuristic

TABLE 1.4: Instances considered per chapter

Instance	Chapter 3	Chapter 4	Chapter 5	Chapter 6
GECCO2019	✓	✓	✓	✓
GECCO2020	✓	✓	✓	✓
GECCO2020 Poster			✓	✓
GECCO 2020 Workshop			✓	✓
GECCO2021	✓	✓	✓	✓
GECCO2021 Workshop			✓	✓
GECCO2022			✓	✓
GECCO2022 Workshop			✓	✓
GECCO2023			✓	✓
GECCO2023 Workshop			✓	✓
ISF2022		✓	✓	✓
N2OR	✓	✓	✓	✓
OR60	✓	✓	✓	✓
OR60F	✓	✓	✓	✓
OR60F2	✓	✓	✓	✓
OR60F3	✓	✓	✓	✓

to further optimise both levels of the schedule. Our solution approach is compared against an integrated mathematical model under different time limits on a set of real and artificial instances. The results showed that the matheuristic finds near-optimal solutions and finds solutions for instances where the mathematical model fails to provide solutions within the one hour time limit. This chapter demonstrates the suitability of the proposed matheuristic in tackling conference scheduling problems as an alternative approach to mathematical models that struggle for some instances.

The mathematical model in chapter 4 has two major differences compared to the models presented in chapter 3. That is, the model in chapter 4 is only concerned with the high-level schedule and it has some of its constraints relaxed. Essentially, a direct comparison is not possible because they have different purposes. While, the models in chapter 3 solve the conference scheduling problem as a whole, the model in chapter 4 only provides a high-level schedule solution for the conference scheduling problem. However,

some similarities and differences are observed which are described next. Both models have some constraints in common such as Equation 4.1 and Equation 3.5, Equation 4.2 and Equation 3.15, Equation 4.7 and Equation 3.4, Equation 4.8 and Equation 3.6, Equation 4.9 and Equation 3.1. The following constraints are relaxed in the model of chapter 4: i) scheduling the same track in parallel (Equation 4.5), ii) limiting number of rooms per track (Equation 4.6 compared to Equation 3.3) and iii) scheduling similar tracks in parallel (Equation 4.4 compared to Equation 3.13). The following constraints of the model in chapter 4 are unique: Equation 4.3, Equation 4.4 and Equation 4.5. Lastly, the objective functions of the models are substantially different. The objective functions of the models in chapter 3 are minimising violations of constraints related to both submissions and tracks, whereas the objective function of the model in chapter 4 only minimises violations of track related constraints.

In chapter 5, we make benchmark data publicly available to facilitate the comparison and evaluation of different developed methods for conference scheduling problems. We also present a selection hyper-heuristic algorithm to solve the benchmark instances and provide computational results. The aim is to raise awareness of the under-studied conference scheduling problem, and to encourage researchers to contribute to the benchmark dataset with new instances, constraints, and solving methods.

In chapter 6, we present the required modifications in the formulations presented in chapter 3 to obtain the equivalent mathematical models with time slots along with computational results. We also present an approximation model with a simpler, relaxed objective function which is obtained through transformations and discuss its performance compared to the exact model. Additionally, we compare the performance of all developed methods of this thesis by solving all available instances, and discuss the benefits and limitations of each method.

Lastly, in chapter 7, the contributions of this thesis are summarised, and potential directions for future work are suggested.

Chapter 2

Methodology

In this chapter, the background of the considered optimisation methods is presented. Different operations research tools such as integer programming, heuristics, and matheuristics were investigated to build the conference scheduler. All developed optimisation methods are included in the conference scheduler allowing the decision-maker to select which one they wish to use as some methods may perform better than others depending on the given CoSP. In the next paragraph, definitions are provided for some terms that are used throughout this thesis.

Any approach to problem solving that employs a pragmatic method which is not fully optimised, perfected, or rationalised is considered a *heuristic*. A *metaheuristic* could be either a high-level procedure or heuristic that aims to find, generate, tune, or select a heuristic which may return a sufficiently good solution to an optimisation problem. A *hyper-heuristic* is a heuristic that aims to automate the selection, combination, generation and adaptation of simpler heuristics processes to solve computational search problems in an efficient way. *Exact algorithms* are algorithms which always provide the optimal solution to an optimisation problem. An *approximation algorithm* is any algorithm that can find approximate solutions to optimisation problems. *Model complexity* defines how difficult it is to solve a mathematical model. A *hard constraint* is a very restrictive constraint that must be satisfied, whereas a *soft constraint* is a constraint that is allowed to be violated. The term *OR tools* refers to problem-solving techniques and methods that fall within the Operations Research field of study. In this thesis, the usage of the term *exact model* refers to an exact algorithm, while the usage of the term *extended model* refers again to an exact algorithm, but it is called extended because it includes additional constraints compared to the exact model.

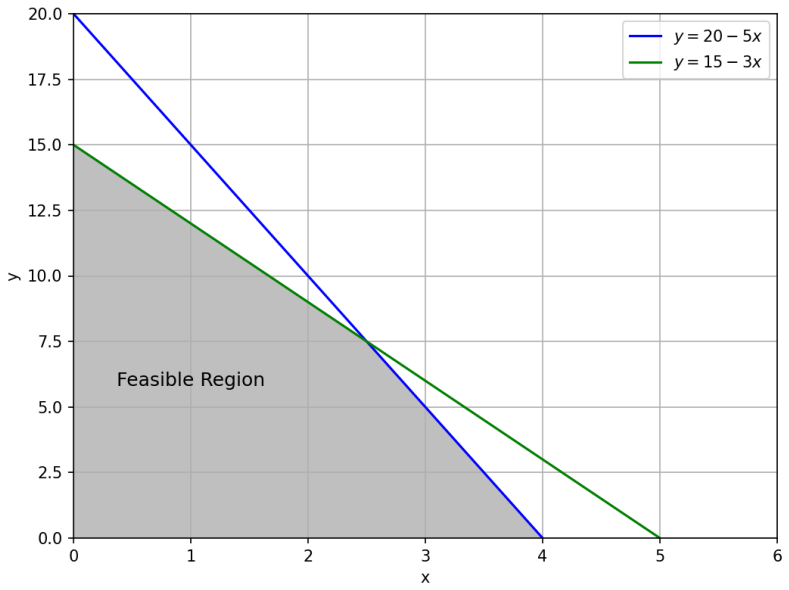


FIGURE 2.1: Feasible Region of a Linear Programming Problem

2.1 Integer Programming

Prior to describing the concept of integer programming, we first need to introduce the classical linear programming. Linear programming is defined as an optimisation method used to obtain the best solution in a mathematical model whose formulation is expressed in linear relationships. In other words, linear programming is an optimisation technique for minimisation or maximisation of a linear objective function, subject to linear equality and inequality constraints. Its feasible region is given by the set of all possible points of an optimisation problem that satisfy the constraints of the problem (e.g., see Figure 2.1) (Williams, 2009; Vanderbei, 1998). A formulation example of a linear programming problem is as follows:

$$\begin{aligned} \min \ & c^\top x \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where x are continuous variables, c and b are given vectors, and A is a given matrix. In this example, the goal is to minimise the objective function, $c^T x$, with respect to the constraints, $Ax \leq b$ and $x \geq 0$.

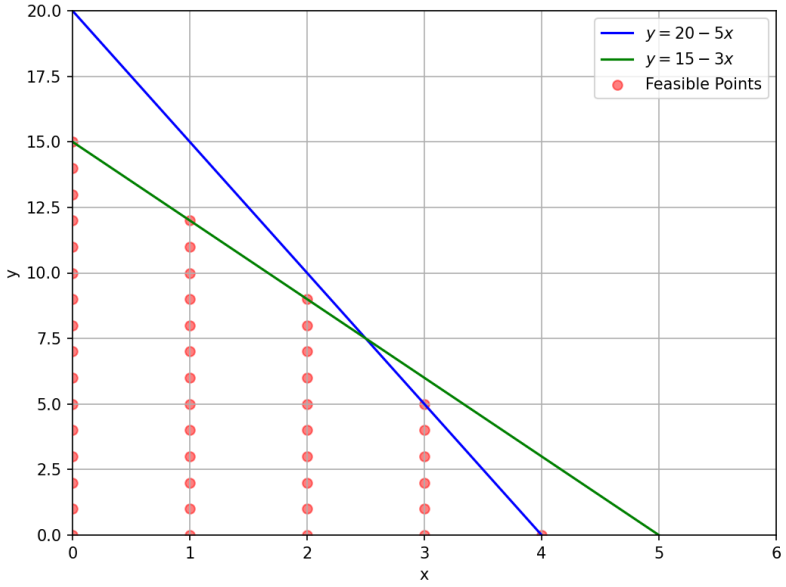


FIGURE 2.2: Feasible Integer Points of an Integer Programming Problem

Integer programming is usually considered as an extension of linear programming where some or all of its variables are restricted to take integer values. A model is defined as pure integer programming when all variables must be integers, whereas a mixed integer programming is a model where only some of the variables are restricted to integer variables. In contrast to linear programming problems, integer programming problems have feasible integer points instead of a feasible region (see Figure 2.2). It should be also noted that the computational complexity of integer programming models is greater than linear programming models because of integer variables and, thus, much more difficult to solve. Due to this increased complexity, it is usually impractical to solve large scale real-world problems exactly with integer programming. Therefore, heuristics are often preferred as an alternative optimisation method which we describe in more detail in section 2.2 (Williams, 2009; Conforti et al., 2014; Papadimitriou, 1981; Garey et al., 1990). The formulation of an integer programming problem is similar to the linear programming problem. For instance, we can rewrite the previous formulation example

to a pure integer programming problem as follows:

$$\begin{aligned} \min\ & c^\top x \\ & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z} \end{aligned}$$

where the only difference, comparing to the previous example, is that x variables are restricted to integer values.

A very practical and useful variant of integer programming is the zero-one linear programming (or binary integer programming) that is used for problems where all variables must be binary (either 0 or 1). The 0-1 variables are widely used to formulate problems in which decisions are discrete by nature (Yes/No decisions), and to facilitate the formulation of logical decisions and statements, e.g., if an ingredient i is used then a proportion of a_i must be respected. Moreover, 0-1 variables can be used to convert non-linear terms into linear such as the product of two 0-1 variables. For instance, assume we want to replace the product $\alpha_1\alpha_2$ with a new 0-1 variable β. We can achieve this by introducing the following constraints:

$$\begin{aligned} \beta &\geq \alpha_1 + \alpha_2 - 1 \\ \beta &\leq \alpha_1 \\ \beta &\leq \alpha_2 \end{aligned}$$

These constraints enforce $\beta = 1$ only when $\alpha_1 = \alpha_2 = 1$, otherwise β is restricted to 0 if α_1, or α_2, or both are 0. Another similar use of 0-1 variables is the approximation of non-linear models as shown in Figure 2.3. An integer programming model can be used to convert a certain type of non-linearity into linear terms at a cost in the size of the converted model though (Williams, 2009). Due to these practical properties of the zero-one variables, binary integer programming models have been extensively used in numerous applications and problems such as scheduling (Floudas and Lin, 2005; Ryan and Foster, 1981), the travelling salesman problem (Miller et al., 1960; Montero et al., 2017), the knapsack problem (Billionnet and Soutif, 2004; Lodi and Monaci, 2003), packing and partitioning problems (Litvinchev et al., 2015; Park et al., 1996; Niemann and Marwedel, 1997), the facility location problem (Canel and Khumawala, 1996; Kratica et al., 2014), and many others ranging from networks and graphs to capital investment problems (Veremyev et al., 2014; Meier et al., 2001; Mansini et al., 2015).

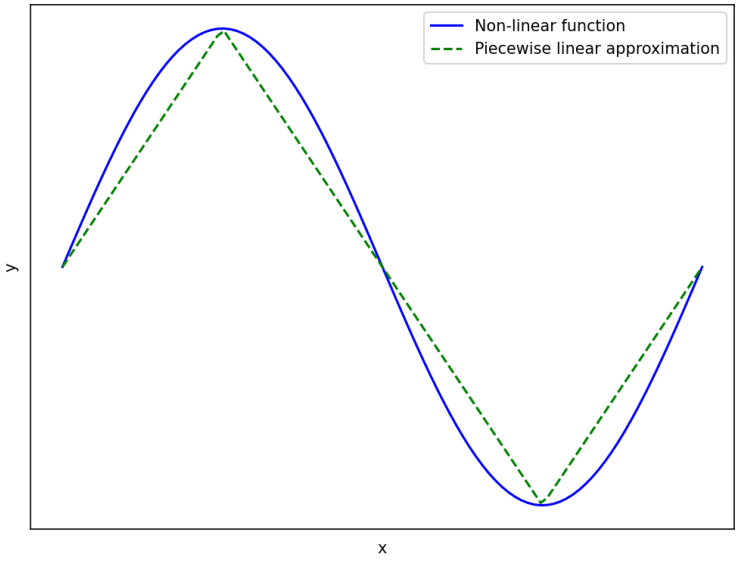


FIGURE 2.3: A piecewise linear approximation to a non-linear function

2.1.1 Relaxations

Relaxation is a well-known modelling strategy within the field of mathematical optimisation to approximate difficult problems by similar problems that are usually easier to solve. The solution of a relaxed problem provides information about the original problem. For instance, removing integrality constraints from an integer programming problem results in a relaxed linear programming problem.

Another relaxation method is the Lagrangian relaxation which allows inequality constraints to be violated. Relaxed constraints are moved to the objective function and are multiplied with a Lagrange multiplier. As a result, a cost is incurred for each violation of the relaxed constraints. For example, assume we are given the following linear programming model:

$$\begin{aligned} \min\ & c^\top x \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

To obtain the Lagrangian relaxation of the problem, constraint $Ax \leq b$ is moved to the objective function and multiplied with a Lagrange multiplier $\lambda^\top$ as follows:

$$\begin{aligned} \min\ & c^\top x + \lambda^\top (Ax - b) \\ & x \geq 0 \end{aligned}$$

where $\lambda^\top \geq 0$. Thus, an approximation of the original problem is achieved in which the objective function is penalised if the relaxed constraint gets violated (Lemaréchal, 2001).

A similar relaxation method to Lagrangian relaxation is the one introduced in Kendall (1975). However, the difference is that in his method the relaxed constraints are not moved to the objective function. Instead, new variables are introduced in both the objective function and the constraints. These variables are used to represent the exceeded violation amount of a constraint and the unused amount of a constraint. Kendall (1975) described this method as a mixture of a conventional objective function and a goal programming objective function which can be formulated as follows:

$$\begin{aligned} \min\ & c^\top x + c'^\top u + c''^\top v \\ & Ax + u - v = b \\ & x \geq 0 \end{aligned}$$

where u indicates the unused amount of the constraint, v indicates the exceeded amount of the constraint, $c'^\top$ represents the cost of not fully utilising the constraint, and $c''^\top$ represents the cost of exceeding the constraint.

Goal programming can be defined as an extension or generalisation of linear programming and it is used in multi-objective optimisation to handle multiple, normally conflicting objectives. It is an approach that transforms a multi-objective formulation to a single-objective formulation. For each objective, a goal is set and the objective is to minimise deviations from the set goals or, in other words, to minimise the violations of constraints. Even though the approach in Kendall (1975) seems similar to goal programming, it is not exactly the same, because in his approach there is no setting of goal values and, thus, no deviations to minimise.

In this paragraph, additional relaxation methods are presented such as the penalty

method, cutting planes, dual relaxation, convexification, surrogate relaxation and semidefinite relaxation. In the penalty method, the original problem is relaxed by moving constraints into the objective function resulting in an unconstrained optimisation problem. The penalty terms penalise the violations of constraints by increasing or decreasing the objective value, thus, encouraging the solver to stay within the feasible region (Di Pillo, 1994). The cutting plane method adds linear constraints to the feasible region of the optimisation problem with the purpose of removing fractional solutions, which results in a tighter relaxation and bound improvement. It is particularly useful in integer programming and mixed-integer programming optimisation problems in which solutions have integer values (Schrijver et al., 1980). Dual relaxation relaxes some constraints of the primal problem and solves the corresponding dual problem which is sometimes easier to solve (Chernyavsky et al., 2023). The goal of the convexification method is to transform a non-convex optimisation problem into a convex one through convex optimisation methods such as convex envelopes, convex hulls, concave-convex procedure and convex surrogates (Bertsekas, 1979). Surrogate relaxation approximates the original objective function or constraints with a simpler surrogate function or constraint set. It is easier to optimise the simpler function compared to the original one. The optimal solution to the surrogate can be used as a good approximation for the original problem or improved further. There are several ways to create surrogate functions depending on the structure of the optimisation problem (Narciso and Lorena, 1999). Semidefinite relaxation relaxes a non-convex problem into a semidefinite programming problem by leveraging the properties of semidefinite matrices. A semidefinite programming problem is a type of convex optimisation problem that is easier to solve (Luo et al., 2010).

Apart from approximating difficult problems, relaxation methods have several additional benefits and uses which are described next. They can be used to obtain bounds in branch-and-bound algorithms for integer programming (e.g., (Jessin et al., 2020)) and to overcome infeasibility issues in mathematical modelling. Additionally, they enable the handling of complex and soft constraints, and allow the expansion of the solution space by setting different weights values. That is, they allow the exploration of multiple solutions along with trade-offs which is beneficial for decision-makers as they are presented with more options. However, relaxation methods have certain drawbacks as well. They require the judgement of decision-makers upon selecting the right solution and the relaxed problem may not converge to the optimal solution of the original problem.

2.2 Hyper-heuristics

Sometimes, it is impractical to solve a problem exactly to obtain the optimal solution for various reasons such as computational complexity, size of the problem, and complexities involved in formulating the problem. In such cases, an effective alternative method is the implementation of heuristic techniques which sacrifice optimality to find near-optimal solutions within a short amount of time.

A well-known heuristic method to solve computationally hard optimisation problems of combinatorial nature is the local search algorithm, or also known as neighbourhood search. Typically, a combinatorial optimisation problem involves an objective function to minimise, or maximise, and a set of feasible, or candidate, solutions. The key mechanism of the local search algorithm is based on a neighbourhood structure, where a neighbourhood is defined as a set of solutions that are slightly different compared to a given solution. Local search algorithms apply local changes to explore neighbourhood solutions within the space of candidate solutions, or the search space, until a termination criterion is met (Johnson et al., 1988; Hansen and Mladenović, 2001). For instance, in a travelling salesman problem, a neighbourhood solution is a tour in which the visiting order of two cities is different. A significant part of heuristic methods is the criterion used to accept or reject a new solution, which is known as the move acceptance criterion. A common move acceptance criterion is the acceptance of the new solution only when it is better than the previous solution (Pirlot, 1996). While local search algorithms are effective in terms of execution time and solution quality, they do have a pitfall. That is, they can get stuck at a local optimum when the neighbourhood search space is poorly structured (e.g., see Figure 2.4). Several methods exist to overcome this pitfall such as the usage of a random restart approach in which the local search algorithm is applied to multiple random initial solutions. Hence, this approach increases significantly the search space and the probability of finding a better local optimum (Lourenço et al., 2002). Another method is the implementation of the ruin and recreate principle where parts of the solution are destroyed and rebuilt or recreated afterwards (Schrimpf et al., 2000). Alternatively, more sophisticated algorithms may be used instead such as simulated annealing, genetic algorithms, ant colony optimisation, or hyper-heuristics which are described next.

Many combinatorial optimisation problems have been solved by means of hyper-heuristics which can be simply defined as a heuristic to choose heuristics. While hyper-heuristics were conceptualised in 1960s, they were formally defined in the early 2000s as a high-level

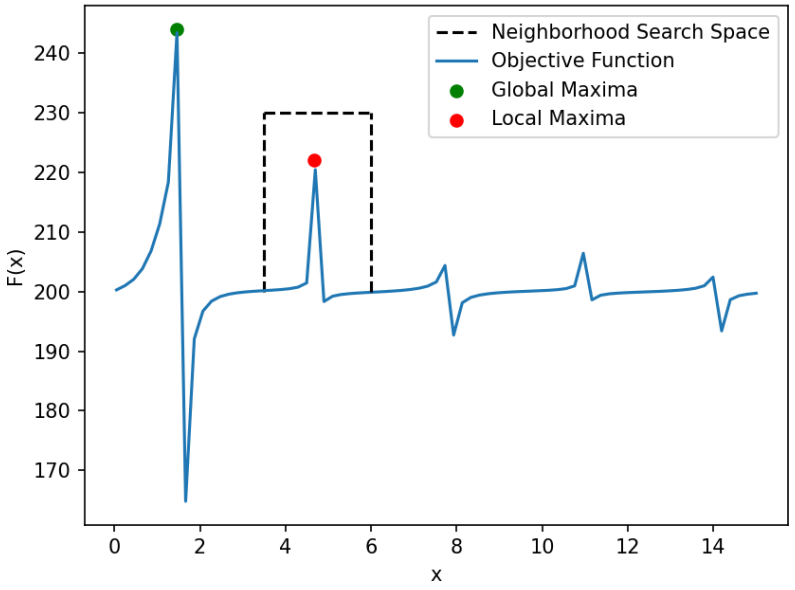


FIGURE 2.4: Local search algorithm stuck at local maxima

automated search methodology for solving computationally hard problems by exploring the search space of low-level heuristics. Hyper-heuristics are classified into the following two categories: *selection* and *generation* hyper-heuristics, which are further divided into: *constructive* and *perturbative* hyper-heuristics.

Selection constructive hyper-heuristics usually explore the space of combinations of low-level constructive heuristics to create an initial solution. The following are some of the common low-level constructive heuristics in scheduling problems used to choose how events are scheduled:

- Largest degree represents the number of potential conflicts that a given event has.
- Largest colour degree, a variation of the largest degree, considers conflicts with scheduled events only.
- Saturation degree indicates the number of feasible time periods in which an event can be scheduled.
- Largest weighted degree represents the number of individuals that could create a potential conflict for a given event.
- Largest enrolment indicates the number of individuals who are interested in (or enrolled for) a given event.

- Random chooses a random event to schedule.

For instance, *s-le-r-lc* is a candidate combination of low-level constructive heuristics, where *s* is the saturation degree, *le* is the largest enrolment, *r* is random and *lc* is the largest colour degree. Selection perturbative hyper-heuristics select and apply a heuristic from a set of low-level heuristics to improve a given solution. They are distinguished into multi-point selection perturbative hyper-heuristics, which are population-based without a distinct heuristic selection or move acceptance strategy, and single-point selection perturbative hyper-heuristics that improve one active current solution and consist of a specific heuristic selection and move acceptance strategy.

Generation constructive hyper-heuristics create new low-level constructive heuristics which can be disposable or reusable. A low-level constructive heuristic that is created for a specific problem instance is defined as disposable, whereas a reusable low-level constructive heuristic is created once and can be used to solve multiple instances of the same problem. Generation perturbative hyper-heuristics have not been as much explored as the aforementioned types of hyper-heuristics. Sabar et al. (2013) made a contribution in this area by presenting the grammatical evolution hyper-heuristic framework. This framework takes as inputs different heuristic selection and move acceptance strategies and automatically creates a local search template by appropriately combining the provided strategies to suit a given problem instance and discover new heuristics (Drake et al., 2020; Pillay, 2016).

Selection constructive hyper-heuristics are mainly used to construct an initial solution or rebuild parts of a solution rather than optimising a given solution, while generation hyper-heuristics are not as widely used as selection perturbative hyper-heuristics in solving scheduling problems. Therefore, selection perturbative hyper-heuristics seem to be the most promising candidate among the different types of hyper-heuristics in providing a complete solution to the CoSP. For the remainder of the thesis, we will refer to the selection perturbative hyper-heuristic simply as hyper-heuristic. A typical framework of a hyper-heuristic usually includes two sequential steps, the heuristic selection and the move acceptance (see Figure 2.5). While the former step represents a strategy for selecting and applying a low-level heuristic, the latter step represents a strategy regarding the acceptance or rejection of the newly created solution (Özcan et al., 2008). More details about heuristic selection and move acceptance are presented in subsection 2.2.1 and subsection 2.2.2.

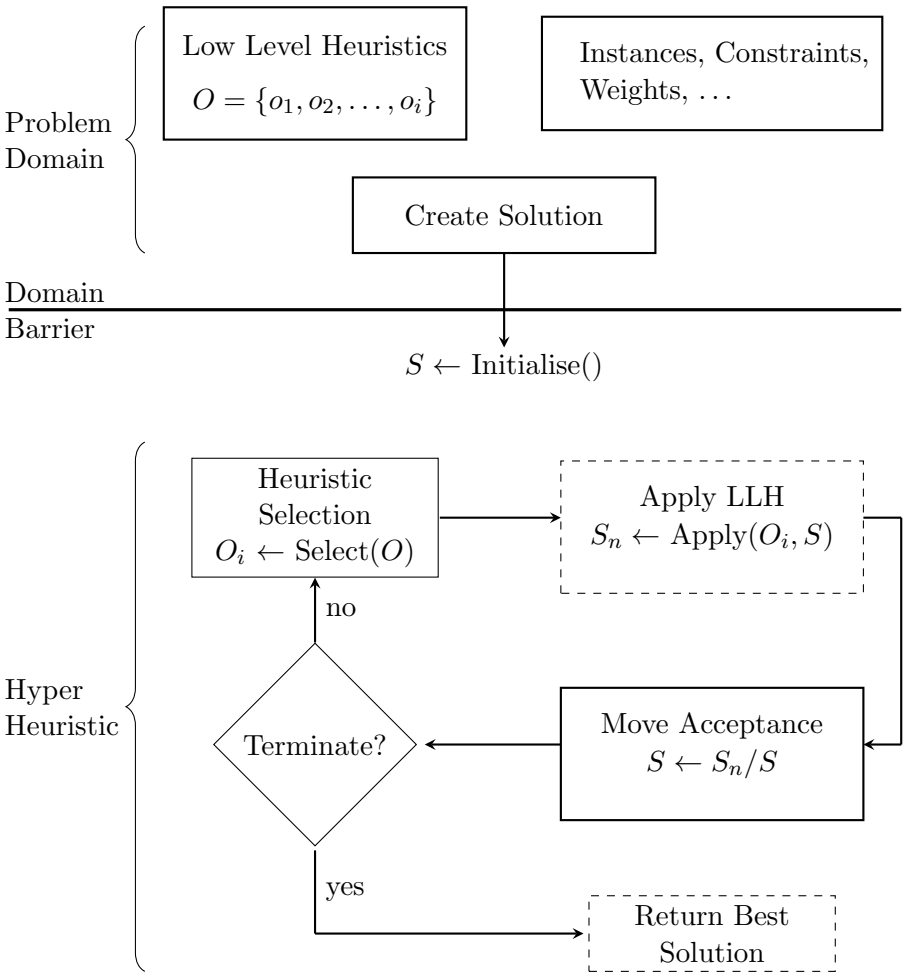


FIGURE 2.5: A flowchart of the hyper-heuristic framework

Hyper-heuristics have become popular due to their advantages over other customised methods. Their major benefit is that they are not limited to a single problem domain or a narrow class of problem instances, instead they are problem independent and are applicable to a wide range of problem instances (Almutairi et al., 2016; Özcan et al., 2008; Drake et al., 2020). Additionally, hyper-heuristics are capable of learning by receiving feedback regarding the performance of low-level heuristics during the optimisation process. Two types of learning processes are identified in the literature: the *online* learning in which the algorithm directly learns while solving the problem, and the *offline* learning in which the algorithm learns by solving a set of training instances. Some recent studies, though, have incorporated a mixture of online and offline learning (Burke et al., 2013; Drake et al., 2020). Furthermore, it is relatively easy to implement hyper-heuristics, compared to other customised methods, as no information is required about the functionality of low-level heuristics. They only require the number or content of the low-level

heuristics and the type of optimisation (minimisation or maximisation) (Chakhlevitch and Cowling, 2008; Kendall and Mohamad, 2004). Last but not least, their purpose is to find an effective method of solving a problem by exploring the space of low-level heuristics. This is in contrast to other metaheuristic algorithms which are limited to finding a good solution by exploring directly the space of solutions (Drake et al., 2012a; Chakhlevitch and Cowling, 2008).

2.2.1 Heuristic Selection Strategies

In this step of the optimisation process, a low-level heuristic is selected from a set of low-level heuristics and is applied to the solution iteratively. Note that a low-level heuristic can be a simple operator, a metaheuristic, or even a hyper-heuristic. The performance of the algorithm depends heavily on the selected strategy as some low-level heuristics may perform better than others at certain stages of the optimisation process. For instance, low-level heuristics that perform major changes to the solution are usually more successful at early stages of optimisation rather than later stages. In addition, the sequence in which low-level heuristics are applied plays a crucial role in terms of performance too, because different solutions are obtained with altered sequences (Chakhlevitch and Cowling, 2008). Heuristic selection strategies are defined as *deterministic* when a selection sequence is predefined, or *stochastic* when the selection depends on a probability distribution or learning mechanism (Drake et al., 2020; Özcan et al., 2008). Some examples of heuristic selection strategies are presented next:

- Simple random: A low-level heuristic is randomly selected.
- Greedy: All low-level heuristics are applied and the best among them is selected at each iteration.
- Reinforcement learning: Each low-level heuristic is assigned an initial minimum score which increases or decreases based on performance. At each iteration, the low-level heuristic with the highest score is selected (Chakhlevitch and Cowling, 2008).
- Tabu search: Low-level heuristics are ranked similarly to reinforcement learning strategy. Upon solution improvement the rank is increased, otherwise the rank is decreased and the low-level heuristic enters the tabu list until the solution at hand changes. This strategy selects the low-level heuristic with the highest rank that is not included in the tabu list (Drake et al., 2012a).

- Sequence-based selection: In this strategy, low-level heuristics are selected based on the sequences of low-level heuristics with the best performance (Almutairi et al., 2016).

2.2.2 Move Acceptance Strategies

At this stage, the newly obtained solution is evaluated and is either accepted or rejected based on the selected strategy. Similarly to heuristic selection strategies, move acceptance strategies can be defined as *deterministic* or *stochastic*. Deterministic strategies usually accept non-worsening solutions, while stochastic strategies may accept or reject a solution according to some probabilistic criterion. Stochastic strategies may accept worse solutions during optimisation which is particularly useful for escaping local optimum solutions and thus diversifying the search space (Drake et al., 2020; Özcan et al., 2008). A few examples of move acceptance strategies are the following:

- Only improving: The solution is only accepted if it is better than the previous.
- Improving and equal: All non-worsening solutions are accepted.
- Simulated annealing: Solutions are accepted based on the following probabilistic function:

$$p_t = e^{-\frac{\Delta f}{\Delta F(1-\frac{t}{T})}}$$

  where Δf is the difference in the objective value at iteration t, T is the maximum number of iterations, and ΔF is the range of the maximum change in the objective value after a low-level heuristic is applied (Ahmed et al., 2015).

- Record-to-record travel: A worse solution is only accepted if it is not significantly exceeding the objective value of the previous solution (Dueck, 1993).
- Late acceptance: Objective values are stored in a set of size L, and a new solution is only accepted if it is better than the L^{th} solution (Burke and Bykov, 2008).
- Great Deluge: Accepts all moves within a dynamic level of the objective value. The initial level is equal to the initial objective value and is linearly updated towards the expected objective value by using the following formula:

$$\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T})$$

where τ_t is the threshold level at iteration t, T is the maximum number of iterations, ΔF is the expected range for the maximum change in the objective value, and f_0 is the final expected objective value. (Kendall and Hussin, 2004)

2.2.3 Hyper-heuristics for timetabling problems

Several timetabling problems such as university examination timetabling, school timetabling and university course timetabling have been successfully tackled by hyper-heuristics.

Ozcan et al. (2009) solved the examination timetabling problem by implementing a hyper-heuristic with a late acceptance strategy. The authors tested several heuristic selection strategies from which the simple random yielded the best results. Their hyper-heuristic utilised four low-level heuristics; a mutation heuristic responsible for rescheduling all exams and three heuristics for exams rescheduling to minimise violations of constraints. Kendall and Hussin (2004) developed a tabu search hyper-heuristic to solve the examination timetabling problem. The authors tested the tabu search with the improving and equal and the great deluge acceptance strategies. A large list of low-level heuristics was utilised which included five heuristics for rescheduling exams, two heuristics for swapping the periods of two exams, a heuristic to un-schedule an exam and five heuristics to reschedule unscheduled exams. Another examination timetabling problem was solved by Özcan et al. (2012) who developed a hyper-heuristic with a reinforcement learning heuristic selection strategy and a great deluge move acceptance strategy. The authors used different types of low-level heuristics: one type was responsible for rescheduling the exam that resulted in the most constraints' violations, another type rescheduled the exam that violated the most a particular constraint and the last type was used to reschedule randomly all the scheduled exams.

Kheiri et al. (2012) solved the school timetabling problem by developing a generalised hyper-heuristic, namely the HySST. Two hyper-heuristics were implemented, one was responsible for the intensification stage and the other for the diversification stage. The former hyper-heuristic used a simple random heuristic selection strategy to select a heuristic from a set of nine mutational low-level heuristics, whereas the latter hyper-heuristic used two hill-climbing heuristics. Raghavjee and Pillay (2012) solved the school timetabling problem with an evolutionary hyper-heuristic. A set of five low-level heuristics was used, four heuristics were mutational and one heuristic was used for allocation and de-allocation. The authors compared the hyper-heuristic with several algorithms,

namely neural networks, tabu and greedy search and a genetic algorithm. Their developed hyper-heuristic outperformed all the algorithms especially in large scale instances.

Soria-Alcaraz et al. (2014) developed an iterated local search hyper-heuristic to solve the university course timetabling problem. The hyper-heuristic used an online learning mechanism as a heuristic selection strategy to select the low-level heuristic with the best chances of improving the solution based on previous performance. Another university course timetabling problem was solved by Kalender et al. (2013) who implemented a hyper-heuristic that used a greedy gradient heuristic selection strategy and a simulated annealing move acceptance strategy. The same hyper-heuristic was also used to solve a school timetabling problem. Burke et al. (2003) tackled the university course timetabling problem by implementing a multi-objective hyper-heuristic. Tabu search was used as a heuristic selection strategy and each heuristic from the set of low-level heuristics optimised a particular objective.

2.3 Matheuristics

The term matheuristic, as the name suggests, refers to an optimisation method in which ideas and methods are combined from both mathematical programming and heuristic techniques. Puchinger and Raidl (2005) provided a general classification of matheuristics based on the nature of combination. They classified matheuristics into two main categories, namely the *collaborative combinations* and the *integrative combinations*. The former category refers to matheuristics in which exact and heuristic algorithms are separate parts and exchange information by being executed sequentially, intertwined or in parallel, while the latter category refers to matheuristics in which one technique is an embedded component of another technique. While Puchinger and Raidl (2005) touched on a generic classification of matheuristics, additional classifications are defined in Archetti and Speranza (2014) which are discussed next:

- *Decomposition approaches:* In this approach, the master problem is decomposed into smaller and simpler sub-problems, where each sub-problem is solved by a specific solution method. Mathematical programming models are then used to solve some or all sub-problems to optimality or sub-optimality.
- *Improvement heuristics:* A heuristic method is used to obtain a solution which is improved via mathematical programming models.

- *Branch-and-price/column generation-based approaches:* In this approach, the aim is to achieve convergence faster by modifying the exact method used.
- *Relaxation-based approaches:* The concept of this approach is to identify attributes of an exact method that significantly slow down the convergence and relax them.

From the above classifications, we will focus and elaborate on decomposition approaches and provide application examples. Decomposition approaches are specifically useful for complex and integrated problems. Dividing the master problem into sub-problems that are handled and solved independently reduces the complexity of the problem and makes it easier to obtain a feasible solution for the master problem. In the context of matheuristics, mathematical models are used to solve some or all of the sub-problems. In cases of large scale real-world problems a time limit can be applied to prematurely stop the exact method and, thus, avoid excessive computational times. A *two-phase approach* is a subclass of decomposition approaches in which the matheuristic algorithm divides the master problem into two phases that are solved separately (Archetti and Speranza, 2014). This approach is in line with collaborative combinations as described in Puchinger and Raidl (2005). A few examples of studies that have implemented a two-phase decomposition matheuristic approach are described next. Flisberg et al. (2009) studied a pickup and delivery problem involving daily routes assignment of logging trucks in forestry which was solved in two phases. In the first phase, an exact model was used to obtain the optimal flow from supply points to demand points, and transport nodes were created based on the obtained solution. Next, in phase two, they used a heuristic method to combine the transport nodes into actual routes. In the study of Yi and Özdamar (2007), a logistic problem arising in disaster response activities was decomposed into two parts that were solved separately. In the first part, the authors obtained location decisions and approximate vehicle routes by solving exactly a simplified version of the complete mathematical formulation. Then, in the second part, they implemented an algorithm that generated pick up and delivery schedule for each vehicle based on the information obtained from the previous phase, and built detailed vehicle routes by solving a system of linear equations. Lastly, Schittekat et al. (2013) studied the school bus routing problem in which the set of visiting stops needs to be determined, the stop where each student has to walk to needs to be defined, and the school bus travelling distance needs to be minimised. The problem was solved in two phases: in phase one, a heuristic method was used to solve the routing part of the problem, and in phase two, an exact model is used to assign students to stops. Further examples of

matheuristic applications including additional classifications can be found in the survey of Ball (2011).

Chapter 3

A Generic Approach to Conference Scheduling with Integer Programming

3.1 Introduction

Conferences are events of great importance to scientific communities. They provide an opportunity for academics and researchers to present their research work and receive feedback from the community, and to learn from other presenters. In addition, participants benefit from networking opportunities, exchanging ideas, and potential future collaborations. Hence, a conference schedule brings a significant opportunity and challenge in offering the best possible experience to every participant. Conference organisers usually struggle to create, or even characterise, the best schedule due to the large number of preferences and constraints involved. Some of the constraints that make conference scheduling an arduous task are requests from presenters to present at a specific time, resolving presenters' conflicts, handling capacity issues, to name a few. Additionally, the COVID-19 pandemic has resulted in many conferences switching to online or hybrid mode, which introduces further complexity due to different timezones involved.

Due to the diverse conference terminology that has been used in the *conference scheduling problem* (CSP) literature, we clarify the conference terminology as used in this paper, which we believe may be applicable to many conferences. While various terms such as paper, presentation, talk, discussion, and panel are used in the literature, we use the

term "submission" to refer to a formal event that requires scheduling at a conference. The term "track" is used to refer to a group of submissions with similar subject, whereas terms such as stream, subject area, and topic are used in the literature. We use the term "time slot" to refer to a fixed predefined amount of time available for presentation, and the term "session" is used to refer to a certain time period of the conference that consists of a number of time slots.

In this paper, we consider the CSP, which includes a set of tracks along with their corresponding submissions, a set of available sessions along with their corresponding time slots, and a set of available rooms. The objective is to achieve a schedule which is feasible to a number of (hard) constraints and minimises violations of a number of preferences (soft constraints) by assigning all tracks into sessions and rooms, and assigning all submissions into sessions. Based on the types of violations, a CSP can be approached from a Presenter-Based Perspective (PBP) or from an Attender-Based Perspective (ABP) (Thompson, 2002). A PBP approach aims to minimise violations associated with presenters' preferences, such as a request to present on a specific day or at a specific time (Sampson, 2004). An ABP approach minimises violations regarding attendants' preferences. Some examples are that all of the attendants wish to attend their favourite session, they do not want to miss a session due to space shortage, and they do not want to choose between two sessions of their interest due to sessions being scheduled concurrently (Zulkipli et al., 2013). Some studies have adopted a mixed approach by considering both presenters' and attendants' preferences (Nicholls, 2007; Vangerven et al., 2018).

The CSP was introduced by Eglese and Rand (1987) and was proved to be $\mathcal{NP}$-hard by Quesnelle and Steffy (2015) and Vangerven et al. (2018). Even though the problem was introduced several decades ago, it has not received much attention from researchers compared to related problems, such as Class and Exam Scheduling (Sampson, 2004). To the best of our knowledge, there are only 16 published studies tackling the CSP (see section 3.2). However, many conferences have different scheduling requirements, objectives, and constraints. As a result, a method that works well for a conference could be unsuitable for another conference. Thus, we believe that a generic framework for conference scheduling is needed. To this end, in this study we present a generic approach by considering both PBP and ABP to generate schedules for conferences in a fully automated manner.

Our work mainly differs from the existing literature in the sense that we generate both

low-level and high-level conference schedules. That is, we consider preferences and constraints associated with both tracks and submissions to create a complete schedule of a conference. In addition, our approach is suitable for hybrid and online conferences as we take timezones into account to avoid scheduling submissions into unsuitable times. To the best of our knowledge, this is the first paper to consider timezone differences in conference scheduling. Moreover, we present a generic approach for conference scheduling by applying the weighted sum method as described in Ehrgott (2005) to create an objective function to optimise. The method is applied to our penalty system designed to accommodate scheduling preferences which are weighted according to their relative importance. An easy-to-use and configurable spreadsheet template is used to meet the demands of different conferences. The template offers a single data format that is easy to adjust in order to fit different conference data. We acknowledge that our template is not suitable for all conferences, as some conferences may have very specific structures, but our aim is to accommodate as many as possible. Furthermore, in contrast to most previous studies which assume that all submissions require the same amount of time for scheduling (one time slot), we allow for submissions to have a different amount of required time slots, such as keynote talks.

In section 3.2, we present related work on conference scheduling problems, followed by section 3.3 which describes the conference scheduling problem as considered in this work. Then, in section 3.4, we present our exact model as a binary integer program with linear objective. Computational results are presented in section 3.5, followed by section 3.6 which presents the extended model with additional constraints along with computational results. Next, in section 3.7, we summarise our work and suggest potential future lines of research. We present our proposed spreadsheet template used for storing input parameters in Appendix C. In Appendix D we present an approximation model that includes a simpler, relaxed objective function of the exact model; we also present formulations of our models at a finer-level including time slots and illustrate that its approximation model makes this more complex model computationally feasible.

3.2 Related Work

A detailed survey on CSP can be found in Vangerven et al. (2018). Apart from those mentioned in Vangerven et al. (2018), we discuss the following studies which are related to our work.

A case study regarding the scheduling of 2001 San Antonio meetings of the Public Choice Society was presented in Potthoff and Munger (2003). The problem required the scheduling of submissions into sessions such that submissions of each track are evenly spread among sessions, and ensuring participants are not scheduled in more than one submission of the same session. The authors implemented an integer programming (IP) model to create the schedule of the conference which included 14 tracks and 96 submissions with 10 sessions available.

Potthoff and Brams (2007) extended the previous work by implementing their proposed IP method on both 2005 and 2006 annual meetings of Public Choice Society in New Orleans. The 2005 annual meeting required the assignment of 76 submissions from 13 tracks into 9 sessions, whereas the 2006 annual meeting included 45 submissions from 6 tracks with 9 sessions available. The model had the same objective and all the constraints as in the work of Potthoff and Munger (2003), and included an additional constraint in the IP formulation which considered the unavailability of some presenters to attend certain sessions. Both generated solutions had all constraints satisfied and successfully accomplished the objective.

Nicholls (2007) developed a simple heuristic algorithm to aid the scheduling process of the 2003 Western Decision Science Institute Annual Meeting. This conference involved 330 registered attendees, 295 submissions, 73 regular sessions including four time slots on average, 11 special sessions (a whole session is required for a submission), and 7 rooms of different sizes. The proposed heuristic was used to assist the Program Chair during the scheduling process rather than autonomously produce the conference schedule. The proposed heuristic algorithm did not have an objective function *per se*, but its main purpose was to resolve conflicts by utilising a set of rules and consider preferences from presenters and attendees.

Zulkipli et al. (2013) addressed the Capacity Planing problem variant of conference scheduling. The conference involved 3 tracks, 60 submissions, 5 sessions with 4 time slots each, and 3 rooms were available for parallel scheduling. They collected preferences from participants to create weights for each submission and used these to form the objective function of their goal programming model. The problem required the assignment of submissions into rooms and time slots in such a manner that each session achieves a balanced number of submissions with respect to the weights.

Edis and Edis (2013) described a case study in which they considered an artificial conference including 10 tracks and 170 submissions with a 3 days time span. Each day had

4 sessions with 5 time slots each, and 3 rooms were available for parallel scheduling. In their case study, the goal of the primary objective was to minimise the concurrent occurrence of same or similar tracks within the same session in each day. In addition, a secondary objective of the problem was to distribute the number of submissions into sessions in a balanced manner. The authors formulated an integer programming model, along with an extended version to address both objectives.

Another case study was conducted by Quesnelle and Steffy (2015) in which they used real data from the 2013 PenguiCon Conference. The conference was attended by around 1000 participants and involved 253 submissions, 195 presenters, and 14 rooms. In their study, the authors provided problem definitions and showed that the scheduling problem under study along with some variants are all $\mathcal{NP}$-hard. They specifically defined and focused on the *Extended Conference Timetable Decision Problem (ECTTD)* and the *Preference Conference Optimisation Problem (PCO)*. The objective of the ECTTD problem is the assignment of presenters into submissions and time slots, as well as the allocation of submissions into rooms based on their availability and compatibility. The PCO problem includes the objective of minimising participants' preferences conflicts by assigning presenters into submissions and time slots, as well as assigning submissions into appropriate rooms. Both ECTTD and PCO were solved with integer programming models.

The CSP of one of the largest conferences within the field of Operations Research, namely the EURO2016 Conference, was addressed in Stidsen et al. (2018). This particular conference included 25 areas of subject, 124 tracks, 1852 submissions, 11 sessions (for each typically 4 time slots were available), 5 buildings with 54 rooms in total, 1600 presenters, and attracted around 2000 participants. The problem required the allocation of subject areas into buildings, and the scheduling of tracks into sessions and rooms so as to comply with the hierarchical structure of the conference. The authors addressed the problem by implementing a multi-objective mixed integer programming (MIP) model which included 5 objectives ranked based on their significance and were sequentially solved following a lexicographic optimisation approach. These objectives were ranked in the following order: 1) Minimisation of the number of areas assigned to different buildings, 2) Maximisation of the number of related areas assigned to the same building, 3) Minimisation of the number of different rooms allocated for each track, 4) Minimisation of the number of time gaps within tracks and, 5) Maximisation of the residual room capacity. Furthermore, room capacity constraints were considered, and parallel scheduling of the same track was not allowed. However, Stidsen et al. (2018) commented

that author conflicts were circumvented due to the policy of the conference that allows only one submission per author, and the room utilisation was partially considered due to insufficient data. The authors clarified that the proposed model generates only the high-level schedule, leaving intentionally the low-level schedule to the track organisers. The success of the proposed model is reflected by the fact that it was also used to schedule the IFORS2017, EURO2018, and IFORS2020 conferences, and a slightly improved version of the model to schedule the subsequent EURO and IFORS conferences.

Although our models share some common scheduling requirements with the model of Stidsen et al. (2018), they differ significantly. Similarly to Stidsen et al. (2018), we have considered the fact that the same track must not run in parallel, we minimise the number of rooms utilised per track, we consider the scheduling of tracks consecutively, and we take room capacities into account. However, in our study, we also consider the following requirements. We allow for conference organisers to express preferences regarding the scheduling of tracks into sessions. Our models take into account that certain rooms might not be available during some sessions. We consider presenters' conflicts where a presenter might have multiple submissions which is circumvented in Stidsen et al. (2018) study due to the policy of the conference. In addition, our approach is suitable for hybrid and online conferences as we consider the timezones of the presenters. Our models accommodate the preferences for sessions and preferences for rooms (accessibility and facility reasons) from presenters. In addition to presenters' conflicts, we also consider attendees' and track chairs' conflicts. Furthermore, we consider the scheduling of submissions that have a different amount of required time slots. In contrast to Stidsen et al. (2018) who generate a high-level schedule, we generate both high-level and low-level schedules considering preferences and requirements on both tracks and submissions levels. In our work we do not consider areas and buildings as described in Stidsen et al. (2018), neither the assignment of similar areas into same buildings, but we allow for conference organisers to specify similar tracks which should not be scheduled in parallel. Our models are significantly different mainly because the EURO conference is unusually large and follows an unusual hierarchical structure, whose characteristics we believe are not representative of typical conferences.

Vangerven et al. (2018) addressed the CSP of four conferences, namely the MathSport 2013, MAPSP 2015 & 2017, and ORBEL 2017. Their main objective was to maximise the satisfaction of attendees in terms of attending their preferred submissions. Preferences of attendees were collected via e-mail by the authors. A secondary objective was the minimisation of session hopping, which occurs when participants miss parts of their

preferred submissions because they are scheduled in different rooms. A third objective of this work was to satisfy the preferences of the presenters. To achieve the three objectives, the researchers proposed a hierarchical three-phased approach. Firstly, the authors maximise the satisfaction of attendees with an integer programming model. Then, they minimise session hopping with either dynamic programming or heuristic approach, and in the last phase, they satisfy the preferences of the presenters with an integer programming model. Their proposed method was implemented on four medium size conferences for which the number of rooms that allowed parallel scheduling ranged between 2 to 4, submissions ranged between 76 to 90, and profiles of attendees ranged between 58 to 101. They also showed that the CSP with n rooms for parallel scheduling is $\mathcal{NP}$-hard when $n \geq 3$.

Another study on CSP was conducted by Manda et al. (2019) who used the dataset from Ecology 2013 for testing purposes to deliver a schedule for the Evolution 2014 conference. While the former conference spanned 5 days including 324 submissions, 8 sessions with 10 time slots each, and 5 rooms, the latter conference spanned 4 days including 1014 submissions, 16 sessions with 5 time slots each, and 14 rooms. In their study, all submissions need to be scheduled into time slots by maximising the coherence within sessions and minimising the similarity between sessions that are scheduled in parallel. Three different approaches, a random, a greedy, and an integer linear programming model were implemented for the initialisation process. These initial solutions were further optimised with a hill climbing algorithm and a simulated annealing algorithm, and with two different optimisation methods. While the first optimisation method optimised the objectives concurrently, the other method was sequential. Through experimentation, the authors found that the different approaches for the initialisation process did not affect the final solution, and that concurrent optimisation was superior to sequential. Therefore, they followed the random initialisation process and the concurrent optimisation method to generate the final schedule. The delivered schedule though was significantly altered by the program committee.

The above case studies indicate that conferences that were considered for an exactly optimised scheduling typically involved up to a few hundred submissions, while a few larger conferences with over a thousand of submissions were addressed using approximate methods. However, there is a large variety of both exact and approximate methods as well as their constraints and objectives. In the next sections we present new models which have been developed to provide a unifying and generic framework for conference scheduling and would fit most of the conferences described in the above case studies.

3.3 Description of the Conference Scheduling Problem

In this section, we describe the essential elements of the problem studied in this paper to keep it self-contained; a more detailed description of the problem is provided in Kheiri et al. (2023). We also discuss the functionality of penalties and weights of our approach and present the real-world conferences that motivated our research. We consider that a conference is defined by a set of track-submission pairs $\mathcal{TSU}$, a set of sessions $\mathcal{S}$, and a set of rooms $\mathcal{R}$, whose descriptions and relationships are described next.

We split the whole time of the conference duration into a set of sessions $\mathcal{S}$ representing certain time periods between breaks, which include lunch and coffee breaks, that allow attendees to move between rooms. That is, we assume that every attendee will stay in the same room during a session. We assume that the sessions in $\mathcal{S} := \{1, 2, \ldots, S\}$ are chronologically ordered, so S is the last session of the conference. Each session $s \in \mathcal{S}$ consists of a number of time slots defined by a set $\mathcal{TS}_s$. Each time slot ts is defined as a fixed predefined amount of time available for scheduling a submission (e.g., 15 or 20 minutes). For instance, if a time slot has a 15 minute duration, then a session that consists of 4 time slots has a 60 minute duration. We assume that all time slots have the same fixed predefined amount of time, but we do not assume that all sessions have the same number of time slots. We use r to denote the room from the set of available rooms $\mathcal{R}$. We assume that submissions are presented in parallel during sessions, and the maximum number of sessions that can be scheduled in parallel is given by the total number of available rooms.

The conference requires scheduling of a number of submissions which we include in set $\mathcal{TSU}$, where each submission $(t, su) \in \mathcal{TSU}$ is uniquely identified as a pair of its track and the submission itself. We assume that each submission (t, su) is categorised into exactly one track from the set of tracks $\mathcal{T}$ based on their subject similarity, i.e., $t \in \mathcal{T}$, and we allow for different tracks to contain different number of submissions $\mathcal{SU}_t$, i.e., $su \in \mathcal{SU}_t$. Each submission usually requires one time slot for scheduling at the conference. However, some submissions, such as a keynote speech, might require additional time slots, which must be scheduled in the same session. For each submission (t, su), we define $n^{(t,su)}$ which indicates its number of required time slots. Similarly, note that a track may utilise more than one session, but we do not know how many sessions beforehand. Although "submission" usually refers to a research paper, it may also refer to any type of formal event that requires scheduling, such as a keynote speech, job

market, tutorial, workshop, or any other event. Additional tracks are created accordingly for such formal events.

Furthermore, let $\mathcal{H}$ be the set of all humans involved in the conference. A "presenter" is defined as a person who presents a submission during the conference; we allow for more than one presenter of each submission (which is required e.g., for a panel discussion). We define a set $\mathcal{TSU}^p$ which contains the submissions for which the presenter $p \in \mathcal{P} \subseteq \mathcal{H}$ is the same.

Moreover, we have the following hard requirements:

- **Tracks must be scheduled in only one room:** It is not allowed for a track to utilise more than one room as this would cause inconvenience for attendees that would have to move between rooms. This also implies that the same track must not be scheduled within the same session in different rooms. Assuming that attendees usually attend the whole track of their interest, scheduling the same track in parallel would result in attendees missing some of their preferred submissions.
- **Schedule must be free of presenters' conflicts:** In many conferences, authors are allowed to present more than one submission. Therefore, we must ensure that two or more submissions which belong to the same presenter are either scheduled within the same room of a session or scheduled within different sessions. Note that a submission does not necessarily have only one presenter, it may include multiple presenters. We could relax this constraint by considering conflicts only on a time slot level instead of session level. However, this would require presenters to move between rooms which is inconvenient, and they would most likely want to attend both of those sessions.

Apart from these hard requirements, we assume that conference organisers have a number of requests. These are soft requirements that are used to adjust scheduling preferences of conference organisers and satisfy additional requirements of a conference. To accommodate requests, we use a penalty system and a weight system. The soft requirements are:

- **Track-Session request:** We allow to penalise scheduling any track t into any session s by assigning a non-negative penalty α_s^t.

- **Track-Room request:** A non-negative penalty β_r^t is applied for scheduling a certain track t into a specified room r. This allows the allocation of tracks with high expected attendance into appropriate rooms and vice versa.

- **Session-Room request:** In case a room r is unavailable for scheduling during a particular session s, we apply a non-negative penalty $\gamma_{s,r}$.

- **Submission-Session request:** We allow to penalise scheduling any submission (t, su) into any session s by assigning a non-negative penalty $\epsilon_s^{(t,su)}$. This allows the accommodation of preferences from presenters regarding their preferred scheduled time.

- **Submission-Timezone request:** We allow to penalise scheduling any submission (t, su) into any session s by assigning a non-negative penalty $\delta_s^{(t,su)}$ as a result of unsuitability of any presenter's timezone. This is analogous to the Submission-Session request but we keep it separate to allow for a different weight. Considering the timezone may be important for online presenters as well as for in-person presenters experiencing a jet-lag.

- **Submission-Room request:** $\zeta_r^{(t,su)}$ specifies a non-negative penalty for scheduling a particular submission (t, su) into a specified room r. This is used for the consideration of special requests from presenters who might need to present at a particular room for accessibility or facilities issues.

Each of the above penalties is weighted by a corresponding non-negative value from set $W = \{w_\alpha, w_\beta, w_\gamma, w_\delta, w_\epsilon, w_\zeta\}$ so as to allow the prioritisation of requests.

The goal is to assign all tracks into rooms and sessions, and assign all submissions into sessions in such a way that weighted penalties are minimised and all hard requirements are satisfied. Practically, the problem requires the generation of two schedules, a high-level schedule which indicates the room and sessions of each track, and a low-level schedule indicating the room and session of each submission. Note that even though we generate a low-level schedule, it is still possible for organisers to rearrange the order of submissions within the same session without any impact on the quality of the solution.

Penalties and weights are used to adjust scheduling preferences. Setting different values for penalties allows the prioritisation of certain requests. For example, if satisfying a certain Track-Session request is more important than another Track-Session request, then we set a greater penalty value for the more important one. In addition, we can

TABLE 3.1: Characteristics of the instances

Instance	$\lvert\mathcal{TSU}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{TS}\rvert$	Required TS	Available TS
GECCO19	202	29	13	10	45	215	450
GECCO20	158	24	7	8	28	161	200
GECCO21	138	27	6	8	24	150	192
N2OR	35	8	4	4	9	36	36
OR60	329	45	8	23	24	417	540
OR60F	279	45	8	23	24	353	540
OR60F2	556	72	16	23	49	702	1,115
OR60F3	1,112	72	32	23	105	1,404	2,403

prioritise the type of requests that are more important to be satisfied with the use of weights. For instance, if satisfying Submission-Room requests is more important than Submission-Session requests, then we set a greater weight value for Submission-Room requests. Thus, by adjusting penalties and weights, we can explore different solutions along with their trade-offs.

The motivation for this work originated from scheduling the Genetic and Evolutionary Computation Conference (GECCO), the OR Society's 60th Annual Conference (OR60), and the New to OR Conference (N2OR) (Kheiri et al., 2023). We present the details of the instances in Table 3.1, where $|\mathcal{TSU}|$ is the number of submissions, $|\mathcal{T}|$ is the number of tracks, $|\mathcal{S}|$ is the number of sessions, $|\mathcal{R}|$ is the number of rooms, $|\mathcal{TS}|$ is the number of time slots, Required TS is the required number of time slots by all the submissions, and Available TS is the number of available time slots for scheduling across all the sessions and rooms, excluding penalised time slots due to $\gamma_{s,r}$. Note that the number of time slots $|TS|$ is given by summing up the time slots of each session, whereas the number of available time slots, Available TS, is given by $|R| \times |TS|$ and subtracting any penalised time slots. These characteristics give some rough idea about the size of each instance, yet do not define a given problem fully as the importance of violating a given constraint is not provided.

3.4 Methodology

In this section, we first discuss the benefits of using our proposed spreadsheet template for conference scheduling problems. Then we provide an overview of the notation, followed by the formal formulation of the exact model.

We use a spreadsheet file to store input data, which follows a specific template that offers flexibility. This flexible template has been created with the purpose of providing a generic approach suitable for conference scheduling problems. By using a single data format we want to minimise the need of modifying algorithms to fit specific data formats provided by conferences. Instead of adjusting algorithms each time, we could just transfer the given data to the template. In addition, our template makes it easy to modify weights and scheduling preferences. We present the spreadsheet file in detail in Appendix C.

3.4.1 Model Notation

Sets and Indices. We use the following sets and their corresponding indices in our formulation:

$t \in \mathcal{T}$: The set of tracks

$su \in \mathcal{SU}_t$: The subset of submissions belonging to a track t

$(t, su) \in \mathcal{TSU}$: The set of submissions where $\{(t, su) : t \in \mathcal{T}$ and $su \in \mathcal{SU}_t\}$

$h \in \mathcal{H}$: The set of humans involved in the conference

$p \in \mathcal{P}$: The set of presenters

$(t, su) \in \mathcal{TSU}^p$: The subset of submissions belonging to presenter p

$r \in \mathcal{R}$: The set of rooms

$s \in \mathcal{S}$: The set of sessions

$ts \in \mathcal{TS}_s$: The subset of time slots belonging to session s

Parameters. We use the following parameters in our formulation:

α_s^t : Penalty for scheduling track t into session s

w_α : Weight of penalty α_s^t

β_r^t : Penalty for scheduling track t into room r

w_β : Weight of penalty β_r^t

$\gamma_{s,r}$: Penalty for utilising room r within session s

w_γ : Weight of penalty $\gamma_{s,r}$

$\delta_s^{(t,su)}$: Penalty for scheduling submission (t, su) within session s
for which the timezone is unsuitable

w_δ : Weight of penalty $\delta_s^{(t,su)}$

$\epsilon_s^{(t,su)}$: Penalty for scheduling submission (t, su) within session s

w_ϵ : Weight of penalty $\epsilon_s^{(t,su)}$

$\zeta_r^{(t,su)}$: Penalty for scheduling submission (t, su) into room r

w_ζ : Weight of penalty $\zeta_r^{(t,su)}$

$MaxS_t$: The upper bound on the number of required sessions of track t

$n^{(t,su)}$: The number of required time slots of submission (t, su)

$|\mathcal{TS}_s|$: The number of time slots within session s

$|\mathcal{TSU}^p|$: The number of submissions belonging to presenter p

$M_s^p = min\{|\mathcal{TSU}^p|, |\mathcal{TS}_s|\}$: The upper bound on the number of submissions that presenter p could possibly present during session s (in the same room)

Decision Variables. We use the following decision variables in our formulation:

$Z_{s,r}^t \in \{0, 1\}$: 1 if track t is scheduled in session s and room r; 0 if not

$Y_r^t \in \{0, 1\}$: 1 if track t is assigned room r; 0 if not

$X_{s,r}^{(t,su)} \in \{0, 1\}$: 1 if submission (t, su) is scheduled in session s and room r; 0 if not

3.4.2 Constraints

In the exact model we have the following (hard) constraints:

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}} X_{s,r}^{(t,su)} = 1 \qquad \forall\ (t,su) \in \mathcal{TSU} \quad (3.1)$$

$$M_s^p X_{s,r}^{(t,su)} + \sum_{r'\in\mathcal{R}\setminus\{r\}} \sum_{(t',su')\in\mathcal{TSU}^p} X_{s,r'}^{(t',su')} \leq M_s^p \quad \forall\ s\in\mathcal{S}, \forall\ r\in\mathcal{R}, \forall\ p\in\mathcal{P}, \forall\ (t,su)\in\mathcal{TSU}^p \quad (3.2)$$

$$\sum_{r\in\mathcal{R}} Y_r^t = 1 \qquad \forall\ t\in\mathcal{T} \quad (3.3)$$

$$\sum_{s\in\mathcal{S}} Z_{s,r}^t - MaxS_t Y_r^t \leq 0 \qquad \forall\ r\in\mathcal{R}, \forall\ t\in\mathcal{T} \quad (3.4)$$

$$\sum_{t\in\mathcal{T}} Z_{s,r}^t \leq 1 \qquad \forall\ s\in\mathcal{S}, \forall\ r\in\mathcal{R} \quad (3.5)$$

$$\sum_{su\in\mathcal{SU}_t} n^{(t,su)} X_{s,r}^{(t,su)} - |\mathcal{TS}_s| Z_{s,r}^t \leq 0 \qquad \forall\ s\in\mathcal{S}, \forall\ r\in\mathcal{R}, \forall\ t\in\mathcal{T} \quad (3.6)$$

$$\sum_{su\in\mathcal{SU}_t} X_{s,r}^{(t,su)} - Z_{s,r}^t \geq 0 \qquad \forall\ s\in\mathcal{S}, \forall\ r\in\mathcal{R}, \forall\ t\in\mathcal{T} \quad (3.7)$$

$$Z_{s,r}^t \in \{0,1\} \qquad \forall\ t\in\mathcal{T}, \forall\ s\in\mathcal{S}, \forall\ r\in\mathcal{R} \quad (3.8)$$

$$Y_r^t \in \{0,1\} \qquad \forall\ t\in\mathcal{T}, \forall\ r\in\mathcal{R} \quad (3.9)$$

$$X_{s,r}^{(t,su)} \in \{0,1\} \qquad \forall\ (t,su)\in\mathcal{TSU}, \forall\ s\in\mathcal{S}, \forall\ r\in\mathcal{R} \quad (3.10)$$

The first set of constraints, Eq. (3.1), ensures that each submission must be scheduled into exactly one session. The next set of constraints, Eq. (3.2), resolves presenters' conflicts, where $(t, su) \in \mathcal{TSU}^p \subset \mathcal{TSU}$ is a set of submissions including a presenter conflict such that $|\mathcal{TSU}^p| > 1$. For every presenter with multiple submissions, this set of constraints handles conflicts depending on the track of such submissions. If conflicting submissions belong to the same track, then such submissions are not allowed to be scheduled within different rooms of the same session. On the other hand, if conflicting submissions belong to different tracks, then such submissions are scheduled within different sessions.

Constraints within Eq. (3.3) ensure that exactly 1 room is allocated to each track. Then we use "bigM" constraints, Eq. (3.4), to allocate tracks to their assigned room, where $MaxS_t$ is the upper bound on the total number of sessions that a given track might require to fit all the submissions. Although we could simply set $MaxS_t$ equal to the total number of sessions, we introduced this scenario to decrease the value of $MaxS_t$ so as to strengthen our formulation. The scenario assumes that a track will utilise at most $MaxS_t$ sessions, where $MaxS_t$ is given by sorting sessions in ascending order based on their number of time slots and sessions are added until the number of time slots is greater or equal to the number of time slots that the given track requires. For example, suppose "Forecasting" track has 6 submissions and "Simulation" track has 4 submissions, each requiring one time slot, and four sessions are available with the following number of time slots: $4, 3, 2, 2$. We sort the sessions $(2, 2, 3, 4)$ and set $MaxS_{Forecasting} = 3$ because $2+2+3 \geq 6$, and $MaxS_{Simulation} = 2$ because $2+2 \geq 4$, rather than setting $MaxS = 4$ for both tracks.

Constraints Eq. (3.5) ensure that at most one track is scheduled into every given session and room. Based on the assignment of Eq. (3.4), "bigM" constraints Eq. (3.6) ensure that at most $|\mathcal{TS}_s|$ time slots are allowed to be utilised for a given session and room, where $|\mathcal{TS}_s|$ is the number of available time slots corresponding to session $s \in \mathcal{S}$, and $n^{(t,su)}$ is the total number of time slots that a given submission requires. For instance, suppose track "Forecasting" is assigned into session "9-11am" and room "A", which is defined as $Z^{Forecasting}_{9-11am,A} = 1$. Also, suppose session "9-11am" has 3 time slots available $|TS_{9-11am}| = 3$ and all $n^{(t,su)} = 1$, then at most 3 submissions corresponding to track "Forecasting" are allowed to be scheduled into session "9-11am" and room "A". The next set of constraints, Eq. (3.7), ensures that a given track is not assigned into a session-room pair for which no submissions are scheduled. In other words, we prevent $Z^t_{s,r} = 1$ if none of the submissions is scheduled within the given session and room.

Lastly, Eq. (3.8), Eq. (3.9), and Eq. (3.10) indicate that our decision variables $Z^t_{s,r}$,, Y^t_r, and $X^{(t,su)}_{s,r}$ are binary.

Note that it is possible for organisers to freely rearrange submissions within sessions. This has no impact on the quality of the solution because the changes are performed on a time slot level.

3.4.3 Objective

Recall from subsection 3.4.1 that $Z^t_{s,r}$ is a binary decision variable which is used for assigning tracks into sessions, where track $t \in \mathcal{T}$, session $s \in \mathcal{S}$, and room $r \in \mathcal{R}$, e.g., when $Z^{Forecasting}_{9-11am,A} = 1$ then track "Forecasting" is allocated into session "9-11am" and room "A". The coefficient of $Z^t_{s,r}$ is a weighted sum of penalties α^t_s, β^t_r, and $\gamma_{s,r}$. Penalty α^t_s is incurred for scheduling a specific track into a specified session (Tracks-Sessions penalty) weighted by w_α. Penalty β^t_r is incurred for scheduling a specific track into a specified room (Tracks-Rooms penalty) weighted by w_β, and penalty $\gamma_{s,r}$ is incurred for utilising a specific room within a specified session (Sessions-Rooms penalty) weighted by w_γ.

Recall also that $X^{(t,su)}_{s,r}$ is a binary decision variable which is used to schedule submissions into sessions, where submission $(t, su) \in \mathcal{TSU}$ corresponds to track $t \in \mathcal{T}$, session $s \in \mathcal{S}$, and room $r \in \mathcal{R}$, e.g., when $X^{(Forecasting,FC1)}_{9-11am,A} = 1$ this means that submission "FC1" corresponding to track "Forecasting" is scheduled in session "9-11am" and room "A". The coefficient of variable $X^{(t,su)}_{s,r}$ is a weighted sum of $\delta^{(t,su)}_s$, $\epsilon^{(t,su)}_s$, and $\zeta^{(t,su)}_r$. Penalty $\delta^{(t,su)}_s$ is a penalty for assigning a specific submission within a session for which the timezone is unsuitable (Submissions-Timezones penalty) weighted by w_δ. e.g., a submission is scheduled within a session that is unsuitable for the timezone of the presenter (03:00 am). Penalty $\epsilon^{(t,su)}_s$ is a penalty for scheduling a specific submission within a specified session (Submissions-Sessions penalty) weighted by w_ϵ, and $\zeta^{(t,su)}_r$ is a penalty for assigning a specific submission into a specified room (Submissions-Rooms penalty) weighted by w_ζ.

Based on the above, we formulate the following objective for the exact model:

$$\begin{aligned} \min \quad & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})Z^t_{s,r} \\ + & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{(t,su)\in\mathcal{TSU}}(w_\delta\delta^{(t,su)}_s + w_\epsilon\epsilon^{(t,su)}_s + w_\zeta\zeta^{(t,su)}_r)X^{(t,su)}_{s,r} \end{aligned} \tag{3.11}$$

The objective function, Eq. (3.11), assigns tracks into rooms and sessions, and submissions into sessions by minimising the penalties associated with both tracks and submissions. To reduce the size and complexity of the model, we assign submissions into time slots of sessions in a post-processing algorithm after the IP model is solved. Note that Eq. (3.11) implies that $\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{(t,su)\in\mathcal{TSU}} X_{s,r}^{(t,su)} = |\mathcal{TSU}|$, where $|\mathcal{TSU}|$ is a constant (the number of submissions). However, the sum of $Z_{s,r}^{t}$ is not a constant which may result in some $Z_{s,r}^{t}$ variables being equal to 1 without any submissions scheduled during a given session and room. We resolve this by including constraints Eq. (3.7). Alternatively, one could use $(1 + w_\alpha\alpha_s^t + w_\beta\beta_r^t + w_\gamma\gamma_{s,r})$ as a coefficient of $Z_{s,r}^{t}$ to ensure that variables, for which the sum of penalties is zero, are minimised.

3.5 Computational Results

In this section, we present the results of the exact model which was tested on a number of real and artificial instances. For the real instances, we obtained past data and scheduling preferences information which were used to set penalties values and weights. The artificial instances were created due to a particular instance being infeasible and to test the models on larger instances.

The results were generated on an i7-11370H CPU Intel Processor with 8 cores at 3.30GHz with 16.00 GB RAM. We used Python 3.8.3 to build our models which were solved using Gurobi 9.5.0. We use the following Gurobi parameters for the exact model; $MIPGap = 0$ and $timeLimit = 3600$. The former parameter allows the solver to terminate only when the gap between the lower and upper objective bound is zero, while the latter parameter implies a time limit of one hour. Note that even though the time required to build the models is negligible for some instances, it is significant for other instances. We report the time required to build the models for each instance in Appendix D.

In Table 3.2, we present the penalty values that were used for each type of constraint, where α_s^t indicates values for Tracks-Sessions penalties, β_r^t indicates values for Tracks-Rooms penalties, $\gamma_{s,r}$ indicates values for Sessions-Rooms penalties, $\delta_s^{(t,su)}$ indicates values for Submissions-Timezones penalties, $\epsilon_s^{(t,su)}$ indicates values for Submissions-Sessions penalties, $\zeta_r^{(t,su)}$ indicates values for Submissions-Rooms penalties, and "-" denotes that the type of penalties is not used. For example, in GECCO19 instance, we set $\beta_r^t = 10$ for cases that are less important to be satisfied, while we set $\beta_r^t = 10000$ for significant cases. Penalty values reflect the importance of satisfying the particular constraint.

TABLE 3.2: Penalty values used for each type of constraint.

Instance	α_s^t	β_r^t	$\gamma_{s,r}$	$\delta_s^{(t,su)}$	$\epsilon_s^{(t,su)}$	$\zeta_r^{(t,su)}$
N2OR	-	-	-	-	[1]	-
GECCO19	[10000]	[10, 10000]	-	-	[1]	-
GECCO20	[10000]	[10, 10000]	[10000]	-	[1, 10]	-
GECCO21	[10000]	[10, 10000]	-	[1, 10]	[10000]	-
OR60	[10, 100]	[1, 100]	[1000]	-	[1]	-
OR60F	[10, 100]	[1, 100]	[1000]	-	[1]	-
OR60F2	[10, 100]	[1, 100]	[1000]	-	[1]	-
OR60F3	[10, 100]	[1, 100]	[1000]	-	[1]	-

For most instances we keep all weights identical and equal to one ($w_\alpha = w_\beta = w_\gamma = w_\delta = w_\epsilon = w_\zeta = 1$). The only exception in which we set different weights is for the GECCO conferences where we use the following weights; $w_\alpha = 100$, $w_\gamma = 10$, and $w_\epsilon = 100$. In addition, we set $w_\delta = 100$ for GECCO21 which was held online.

Our results are summarised in Table 3.3, where Objective indicates the aggregation of penalties caused by violations of soft constraints, Gap indicates the relative gap between the two objective bounds, Time indicates the required time for the solver to terminate in seconds, and N/A indicates the value is not available. The violations of soft constraints are presented in detail in Table 3.4, where α's indicates incurred weighted Tracks-Sessions penalties (i.e., $\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}} w_\alpha \alpha_s^t Z_{s,r}^t$), β's indicates incurred weighted Tracks-Rooms penalties, γ's indicates incurred Sessions-Rooms penalties, δ's indicates incurred Submissions-Timezones penalties, ϵ's indicates incurred Submissions-Sessions penalties, ζ's indicates incurred Submissions-Rooms penalties, N/A indicates the value is not available, and "-" denotes that the type of penalties is not used. For each instance, we set a time limit of one hour and we report the number of variables and constraints, the objective value, the gap which shows the relative gap between the two objective bounds, and the time required for the solver to terminate in seconds. An infeasible status for the objective means that the solution is infeasible. Additionally, note that even though we include the Submission-Room request ($\zeta_r^{(t,su)}$) in the model, there is no available data for this request.

Our model obtained the optimal solution for the N2OR conference instantly without any violations. The model achieved a solution with a relative gap of 0.001% within 17 seconds for GECCO19, but the optimal solution was not found within the time limit of 1 hour. The solution had 1 violation for Tracks-Sessions requests ($w_\alpha \times \alpha_s^t = 100 \times 10,000 = 1,000,000$) and 1 violation for Tracks-Rooms requests ($w_\beta \times \beta_r^t = 1 \times 10 =$

Table 3.3: Exact Model Results

Instance	Variables	Constraints	Objective	Gap (%)	Time (s)
N2OR	720	347	0	0.000	0.1
GECCO19	30,320	26,911	1,000,010	0.001	3,600.0
GECCO20	10,384	5,750	6,110	0.000	8.3
GECCO21	8,136	4,797	11,130	0.000	19.7
OR60	69,851	36,185	Infeasible	N/A	3.6
OR60F	60,651	30,983	424	0.000	13.9
OR60F2	232,760	77,724	10	0.000	88.9
OR60F3	873,080	153,720	0	0.000	137.4

Table 3.4: Exact Model Violations

Instance	α's	β's	γ's	δ's	ϵ's	ζ's	Total
N2OR	-	-	-	-	0	-	0
GECCO19	1,000,000	10	-	-	0	-	1,000,010
GECCO20	0	10	0	-	6,100	-	6,110
GECCO21	0	30	-	11,100	0	-	11,130
OR60	N/A	N/A	N/A	-	N/A	-	N/A
OR60F	400	0	0	-	24	-	424
OR60F2	0	0	0	-	10	-	10
OR60F3	0	0	0	-	0	-	0

10). GECCO20 had 1 violation for Tracks-Rooms requests ($w_\beta \times \beta_r^t = 1 \times 10 = 10$) and 16 violations for Submissions-Sessions requests ($w_\epsilon \times (11 \times \epsilon_s^{(t,su)} + 5 \times \epsilon_s^{(t,su)}) = 100 \times (11 \times 1 + 5 \times 10) = 6,100$). The solution of GECCO21 had 3 violations for Tracks-Rooms requests ($w_\beta \times 3 \times \beta_r^t = 1 \times 3 \times 10 = 30$) and 30 violations for Submissions-Timezones requests ($w_\delta \times (21 \times \delta_s^{(t,su)} + 9 \times \delta_s^{(t,su)}) = 100 \times (21 \times 1 + 9 \times 10) = 11,100$). OR60 had some extensive tracks that resulted in infeasibility due to constraints Eq. (3.6) and Eq. (3.7). In other words, the number of available sessions was not enough to avoid scheduling the same track in parallel, and consequently tracks could not be limited to utilise only one room. Therefore, we reduced the instance by removing submissions of such tracks to create a feasible version of OR60, which we refer to as OR60F. The model found the optimal solution which had 4 violations for Tracks-Sessions requests ($w_\alpha \times 4 \times \alpha_s^t = 1 \times 4 \times 100 = 400$) and 24 violations for Submissions-Sessions requests ($w_\epsilon \times 24 \times \epsilon_s^{(t,su)} = 1 \times 24 \times 1 = 24$). OR60F2 and OR60F3 are both larger versions of the OR60 instance. The solution of the former instance had only 10 violations for Submissions-Sessions requests ($w_\epsilon \times 10 \times \epsilon_s^{(t,su)} = 1 \times 10 \times 1 = 10$), while the solution of the latter instance had no violations.

3.5.1 Infeasible Instances

Sometimes, it is not possible to completely schedule each track into exactly one room, which results in an infeasible model (e.g., OR60). A solution to this issue is to relax constraints Eq. (3.3) by changing the right hand side to $\leq MaxR_t$, where $MaxR_t$ is the maximum number of rooms that could be assigned to a track. We followed this procedure for the OR60 instance which we describe next. Firstly, we identified the four tracks that required more than one room to be scheduled. Specifically, three tracks had to utilise 2 rooms and one track had to utilise 3 rooms. However, the model was still infeasible because of presenters' conflicts and we had to further relax constraints Eq. (3.3). After changing the right hand side value several times, we found a feasible model in which two tracks utilise 2 rooms, one track utilises 3 rooms, and one track utilises 4 rooms. Our model found the optimal solution within 51.2 seconds which had an objective value of 106. The solution had 1 violation for Tracks-Sessions requests ($w_\alpha \times \alpha_s^t = 1 \times 100 = 100$), 3 violations for Tracks-Rooms requests ($w_\beta \times 3 \times \beta_r^t = 1 \times 3 \times 1 = 3$), and 3 violations for Submissions-Sessions requests ($w_\epsilon \times 3 \times \epsilon_s^{(t,su)} = 1 \times 3 \times 1 = 3$).

3.6 Extended Formulation

In this section, we present an extension of our formulation in which we consider additional constraints including some with non-linear terms. We first provide additional definitions, followed by a discussion of the additional soft and hard requirements. Next, we present the formal formulation of the extended model along with computational results and we conclude the section discussing the limitations of the extended model.

We refer to an "attendee" as a person who is a spectator of a submission, and define a set $\mathcal{TSU}^a$ which includes the submissions that the attendee $a \in \mathcal{A} \subseteq \mathcal{H}$ wishes to attend. A "track chair" is defined as the person who attends all the submissions of a track, and the set $\mathcal{T}^c$ consists of tracks chaired by the same person $c \in \mathcal{C} \subseteq \mathcal{H}$. Note that a human is allowed to have multiple roles and may attend a conference as a presenter, an attendee, and a track chair.

We consider the following additional hard requirements:

- **Avoid scheduling similar tracks in parallel:** Organisers can specify some tracks as being similar and request not to schedule such tracks in parallel. For each track t we denote a set $\mathcal{T}^t \subset \mathcal{T}$ of similar tracks including t.

- **Schedule must be free of attendees' conflicts:** For attendees who have declared attending preferences, we resolve conflicts by scheduling such submissions in different sessions.
- **Schedule must be free of track chairs' conflicts:** In case of a track chair being responsible for more than one track, we schedule such tracks within different sessions.

Additionally, we consider the following soft requirement:

- **Consecutive track sessions:** Tracks are preferred to be scheduled in consecutive sessions to achieve a cohesive schedule.

The additional soft requirement, is weighted by a non-negative value π_K. This weight, however, is a subsidy, not penalty, and therefore it has a negative sign as the objective is to be minimised. We model the consecutive tracks requirement as $Z^t_{s,r} \times Z^t_{s+1,r}$ which decreases the objective value by π_K when a track t is scheduled consecutively within sessions s and $s+1$. For instance, suppose we need to schedule track "Forecasting" into room "A" and two sessions from the set $S = \{9{-}11am, 11{-}1pm, 1{-}3pm\}$ are required. If "9-11am" and "1-3pm" sessions are used, then $Z^{Forecasting}_{9-11am,A} \times Z^{Forecasting}_{11-1pm,A} = 1 \times 0 = 0$ and $Z^{Forecasting}_{11-1pm,A} \times Z^{Forecasting}_{1-3pm,A} = 0 \times 1 = 0$. On the other hand, if either "9-11am" and "11-1pm" or "11-1pm" and "1-3pm" are used, then we prefer any of these combinations which would result in decreasing the objective value by π_K given that none other penalties are incurred. Note that the more consecutive variables become equal to 1, the fewer the violations of the consecutive tracks soft constraint is achieved. By including the additional soft requirement in Eq. (3.11), we achieve the following objective function with non-linear terms:

$$\begin{aligned} \min \quad & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})Z^t_{s,r} \\ & + \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{(t,su)\in\mathcal{T}\mathcal{S}\mathcal{U}}(w_\delta\delta^{(t,su)}_s + w_\epsilon\epsilon^{(t,su)}_s + w_\zeta\zeta^{(t,su)}_r)X^{(t,su)}_{s,r} \\ & - \pi_K \times \sum_{s\in\mathcal{S}\setminus\{S\}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}} Z^t_{s,r} \times Z^t_{s+1,r} \end{aligned} \tag{3.12}$$

Then, we convert the non-linear terms of the objective function into linear by introducing new binary variables. Let $K^t_{s,r}$ be a product variable of $Z^t_{s,r}$ and $Z^t_{s+1,r}$ which is used to schedule tracks in a consecutive manner.

3.6.1 Constraints

In the extended model we have the following (hard) constraints:

(3.1) – (3.10)

$$\sum_{r \in R} \sum_{t' \in T^t} Z_{s,r}^{t'} \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, t \in \mathcal{T} \quad (3.13)$$

$$M_s^a X_{s,r}^{(t,su)} + \sum_{r' \in \mathcal{R} \setminus \{r\}} \sum_{(t',su') \in \mathcal{TSU}^a} X_{s,r'}^{(t',su')} \leq M_s^a \quad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, a \in \mathcal{A}, \forall\, (t,su) \in \mathcal{TSU}^a \quad (3.14)$$

$$\sum_{r \in \mathcal{R}} \sum_{t^c \in \mathcal{T}^c \subset \mathcal{T}} Z_{s,r}^{t^c} \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, c \in \mathcal{C} \quad (3.15)$$

$$K_{s,r}^t \leq Z_{s,r}^t \qquad \forall\, t \in \mathcal{T}, \forall\, r \in \mathcal{R}, \forall\, s \in \mathcal{S} \setminus \{S\} \quad (3.16)$$

$$K_{s,r}^t \leq Z_{s+1,r}^t \qquad \forall\, t \in \mathcal{T}, \forall\, r \in \mathcal{R}, \forall\, s \in \mathcal{S} \setminus \{S\} \quad (3.17)$$

$$K_{s,r}^t \geq Z_{s,r}^t + Z_{s+1,r}^t - 1 \qquad \forall\, t \in \mathcal{T}, \forall\, r \in \mathcal{R}, \forall\, s \in \mathcal{S} \setminus \{S\} \quad (3.18)$$

$$K_{s,r}^t \in \{0,1\} \qquad \forall\, t \in \mathcal{T}, \forall\, r \in \mathcal{R}, \forall\, s \in \mathcal{S} \setminus \{S\} \quad (3.19)$$

Constraints Eq. (3.13) prevent scheduling specified tracks in parallel, where $\mathcal{T}^t \subset \mathcal{T}$ is a set of similar tracks such that $|\mathcal{T}^t| > 1$. The next set of constraints, Eq. (3.14), resolves attendees' conflicts, where $(t, su) \in \mathcal{TSU}^a \subset \mathcal{TSU}$ is a set of submissions including an attendee conflict such that $|\mathcal{TSU}^a| > 1$. In addition, M_s^a is an upper bound on the number of declared submissions that attendee a could possibly attend during session s given by $min\{|TSU^a|, |\mathcal{TS}_s|\}$, where $|TSU^a|$ is the number of declared submissions by attendee a. For every attendee with multiple declared submissions, this set of constraints

handles conflicts depending on the track of such submissions. If conflicting submissions belong to the same track, then such submissions are not allowed to be scheduled within different rooms of the same session. On the other hand, if conflicting submissions belong to different tracks, then such submissions are scheduled within different sessions. The next set of constraints, Eq. (3.15), resolves track chairs' conflicts, where $t^c \in \mathcal{T}^c \subset \mathcal{T}$ is a set of tracks including a track chair conflict such that $|\mathcal{T}^c| > 1$. For every track chair responsible for more than one track, this set of constraints ensures that such tracks are not scheduled within the same session. Constraints from Eq. (3.16) up to Eq. (3.18) introduce auxiliary variables $K^t_{s,r}$ which we will use to convert the non-linear terms in the objective into linear terms, while constraints Eq. (3.19) indicate that these variables are binary.

3.6.2 Objective

After replacing the non-linear terms in Eq. (3.12), we obtain the following objective function:

$$\begin{aligned} \min \quad & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})Z^t_{s,r} \\ & + \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{(t,su)\in\mathcal{TSU}}(w_\delta\delta^{(t,su)}_s + w_\epsilon\epsilon^{(t,su)}_s + w_\zeta\zeta^{(t,su)}_r)X^{(t,su)}_{s,r} \\ & - \pi_K \times \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}K^t_{s,r} \end{aligned} \tag{3.20}$$

The new objective function, Eq. (3.20), generates a schedule by minimising the penalties related to both tracks and submissions, and the violations regarding consecutive track sessions.

3.6.3 Computational Results

We use the same weights for the extended model as we did for the exact model. Additionally, we use the following weight for consecutive track sessions; $\pi_K = 10$. We also used the same Gurobi parameters and included an additional parameter; $IntegralityFocus = 1$, which forces variables to take exact integer values. This additional parameter was used because we noticed that sometimes "bigM" constraints were violated due to variables

TABLE 3.5: Penalties & Constraints per instance

Instance	Penalties	Conflicts	Similar tracks
N2OR	10	1	1
GECCO19	405	149	12
GECCO20	290	54	15
GECCO21	112	42	11
OR60	1,478	98	15
OR60F	1,382	70	0
OR60F2	2,059	70	0
OR60F3	3,457	70	0

that meant to be zero, instead took non-trivial values. As a result of this side-effect, infeasible solutions were accepted by the solver. The exact model was free of this side-effect and therefore we did not use that parameter.

For the extended model, the objective values have been computed through evaluation functions in order to obtain the objective value of the exact model, because the objective value of the extended model itself does not provide reliable information regarding the quality of the solution. In Table 3.5 we present the penalties along with constraints for each instance, where Penalties indicate the number of penalties, Conflicts indicate the number of all conflict types, and Similar tracks indicates the number of similar tracks. In Table 3.6 we present the results, where Objective indicates the aggregation of penalties caused by violations of soft constraints, Gap indicates the relative gap between the two objective bounds, Time indicates the required time for the solver to terminate in seconds, and N/A indicates the value is not available. An infeasible status for the objective means that the model is infeasible, while unknown indicates that the solver did not find a feasible solution within the time limit of one hour (3,600 seconds). The extended model is significantly larger compared to the exact model in the number of variables and especially in the number of constraints. Note that even though we include attendees' and track chairs' conflicts in the model, we do not have available real data for these constraints.

In Table 3.7, we present in detail the violations of soft constraints for the extended model, where K's indicates incurred weighted penalties of consecutive tracks (other notation as in Table 3.4). The optimal solution was found for the N2OR instance, which had only 1 violation for a Submissions-Sessions request ($w_\epsilon \times \epsilon_s^{(t,su)} = 1 \times 1 = 1$). For GECCO19, the model found the optimal solution, which had 2 violations for Tracks-Sessions ($w_\alpha \times 2 \times \alpha_s^t = 100 \times 2 \times 10,000 = 2,000,000$) and 7 violations regarding consecutive tracks

Table 3.6: Extended Model Results

Instance	Variables	Constraints	Objective	Gap (%)	Time (s)
N2OR	816	671	1	0.00	0.8
GECCO19	33,800	38,950	2,000,070	0.00	57.5
GECCO20	11,536	10,095	7,750	0.00	51.8
GECCO21	9,216	8,541	11,130	0.00	20.5
OR60	77,096	58,016	Infeasible	N/A	4.8
OR60F	67,896	52,718	433	2.50	3,600.0
OR60F2	257,600	152,244	Unknown	N/A	3,600.0
OR60F3	924,416	307,728	Unknown	N/A	3,600.0

Table 3.7: Extended Model Violations

Instance	α's	β's	γ's	δ's	ϵ's	ζ's	K's	Total
N2OR	-	-	-	-	1	-	0	1
GECCO19	2,000,000	0	-	-	0	-	70	2,000,070
GECCO20	0	10	0	-	7,700	-	40	7,750
GECCO21	0	30	-	11,100	0	-	0	11,130
OR60	N/A	N/A	N/A	-	N/A	-	N/A	N/A
OR60F	400	0	0	-	33	-	0	433
OR60F2	N/A	N/A	N/A	-	N/A	-	N/A	N/A
OR60F3	N/A	N/A	N/A	-	N/A	-	N/A	N/A

($\pi_K \times 7 \times K = 10 \times 7 \times 1 = 70$). The model achieved the optimal solution for GECCO20 with the following violations; 1 violation for Tracks-Rooms ($w_\beta \times \beta_r^t = 1 \times 10 = 10$), 4 violations for consecutive tracks ($\pi_K \times 4 \times K = 10 \times 4 \times 1 = 40$), and 14 violations for Submissions-Sessions ($w_\epsilon \times (7 \times \epsilon_s^{(t,su)} + 7 \times \epsilon_s^{(t,su)}) = 100 \times (7 \times 1 + 7 \times 10) = 7,700$). The solution of GECCO21 is also optimal with 3 violations for Tracks-Rooms requests ($w_\beta \times 3 \times \beta_r^t = 1 \times 3 \times 10 = 30$) and 30 violations for Submissions-Timezones requests ($w_\delta \times (21 \times \delta_s^{(t,su)} + 9 \times \delta_s^{(t,su)}) = 100 \times (21 \times 1 + 9 \times 10) = 11,100$). The time limit was reached for OR60F where the model achieved a solution with a 2.5% optimality gap. The solution had 4 violations for Tracks-Sessions requests ($w_\alpha \times 4 \times \alpha_s^t = 1 \times 4 \times 100 = 400$), and 33 violations for Submissions-Sessions requests ($w_\epsilon \times 33 \times \epsilon_s^{(t,su)} = 1 \times 33 \times 1 = 33$). For the remaining instances, OR60F2 and OR60F3, the model reached the time limit without finding any solution.

3.6.3.1 Infeasible Instances

We modified our extended model as in subsection 3.5.1 and tried to solve OR60, however, this time our model was still infeasible due to constraints Eq. (3.13) that prevent scheduling similar tracks in parallel. We first tried to identify which particular constraints to relax from Eq. (3.13), but it was not a straightforward task. Therefore, we had to remove some similar tracks restrictions which decreased the number of similar tracks from 15 to 8. Specifically, we removed six tracks that were labelled as similar with one track, and another pair of tracks. This led to a feasible model which reached the time limit of one hour and returned a solution with an objective value of 143. The solution had 1 violation for Tracks-Sessions requests ($w_\alpha \times \alpha_s^t = 1 \times 100 = 100$), 30 violations for Tracks-Rooms requests ($w_\beta \times 30 \times \beta_r^t = 1 \times 30 \times 1 = 30$), and 13 violations for Submissions-Sessions requests ($w_\epsilon \times 13 \times \epsilon_s^{(t,su)} = 1 \times 13 \times 1 = 13$).

3.7 Conclusion

This work has provided two integer programming models along with a generic approach to address conference scheduling problems. We have shown that our approach generates low-level schedules for conferences in a fully automated manner. Our weighted penalty system allows the exploration of multiple solutions by adjusting the weights of the soft constraints. An easy-to-use spreadsheet template is used to fit the needs of different conferences. Apart from in-person conferences, we have considered timezone constraints in this work which also makes it suitable for scheduling hybrid and online conferences. We have demonstrated the suitability of our mathematical models by testing them on real data from five different conferences and on additional artificial instances. The results have shown the success of the exact model in finding optimal solutions for almost all instances. The extended model also found optimal and near-optimal solutions for some instances, and revealed some limitations due to its increased size and the complexity of some constraints.

The additional constraints in the extended model add much more complexity, but we believe such constraints are essential for conference scheduling. Having many hard constraints brings limitations to our models in terms of feasibility. In addition, some conferences have to schedule same tracks in parallel due to limited number of sessions, such as OR60. Ideally, to achieve a more robust generic approach, we would want to convert most hard constraints into soft so as to explore additional trade-offs and solutions.

However, such a mathematical model would be too slow in terms of computational time. Therefore, in order to overcome such limitations, we suggest the investigation of alternative methods for future research, such as heuristics (Pylyavskyy et al., 2020), to develop an approach to the largest and most complex conference scheduling problems. Lastly, in our future work along with developing heuristics, we will also consider the submissions ordering constraint, which will allow organisers to express preferences regarding the presentation sequence of submissions within their tracks.

Acknowledgements

The work reported in this paper has been supported by the UK Research and Innovation through the Programme Grant EP/V520214/1.

Chapter 4

A Two-phase Matheuristic Approach to Conference Scheduling Problems

4.1 Introduction

Academic and professional conferences are formal events that foster the advancement of knowledge and the dissemination of innovation across various disciplines. Researchers and innovators are benefited from such gatherings as they have the opportunity to share their latest findings, exchange ideas and critical feedback, as well as collaborate and network with other peers from different institutions and backgrounds. In addition, these events usually include keynote speeches, tutorials, and workshops that inspire innovative thinking and promote interdisciplinary learning to its participants. However, providing these benefits to participants depends heavily on the conference schedule, which poses a challenging task for the conference organisers. More often than not, a different group of organisers is appointed each time to create the conference schedule. Due to various reasons, such as limited time and lack of experience, the organisers typically schedule the conference manually, which is an arduous and often error-prone process done under time pressure. Therefore, as mentioned in Sampson and Weiss (1996) and Sampson (2004), organisers are usually satisfied with a feasible conference schedule regardless of its quality. Additionally, sometimes the schedule requires last-minute changes, after being already published, resulting in an overwhelming experience overall.

In this paper, we address the conference scheduling problem (CSP) as described in Pylyavskyy et al. (2024a). Despite the fact that the CSP was introduced several decades ago by Eglese and Rand (1987), it has not been studied as much as its related problems, namely the class and exam scheduling problems (Sampson, 2004). While some recent studies have focused on clustering approaches to group submissions into tracks (Bulhões et al., 2022; Gündoğan and Kaya, 2022), other studies are software related that only aid organisers during the scheduling process (Kim et al., 2013; Bhardwaj et al., 2014). However, due to conferences having different requirements and objectives, most studies have provided solutions to address rather specific requirements of the conferences being studied per se.

The paper of Pylyavskyy et al. (2024a) is extended in this paper where we present an alternative solution approach to CSPs. Even though the Integer Programming (IP) models developed by Pylyavskyy et al. (2024a) are efficient for some instances, they have certain limitations. The authors assumed that certain requirements are always considered as hard constraints, however, this is not always the case and should be decided by conference organisers. In addition, it would be beneficial to have the flexibility of switching any constraint from hard to soft and vice versa, but this is not a straightforward task in mathematical programming due to some constraints being highly complex. Furthermore, the extended IP model which includes additional constraints failed to return a solution within the one hour time limit for certain instances. Lastly, an extensive amount of time is required to build the IP models for large instances. To this end, we present a robust alternative solution approach in section 4.4 that overcomes these limitations and maintains the generic approach to CSPs.

The remainder of this paper is structured as follows; in section 4.2 the CSP literature review is provided, followed by section 4.3 in which we describe the problem. Next, our decomposed matheuristic approach is presented in section 4.4. Then, we present computation results against the extended IP model in section 4.5, followed by the conclusion in section 4.6.

4.2 Literature review

According to Thompson (2002), a CSP is either approached by a Presenter-Based Perspective (PBP) or an Attender-Based Perspective (ABP) approach. The aim of the former approach is the scheduling of a conference by considering only preferences of

presenters and maximising their satisfaction. On the other hand, the latter approach aims to maximise the satisfaction of attendees. However, some studies have adopted a mixed approach by considering both presenters' and attendees' satisfaction. A detailed literature review of CSPs is provided in Pylyavskyy et al. (2024a), and in the following paragraphs, we discuss additional recent publications. We also classify previous papers based on their followed approach in Table 4.1, and summarise the requirements considered in the literature in Table 4.2. Note that the terminology used in this paper is as described in Pylyavskyy et al. (2024a).

Table 4.1: Classification of publications based on approach taken

ABP	PBP	Both
Eglese and Rand (1987)	Potthoff and Munger (2003)	Nicholls (2007)
Sampson and Weiss (1995)	Potthoff and Brams (2007)	Stidsen et al. (2018)
Le Page (1996)	Edis and Edis (2013)	Vangerven et al. (2018)
Sampson (2004)		Patro et al. (2022)
Zulkipli et al. (2013)		Riquelme et al. (2022)
Quesnelle and Steffy (2015)		Pylyavskyy et al. (2024a)
Manda et al. (2019)		
Rezaeinia et al. (2024)		

Patro et al. (2022) defined the *Virtual Conference Scheduling* (VCS) problem in which the goal is to schedule submissions into time slots by maximising the efficiency and fairness objectives. While the former objective maximises the total attendance in the conference, the latter objective maximises the attendee satisfaction, which depends on their interest in a specific submission and their availability to attend it, and the speaker satisfaction which depends on whether their submission is scheduled in a time slot that provides high attending availability for the interested attendees. The authors consider the preferences of attendees to attend certain submissions as well as their availability based on time zone information. They presented an IP formulation suitable for small conferences and a rounding heuristic along with a clustering approach for larger conferences. Their methods successfully generated balanced conference schedules in terms of efficiency and fairness when tested on real and artificial datasets. However, it should be noted that the VCS problem is restricted to a single track conference scenario without parallel sessions, and the speaker satisfaction metric does not consider the availability of the speaker to present based on time zone information.

A case study regarding the scheduling of GECCO 2019 was presented in Riquelme et al. (2022). The authors defined the problem as a *Track-Based CSP* which requires the

assignment of tracks into sessions and rooms, and the assignment of submissions into time slots of sessions in such a way that the number of missing seats is minimised and certain hard constraints are satisfied. They also presented an instance generator which they used to generate 45 artificial instances with similar characteristics to GECCO 2019. In their study, the authors developed an IP model and a simulated annealing metaheuristic, and evaluated the performance of the two methods by solving the artificial instances. Computational results showed that the IP model managed to optimally solve most instances but failed to solve several instances within the one hour time limit. On the other hand, the simulated annealing metaheuristic replicated the results of the IP model and found decent solutions for the unsolved instances within a short amount of time.

Rezaeinia et al. (2024) tackled the CSP as described in Vangerven et al. (2018) and proposed three optimisation approaches to support the scheduling of LOGMS, INFORMS TSL Workshop and ICSP. Attendees' preferences were collected through an online survey which the authors used to form the objective function, a utility function, of the IP formulations. The authors developed two optimisation approaches assuming that submissions have been already grouped into tracks, namely a single integrated model and a two-model decomposition approach. The former approach maximises the utility function by optimising both high and low level schedules simultaneously, whereas the latter approach initially generates a high level schedule by maximising the utility function on a session level and, then, it generates the low level schedule, based on the previously obtained solution, by maximising the utility function on a time slot level. In the third optimisation approach, which is a relaxed model, they allow submissions to be freely scheduled regardless of their assigned track. Following different experimentation scenarios, the results showed that the single integrated model is impractical in terms of computational time and the relaxed model produced schedules where some tracks contained submissions which were too disconnected from each other based on feedback from the organising committee. The two-model decomposition approach proved to be the most effective approach, finding high-quality solutions within a short amount of time and resulting in its adoption to schedule the three conferences reported.

Pylyavskyy et al. (2024a) presented a generic approach to schedule conferences, using data from Kheiri et al. (2023). They designed a penalty system to accommodate scheduling preferences combined with a weighted sum method to form the objective function, which minimises the penalties associated with tracks and submissions. In addition, the

authors provided a spreadsheet template that is easy to customise to fit different conference data. Two IP models, an exact and an extended model, were developed and tested on both real and artificial instances. The exact model performed well for most instances considering multiple basic constraints involved in CSPs. While the extended model had a decent performance for some instances, it struggled to provide a sufficient performance for others. To the best of our knowledge, this was the first paper that presented a generic framework to tackle CSPs, including time zones which makes it suitable for in-person, hybrid, and online conferences.

In Table 4.2, we present conference requirements, which we describe in detail in section 4.3, that have been considered in previous studies. As seen in the table, the most studied requirements are presenters' conflicts, attendees' conflicts, and room capacities. On the other hand, requirements such as consecutive tracks, similar tracks, session hopping, and rooms preferences have been largely overlooked by most studies. Some of these studies have included requirements which we do not consider in this paper. One of these is the collection of ranked preferences from attendees, instead we allow for attendees to declare their preferred submissions without ranking. Another requirement is the scheduling of submissions among sessions in a balanced manner and the consideration of time zone information for attendees who are not speakers, which we plan to include in our solution approach in the future. Lastly, the EURO conference, as described in the paper of Stidsen et al. (2018), includes requirements related to areas and buildings. In our paper we do not consider such requirements because the EURO conference is unusually large and follows an unusual hierarchical structure, which makes it rather unique compared to typical conferences and, thus, requires a very specific solution approach.

In this paper, we consider all the requirements from Table 4.2 and provide an alternative solution approach to address certain limitations of the mathematical models developed in Pylyavskyy et al. (2024a). First of all, we present a solution approach that easily converts any hard constraint into soft and vice versa by simply changing the weights values. Secondly, our solution approach can handle constraints that need to be resolved on a time slot level, not only session level. Thirdly, the assignment of submissions into rooms and time slots of sessions is determined by the proposed matheuristic, whereas Pylyavskyy et al. (2024a) used a post-processing algorithm to accomplish this task. Additionally, we show that our solution approach achieves near optimal solutions within a short amount of time and provides solutions for large instances where the previously developed extended model fails. Lastly, the proposed matheuristic provides fast solutions,

TABLE 4.2: Requirements of conferences considered in the literature

Requirement	Eglese and Rand (1987)	Sampson and Weiss (1995)	Thompson (2002)	Potthoff and Munger (2003)	Sampson (2004)	Potthoff and Brams (2007)	Nicholls (2007)	Zulkipli et al. (2013)	Edis and Edis (2013)	Quesnelle and Steffy (2015)	Stidsen et al. (2018)	Vangerven et al. (2018)	Manda et al. (2019)	Patro et al. (2022)	Riquelme et al. (2022)	Rezaeinia et al. (2024)	Pylyavskyy et al. (2024a)
Speakers' conflicts			✓	✓	✓	✓	✓		✓	✓			✓		✓		✓
Speakers' preferences						✓	✓		✓	✓		✓	✓		✓		✓
Rooms preferences	✓									✓							✓
Attendees' conflicts	✓	✓	✓	✓	✓	✓	✓			✓		✓				✓	✓
Rooms capacities		✓	✓		✓		✓	✓		✓	✓				✓		✓
Similar tracks													✓				✓
Parallel tracks				✓		✓			✓		✓				✓		✓
Session hopping											✓	✓					✓
Track chairs' conflicts	✓	✓		✓		✓	✓								✓		✓
Tracks' scheduling preferences		✓							✓								✓
Rooms unavailability	✓		✓							✓							✓
Consecutive tracks											✓						✓
Speakers' time zones																	✓

while the integrated IP models requires an extensive amount of time only to build the model for some instances.

4.3 Problem description

We follow the problem description of the CSP and apply the weighted penalty system as described in Pylyavskyy et al. (2024a). However, in contrast to Pylyavskyy et al. (2024a) where submissions are assigned into time slots of sessions with a post-processing algorithm, the assignment of submissions into time slots is determined by the proposed solution approach. In addition to this, our solution approach also determines the assignment of submissions into rooms as we do not assume that scheduling the same track in parallel is a hard constraint. Moreover, our proposed solution approach can resolve presenters' conflicts and attendees' conflicts on a time slot level as well, whereas the mathematical programming models of Pylyavskyy et al. (2024a) only resolve these conflicts on a session level.

The conference requirements are classified as described in Thompson (2002), and we introduce an additional class dedicated to functional requirements. The requirements are categorised into the following three classes; *PBP*, *ABP*, and *Functional* requirements, which we describe in detail next.

1. *PBP:* This class includes requirements that focus on maximising the satisfaction of presenters attending a conference.
    - *Presenters' conflicts:* This is a hard requirement which implies that a presenter cannot be present in two different places at the same time. In other words, many conferences allow authors to submit multiple research papers and, therefore, such submissions must not be scheduled in parallel. Usually, presenters' conflicts are resolved on session level, however, sometimes it might be preferred to resolve those on time slot level.
    - *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.
    - *Presenters' time zones:* On the occasion of an online or hybrid conference, presenters may request the consideration of time zone differences upon scheduling.
    - *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.
2. *ABP:* Requirements in this class aim to maximise the satisfaction of attendees attending a conference.
    - *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, where the attendee will encounter the dilemma of which one to attend. Note that attendees' conflicts may be resolved either on a session or a time slot level.
    - *Rooms capacities:* Conference organisers, who know information about the expected attendance of tracks, may request the scheduling of tracks with high expected attendance into large rooms and vice versa. Overcrowded rooms would cause an unpleasant experience to attendees.

- *Similar tracks:* Sometimes, conferences have a number of tracks which are similar with the potential of attracting the interest of the same audience. Therefore, organisers may request that similar tracks are not scheduled in parallel to prevent attendees from missing their favourite presentations.
- *Parallel tracks:* It is usually preferred to avoid scheduling the same track in parallel. Otherwise, it is likely that attendees will miss their favourite presentations.
- *Session hopping:* Having a track scheduled in multiple rooms would cause inconvenience to attendees who want to attend the whole track as they would have to switch rooms frequently. Therefore, it is preferable to minimise the number of rooms that each track utilises.

3. *Functional:* In this class, we include requirements that are somewhat irrelevant to presenters and attendees, but they allow the accommodation of potential requests from the conference board or organisers and address other potential issues instead.

    - *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.
    - *Tracks' scheduling preferences:* The conference board or organisers may request for various reasons either to schedule (or not) a track into specific sessions.
    - *Rooms unavailability:* Sometimes, certain rooms might be unavailable for utilisation during certain sessions. Therefore, organisers may declare some rooms as unavailable for some sessions.
    - *Consecutive tracks:* It is usually preferred to have tracks scheduled in a consecutive manner.

We acknowledge that not all of the above requirements might be of interest to conferences, and we know, based on literature evidence and based on our experience, that different conferences would prefer some requirements over others. Therefore, each requirement is assigned a weight to reflect its subjective significance. To this end, the objective is the assignment of tracks into sessions and rooms, and the allocation of submissions into sessions, rooms and time slots in such a way that the violations of the weighted requirements are minimised.

TABLE 4.3: Characteristics of the instances

Instance	$\lvert\mathcal{SU}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{TS}\rvert$	Required $\mathcal{TS}$	Available $\mathcal{TS}$
N2OR	35	8	4	4	9	36	36
GECCO19	202	29	13	10	45	215	450
GECCO20	158	24	7	8	28	161	200
GECCO21	138	27	6	8	24	150	192
OR60	329	45	8	23	24	417	540
ISF22	311	49	11	10	36	317	331
OR60F	279	45	8	23	24	353	540
OR60F2	556	72	16	23	49	702	1,115
OR60F3	1,112	72	32	23	105	1,404	2,403

The motivation for this paper originated from scheduling the Genetic and Evolutionary Computation Conference (GECCO), the OR Society's 60th Annual Conference (OR60), the New to OR Conference (N2OR), and the International Symposium on Forecasting (ISF) (Kheiri et al., 2023). We present the characteristics of the instances in Table 4.3, where $|\mathcal{SU}|$ is the number of submissions, $|\mathcal{T}|$ is the number of tracks, $|\mathcal{S}|$ is the number of sessions, $|\mathcal{R}|$ is the number of rooms, $|\mathcal{TS}|$ is the total number of time slots across all sessions, Required $\mathcal{TS}$ is the required number of time slots by all the submissions, and Available $\mathcal{TS}$ is the number of available time slots for scheduling across all the sessions and rooms excluding penalised slots. Note that the number of time slots $|TS|$ is given by summing up the time slots of each session, whereas the number of available time slots, Available TS, is given by $|R| \times |TS|$ and subtracting any penalised time slots. Also, note that OR60F, OR60F2, and OR60F3 are artificial instances.

4.4 Solution approach

Considering the $\mathcal{NP}$-hard complexity of the problem as proved by Quesnelle and Steffy (2015) and Vangerven et al. (2018), we have followed a matheuristic decomposition approach, as described in Archetti and Speranza (2014), to solve the CSP in two phases. In the first phase, we create the high-level conference schedule by considering only the requirements with regard to tracks. The goal of the first phase is to schedule tracks into sessions and rooms by minimising the weighted penalties (i.e., rooms capacities, similar tracks, tracks' scheduling preferences, etc.) and by satisfying as many requirements as possible associated with tracks (e.g., parallel tracks, session hopping, etc.). Based on this solution, in phase two, we create the low-level conference schedule where submissions are allocated into sessions, rooms and time slots, and we further optimise both levels of

the schedule by minimising the violations of the weighted requirements associated with both tracks and submissions. Each phase is described in detail.

4.4.1 Phase one: creation of a high-level conference schedule

An integer programming model has been developed to schedule tracks into sessions and rooms, from which we obtain an optimised high-level schedule. We first present an overview of the notation, followed by the formal formulation of our model.

Sets and indices

The following sets and indices are used in our formulation:

$t \in \mathcal{T}$: The set of tracks

$su \in \mathcal{SU}_t$: The subset of submissions with multiple time slots belonging to track t

$(t, su) \in \mathcal{TSU}$: The set of submissions with multiple time slots

where $\{(t, su) : t \in \mathcal{T} \text{ and } su \in \mathcal{SU}_t\}$

$ts \in \mathcal{TS}_s$: The subset of time slots belonging to session s

$r \in \mathcal{R}$: The set of rooms

$s \in \mathcal{S}$: The set of sessions

$c \in \mathcal{C}$: The set of track chairs

$t \in \mathcal{T}^c$: The set of tracks chaired by the same track chair

$t \in \mathcal{T}^{t'}$: The set of tracks that are similar to track t'

Parameters

The following parameters are used in our formulation:

α_s^t : Penalty for scheduling track t into session s (tracks' scheduling preferences)

w_α : Weight of penalty α_s^t

β_r^t : Penalty for scheduling track t into room r (rooms capacities)

w_β : Weight of penalty β_r^t

$\gamma_{s,r}$: Penalty for utilising room r within session s (rooms unavailability)

w_γ : Weight of penalty $\gamma_{s,r}$

$\lambda^{t,t'}$: Penalty for scheduling tracks t and t' in parallel

w_η : Weight of violation η_s^t

w_θ : Weight of violation θ^t

w_ι : Weight of violation ι_s^t

$|\mathcal{S}|$: The number of total sessions s available

$|\mathcal{TS}_s|$: The number of time slots ts within session s

$ReqTS_t$: The number of required time slots ts to schedule track t

$n^{(t,su)}$: The number of required time slots of submission (t, su)

Decision variables

The following decision variables are used in our formulation:

$Z_{s,r}^t \in \{0,1\}$: 1 if track t is scheduled in session s and room r; 0 if not

$Y_r^t \in \{0,1\}$: 1 if track t is assigned room r; 0 if not

$X_{s,r}^{(t,su)} \in \{0,1\}$: 1 if submission with multiple time slots (t, su)
is scheduled in session s and room r; 0 if not

Other variables

The following variables are used in our formulation to relax certain constraints:

$\eta_s^t \in \mathbb{Z}^+$: Violation amount for constraint similar tracks

$\theta^t \in \mathbb{Z}^+$: Violation amount for constraint session hopping

$\iota_s^t \in \mathbb{Z}^+$: Violation amount for constraint parallel tracks

Constraints

The following constraints are included in our formulation:

$$\sum_{t \in \mathcal{T}} Z_{s,r}^t \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R} \quad (4.1)$$

$$\sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}^c \subset \mathcal{T}} Z_{s,r}^t \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, c \in \mathcal{C} \quad (4.2)$$

$$\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} |\mathcal{TS}_s| \times Z_{s,r}^t \geq ReqTS_t \qquad \forall\, t \in \mathcal{T} \quad (4.3)$$

$$\sum_{r \in R} Z_{s,r}^t + \sum_{r \in R} Z_{s,r}^{t'} - \eta_s^t - \iota_s^t - \iota_s^{t'} \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, t \in \mathcal{T}, \forall\, t' \in \mathcal{T}^t \quad (4.4)$$

$$\sum_{r \in \mathcal{R}} Z_{s,r}^t - \iota_s^t \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, t \in \mathcal{T} \quad (4.5)$$

$$\sum_{r \in \mathcal{R}} Y_r^t - \theta^t = 1 \qquad \forall\, t \in \mathcal{T} \quad (4.6)$$

$$\sum_{s \in \mathcal{S}} Z_{s,r}^t - |\mathcal{S}| \times Y_r^t \leq 0 \qquad \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T} \quad (4.7)$$

$$\sum_{su \in \mathcal{SU}_t} n^{(t,su)} \times X_{s,r}^{(t,su)} - |\mathcal{TS}_s| \times Z_{s,r}^{t} \leq 0 \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T} \quad (4.8)$$

$$\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} X_{s,r}^{(t,su)} = 1 \qquad \forall\, (t, su) \in \mathcal{TSU} \quad (4.9)$$

$$Z_{s,r}^{t} \in \{0, 1\} \qquad \forall\, t \in \mathcal{T}, \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R} \quad (4.10)$$

$$Y_{r}^{t} \in \{0, 1\} \qquad \forall\, t \in \mathcal{T}, \forall\, r \in \mathcal{R} \quad (4.11)$$

$$X_{s,r}^{(t,su)} \in \{0, 1\} \qquad \forall\, (t, su) \in \mathcal{TSU}, \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R} \quad (4.12)$$

Constraint (4.1) allows at most one track to be scheduled for each session-room pair. Constraint (4.2) resolves track chairs' conflicts, where $t \in \mathcal{T}^c \subset \mathcal{T}$ is a set of tracks including a track chair conflict such that $|\mathcal{T}^c| > 1$. Constraint (4.3) ensures that each track is assigned a sufficient number of sessions to be completely scheduled, where $|\mathcal{TS}_s|$ indicates the number of available time slots in session s, and $ReqTS_t$ indicates the amount of time slots required by each track t. Constraint (4.4) is a relaxed constraint which prevents similar tracks to be scheduled in parallel, where $\mathcal{T}^t \subset \mathcal{T}$ is a set of similar tracks such that $|\mathcal{T}^t| > 1$, and η_s^t indicates the violated amount. Note that we include ι_s^t and $\iota_s^{t'}$ in constraint (4.4) because in some cases the same track may be scheduled in parallel. Constraint (4.5) is also a relaxed constraint which prevents the same track to be scheduled in parallel, where ι_s^t is the violated amount of constraint. Constraint (4.6) is another relaxed constraint which assigns each track into one room to prevent session hopping, where θ^t indicates the violated amount of constraint. Constraint (4.7) is a "bigM" constraint which allows the allocation of tracks only into their assigned room (or rooms if $\theta^t \neq 0$), where $|\mathcal{S}|$ is the total number of sessions available at the conference. We use another "bigM" set of constraints (4.8) to allocate submissions with multiple time slots into sessions and rooms, where $|\mathcal{TS}_s|$ is the number of available time slots corresponding to session $s \in \mathcal{S}$, and $n^{(t,su)}$ is the total number of time slots that a given submission requires. Constraints (4.9) ensures that all submissions with multiple time slots must be scheduled into exactly one session and room. Lastly, constraints (4.10), (4.11), and (4.12) indicate that our decision variables $Z_{s,r}^t$, Y_r^t, and $X_{s,r}^{(t,su)}$ are binary.

Objective

As defined previously, $Z^t_{s,r}$ is a binary decision variable which schedules tracks into sessions, where track $t \in \mathcal{T}$, session $s \in \mathcal{S}$, and room $r \in \mathcal{R}$. For instance, when $Z^{Fin}_{MM,1} = 1$ then track "Fin" (Finance) is scheduled into session "MM" (Monday-Morning) and room "1". We use a weighted sum of penalties α^t_s, β^t_r, and $\gamma_{s,r}$ as the coefficient of $Z^t_{s,r}$. When a specific track is scheduled into a non-preferred session, a penalty α^t_s (tracks' scheduling preferences) is incurred weighted by w_α. A penalty β^t_r incurs when a track is scheduled into a room for which the capacity is violated, weighted by w_β. Similarly, when a track is scheduled into a room that is unavailable during a specified session, a penalty $\gamma_{s,r}$ (rooms unavailability) is incurred weighted by w_γ.

Recall that $X^{(t,su)}_{s,r}$ is also a binary decision variable that allocates submissions with multiple time slots into sessions and rooms, where track $t \in \mathcal{T}$, submission with multiple time slots $su \in \mathcal{SU}_t$, session $s \in \mathcal{S}$, and room $r \in \mathcal{R}$. For instance, when $X^{(Fin,F1)}_{MM,1} = 1$ then submission "F1" from track "Fin" is allocated into session "MM" and room "1".

Additionally, recall that η^t_s, θ^t, and ι^t_s are positive integer variables which indicate the violated amount of similar tracks, session hopping, and parallel tracks constraints respectively. These violations are weighted by w_η, w_θ, and w_ι respectively.

To this end, the objective function is formulated as follows:

$$\begin{aligned}\min \quad & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})Z^t_{s,r} + \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}}\sum_{t'\in\mathcal{T}^t} w_\eta\lambda^{t,t'}\eta^t_s \\ & + \sum_{t\in\mathcal{T}} w_\theta\theta^t + \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} w_\iota\iota^t_s \end{aligned} \tag{4.13}$$

The objective function (4.13) schedules tracks and submissions with multiple time slots into sessions and rooms by minimising the weighted violations of constraints and the weighted penalties with regard to tracks.

4.4.2 Phase two: creation of a low-level conference schedule and further optimisation

In phase two, we create the low-level schedule by randomly assigning the submissions into sessions, rooms and time slots based on the high-level schedule obtained from phase one. However, this only results in a sub-optimal solution due to our decomposition approach

and, thus, we further optimise the complete schedule with a heuristic approach, which we describe next.

4.4.3 Selection perturbative hyper-heuristic framework

Hyper-heuristics are high-level heuristic methodologies that provide solutions to computationally hard problems. Among other benefits, their framework notably offers problem domain independence, learning mechanisms applicability, enhanced exploration of the solution space, and identification of an efficient problem-solving method. In contrast to other customised methods, hyper-heuristics do not require expert knowledge of the domain of the problem (Chakhlevitch and Cowling, 2008). Instead of finding a sufficient solution by searching the space of solutions, hyper-heuristics identify an effective method of solving a problem by searching the space of low-level heuristics (Chakhlevitch and Cowling, 2008; Drake et al., 2012b). Hyper-heuristics are classified into two broad categories, namely *generation* and *selection* hyper-heuristics. While the former method generates new heuristics, the latter selects a low-level heuristic from a predefined set and applies it to the solution. In addition to this, hyper-heuristics are further divided into the following two subcategories; 1) *constructive* methods which construct a solution from scratch by implementing a set of heuristics at different phases of the construction process and 2) *perturbative* methods which use a complete initial solution and apply a set of heuristics in a perturbative way to improve this solution. A detailed classification of selection hyper-heuristics can be found in the paper of Drake et al. (2020).

As shown in algorithm 1, we have developed a selection perturbative hyper-heuristic with a "repair" mechanism to further optimise the complete conference schedule. The algorithm uses two parameters (q_t and q_i) that are needed to activate the "repair" mechanism which we describe next. In each iteration, the algorithm randomly selects a low-level heuristic and applies it to the current solution. Then, the new solution is accepted if it is better or equal to the previous solution. Otherwise, it is rejected and the q variable, which counts the number of rejections, is increased by one. When q variable is equal to q_t, which represents the tolerance limit of non-improvements, the repair approach is activated for a duration of q_i iterations. In each iteration of the "repair" mechanism we scan the solution to identify the three most violated constraints and randomly select one of these to improve. Then, the indices of the tracks or submissions that violate the selected constraint are identified and one of them is provided at random to a "repair" low-level heuristic in the next stage. Each constraint has one or two

"repair" low-level heuristics candidates for selection as shown in Table 4.4. Next, the index of the track or submission is provided to the selected "repair" low-level heuristic which is applied to the current solution. A new solution is only accepted if it is better or equal to the previous solution. When q_i iterations are reached, the q variable resets to zero, and this iterative process continues until we reach the time limit. We suggest the parameters values to be set to $q_t = 500$ and $q_i = 100$ for which we observed a satisfying performance during preliminary experiments.

Algorithm 1: Hyper-heuristic algorithm

```
Let L represent the set of low-level heuristics
Let RepL represent the set of "repair" low-level heuristics
Let S represent the current solution
Let S_new represent the new solution
Let f represent the objective value of a given solution as defined in Equation 4.14
Let q represent the number of non-improvements
Let q_t represent the tolerance limit of non-improvement
Let q_i represent the duration of the "repair" mechanism
Let V represent a list with the three most violated constraints
Let I represent the index of the track or submission that violates a constraint
repeat
    L_i ← Select(L);
    S_new ← ApplyLowLevelHeuristic(L_i, S);
    if f(S_new) ≤ f(S) then
        S ← S_new;
    else
        q = q + 1;
    end
    if q = q_t then
        repeat
            V ← ScanSolution(S);
            V_i ← Select(V);
            I ← ReturnIndex(V_i, S);
            RepL_i ← Select(RepL);
            S_new ← ApplyRepairLowLevelHeuristic(RepL_i, I, S);
            if f(S_new) ≤ f(S) then
                S ← S_new;
            end
        until q_i iterations;
        q = 0;
    end
until TimeLimit;
```

Low-level heuristics

We utilise two types of low-level heuristics: the first type does not use the "repair" mechanism, while the other type does. The following 5 low-level heuristics do not use the "repair" mechanism:

- **LLH1:** Randomly selects two tracks belonging to different sessions and swaps them. The low-level schedule is adjusted accordingly.
- **LLH2:** Randomly selects two tracks of the same session and swaps their rooms. The low-level schedule is adjusted accordingly.
- **LLH3:** Randomly selects two submissions of the same session, and swaps their time slots. This low-level heuristic only considers submissions of the same length. That is, they require the same number of time slots to be scheduled. This has no impact on the high-level schedule.
- **LLH4:** Randomly selects two submissions belonging to different sessions and swaps them. This low-level heuristic considers submissions of any length. This has no impact on the high-level schedule.
- **LLH5:** Repeats LLH1 for k times, where $2 \leq k \leq 4$.

"Repair" low-level heuristics

The following low-level heuristics use the "repair" mechanism:

- **LLH6:** Swaps a randomly selected track with another given track. The low-level schedule is adjusted accordingly.
- **LLH7:** Swaps the rooms of a randomly selected track and a given track. The low-level schedule is adjusted accordingly.
- **LLH8:** Swaps the time slots of a randomly selected submission and a given submission. This has no impact on the high-level schedule.
- **LLH9:** Swaps the sessions of a randomly selected submission and a given submission. This has no impact on the high-level schedule.

TABLE 4.4: The "repair" low-level heuristics used for each constraint

Constraint	LLH6	LLH7	LLH8	LLH9	LLH10
Speakers' conflicts (Session level)				✓	✓
Speakers' conflicts (Time slot level)			✓	✓	
Speakers' preferences				✓	
Speakers' time zones				✓	
Rooms preferences				✓	
Attendees' conflicts (Session level)				✓	✓
Attendees' conflicts (Time slot level)			✓	✓	
Rooms capacities	✓	✓			
Similar tracks	✓				
Parallel tracks	✓				
Session hopping	✓	✓			
Track chairs' conflicts	✓				
Tracks' scheduling preferences	✓				
Rooms unavailability	✓	✓			
Consecutive tracks	✓				

- **LLH10:** This low-level heuristic has two heuristics embedded and one of them is selected randomly. The first heuristic swaps the submissions of a given session with the submissions of another randomly selected session. This has no impact on the high-level schedule. The second heuristic randomly selects two submissions of a given track and moves them into a free session (a session with available time slots that could fit both submissions), if such session exists. The high-level schedule is adjusted accordingly.

To further clarify the "repair" mechanism implementation an example is provided. Assume the repair approach is activated, the solution is scanned and the three most violated constraints found are presenters' preferences, tracks' scheduling preferences, and rooms capacities. The tracks' scheduling preferences has been randomly selected for improvement and the solution is scanned again to find the indices that violate the tracks' scheduling preferences constraint. Assume four indices were found from which one is selected at random, for example, Track 1 from Session 2 and Room 2. Next, a "repair" low-level heuristic is selected, in this case LLH6 (see Table 4.4). If a constraint has two "repair" low-level heuristics associated with it, then one of them is selected at random.

Next, the index of Track 1 is passed to LLH6 which swaps it with another randomly selected track. That is, LLH6 selects a track from a random session and room, and swaps it with Track 1 from Session 2 and Room 2. This results in a new solution with a new objective. If the new objective is equal or better than the previous objective, then the new solution becomes the current solution. Otherwise, the new solution is scrapped and the previous solution is restored.

Objective function

As we described in section 4.3, we have three different classes of constraints which are weighted based on their subjective significance. Some constraints, such as presenters' conflicts, that are naturally considered as hard constraints can be satisfied by setting a sufficiently greater weight compared to other constraints. To this end, the objective is to minimise the violations of the weighted constraints, as shown in Eq. (4.14):

$$Min \sum_{i=1}^{n} w_{C_i} \times V_{C_i} \tag{4.14}$$

where w_{C_i} indicates the corresponding weight of constraint C_i, V_{C_i} indicates the corresponding violated amount of constraint C_i, and n is the total number of constraints.

4.5 Computational results

In this section, we evaluate the performance of the developed matheuristic on a set of real and artificial instances taken from Kheiri et al. (2023), and we compare our solution approach against the extended IP model developed in Pylyavskyy et al. (2024a) under different parameters. Note that we have relaxed constraints similar tracks, session hopping, and parallel tracks of the extended IP model, as we did for Phase 1, to avoid infeasibility and make the model comparable.

We used an i7-11370H CPU Intel Processor with 8 cores at 3.30GHz with 16.00 GB RAM to generate our results. Our mathematical model and algorithm was developed in Python 3.8.3, and we used Gurobi 9.5.0 in phase one. We present the weight values used for each instance in Table 4.5 and in Table 4.6, where "-" denotes that a requirement is not used. Although we distinguish constraints into hard and soft, it is not a strict rule as some conferences may consider a hard constraint as a soft constraint and vice versa.

For instance, while session hopping was a hard constraint for most instances, this was not the case for ISF22. As shown in Table 4.5, we have set very high weights for these hard constraints.

TABLE 4.5: Weights of hard requirements used per instance by matheuristic

Requirement	N2OR	GECCO19	GECCO20	GECCO21	OR60	ISF22	OR60F	OR60F2	OR60F3
Speakers' conflicts	10^3	10^7	10^7	10^7	10^5	10^4	10^5	10^5	10^5
Similar tracks	10^3	10^7	10^7	10^7	10^5	10^4	10^5	10^5	10^5
Parallel tracks	10^3	10^7	10^7	10^7	10^5	10^4	10^5	10^5	10^5
Session hopping	10^3	10^7	10^7	10^7	10^5	50	10^5	10^5	10^5
Track chairs' conflicts	-	-	-	-	-	10^4	-	-	-
Rooms unavailability	-	-	10	-	1	10^5	1	1	1

TABLE 4.6: Weights of soft requirements used per instance by matheuristic

Requirement	N2OR	GECCO19	GECCO20	GECCO21	OR60	ISF22	OR60F	OR60F2	OR60F3
Speakers' preferences	1	10^2	10^2	10^2	1	10^2	1	1	1
Speakers' time zones	-	-	-	10^2	-	-	-	-	-
Rooms preferences	-	-	-	-	-	-	-	-	-
Attendees' conflicts	-	-	-	-	-	10	-	-	-
Rooms capacities	-	1	1	1	1	10^2	1	1	1
Tracks' scheduling preferences	-	10^2	10^2	10^2	1	10^2	1	1	1
Submissions ordering	-	-	-	-	-	-	-	-	-
Consecutive tracks	10	10	10	10	10	1	10	10	10

We first evaluate the performance of our matheuristic over 30 runs against the extended model, where we set a time limit for both methods based on the size of the instance. The instances have been categorised into small, medium, and large based on the number of submissions. Small instances contain no more than 100 submissions, medium instances contain up to 500 submissions, and large instances contain more than 500 submissions. We set a 3 seconds time limit for small instances, a 100 seconds time limit for medium

TABLE 4.7: The performance of Matheuristic against Extended model over 30 runs

	Extended IP		Matheuristic					
Instance	Objective	Time (s)	Best Obj.	Avg Obj.	Std.	Ph.1 Avg Time (s)	Ph.2 Avg Time (s)	Total Time (s)
N2OR	**1***	0.8	**1***	1	0	0.02	2.98	3.00
GECCO19	**2,000,070***	57.5	2,000,100	2,234,830	503,386	0.57	99.43	100.00
GECCO20	**7,750***	51.8	8,760	17,993	25,137	0.06	99.94	100.00
GECCO21	**11,130***	20.5	11,140	17,478	8,021	0.43	99.57	100.00
ISF22	N/A	100.00	151	10,241	30,523	3.07	96.93	100.00
OR60	N/A	100.00	6,900,745	7,011,070	115,537	21.81	78.19	100.00
OR60F	443	100.00	566	30,660	53,469	2.86	97.14	100.00
OR60F2	N/A	300.00	487	567	188	21.84	278.16	300.00
OR60F3	N/A	300.00	738	17,417	53,067	90.00	210.00	300.00

instances, and a 300 seconds time limit for large instances. The results are presented in Table 4.7, where Objective indicates the aggregation of penalties caused by violations of soft constraints, Time indicates the required time for the solver to terminate, Best Obj. indicates the best objective found, Avg Obj. indicates the average objective value, Std. indicates the standard deviation, Ph.1 Avg Time indicates the average time of phase 1, Ph.2 Avg time indicates the average time of phase 2, and Total Time indicates the total time for the matheuristic to terminate. The optimal objective value is denoted with an asterisk, and N/A indicates that the value is unavailable. We observe that the matheuristic has found near-optimal solutions for GECCO instances and OR60F within the time limit. In addition, it found the optimal solution for N2OR, and found solutions for the instances where the extended model did not return a solution within the time limit.

Next, we compare again the performances of the two methods but we only allow the matheuristic to run once with a time limit that is equal to the time that it takes for the extended model to either terminate or reach the time limit of one hour. Note that we also include the building time of the models for both methods. We present the results in Table 4.8, where t_b indicates the time required to build the model, t_s indicates the required time for the solver to terminate, and t_t indicates the total time required. All times are in seconds (other notation as in Table 4.7). We observe that the extended model managed to find solutions for ISF22 and OR60 compared to Table 4.7. The relative gap between the two objective bounds was 66.7% for ISF22 and 0.7% for OR60. The main benefit of the matheuristic is observed for instances ISF22, OR60F2, and OR60F3 which found a better solution than the extended model for ISF22, and it found solutions for the large instances where the extended model struggles. In addition, the matheuristic found the optimal solution for N2OR at the same time as the extended model. For the remaining instances, we observe that the extended model finds superior solutions, but the performance of the matheuristic is not significantly worse.

TABLE 4.8: The performance of Matheuristic against Extended model over 1 run including model building time

	Extended IP				Matheuristic			
Instance	Objective	t_b	t_s	t_t	Objective	Ph.1 Time	Ph.2 Time	Total Time
N2OR	**1***	0.1	0.8	0.9	**1***	0.0	0.9	0.9
GECCO19	**2,000,070***	152.2	57.5	209.7	2,000,110	1.3	208.4	209.7
GECCO20	**7,750***	5.7	51.8	57.5	9,510	0.4	57.1	57.5
GECCO21	**11,130***	2.2	20.5	22.7	11,240	0.3	22.4	22.7
ISF22	161	152.9	3,600.0	3,752.9	150	6.3	3,746.6	3,752.9
OR60	5,200,592	174.9	3,600.0	3,774.9	6,900,729	22.6	3,722.3	3,774.9
OR60F	443	99.2	3,600.0	3,699.2	527	3.7	3,695.5	3,699.2
OR60F2	N/A	344.5	3,600.0	3,944.5	278	25.2	3,919.3	3,944.5
OR60F3	N/A	1,210.1	3,600.0	4,811.8	677	128.2	4,683.6	4,811.8

4.6 Conclusion

In this paper, we have significantly extended the size and complexity of conferences that can be solved, which were not possible to be optimally solved by Pylyavskyy et al. (2024a), at a negligible loss of optimality, while in practice conference organisers are happy if they at least find a feasible solution. We proposed a decomposed robust matheuristic solution approach that consists of two phases as an alternative solution approach to CSPs. In phase one, an integer programming model is presented to generate an optimised high-level schedule. Then, based on the created high-level schedule, we create the low-level schedule in phase two, where both levels of the schedule are further optimised by a selection perturbative hyper-heuristic algorithm. The performance of our solution approach has been evaluated under different time limits on real and artificial data taken from Kheiri et al. (2023). The results showed that the matheuristic achieved near-optimal solutions and provided solutions within a short amount of time for instances where the integrated mathematical model fails to return a solution within the one hour time limit. In addition, the matheuristic offers fully customised solutions as one can choose the significance of different types of constraints by setting the weights accordingly. A last benefit of the matheuristic is that due to the decomposition approach, a negligible amount of time is required to build the IP used in phase one in comparison to the extensive amount of time required by the integrated mathematical models, especially for large instances. Overall, we showed that the matheuristic can be efficiently used as an alternative solution approach for cases where the mathematical models struggle. Lastly, in the next paragraph we recommend potential directions for future work.

A potential future work could include the exploration of the characteristics that make an

instance harder to solve compared to others. Throughout our work we noticed that the ISF22 instance is much harder to solve exactly in comparison to other instances of similar size. We suspect that the hardness might be caused by the limited number of available time slots, as shown in Table 4.3. However, we are not sure what makes the ISF22 a harder instance and other characteristics could also play a vital role. Another potential direction for future work is the exploration of the CSP with more scheduling freedom. That is, we assume that conference organisers have already categorised submissions into tracks, and decided upon the number of sessions as well as the number of time slots per session. However, in some cases conference organisers could allow certain submissions to be flexible and provide alternative eligible tracks for that submissions. Furthermore, instead of assuming fixed sessions and time slots, conference organisers could allow for changes in the structure of the schedule. In other words, they could provide different options regarding the number of sessions, and a range of minimum and maximum number of time slots for sessions. Lastly, based on our scheduling experience, we are aware that more often than not last-minute changes are required to a schedule that has already been published, which allows very limited alterations on the schedule. Until now, we have tackled such issues with a manual intervention, but ideally such alterations could be handled with optimisation techniques that have been successfully applied to minimal perturbation problems, such as Barták et al. (2003) and Phillips et al. (2017).

Funding

This work was supported by the UK Research and Innovation under Grant EP/V520214/1.

Chapter 5

CoSPLib – A Benchmark Library for Conference Scheduling Problems

5.1 Introduction

Conferences play a crucial role in academia, providing researchers with the opportunity to present their ideas and receive valuable feedback. Additionally, they serve as a platform for networking with colleagues in similar fields, fostering the formation of new collaborations. Researchers benefit from the opportunity to learn about new methods and ideas in their fields. When organising a conference it is therefore important to have a schedule that allows the conference to run smoothly. The conference scheduling problem (CoSP) is a combinatorial optimisation problem that was initially presented by Eglese and Rand (1987). Even though the CoSP relates to well-studied problems, such as class and exam scheduling problems, it has not received much attention from researchers (Sampson and Weiss, 1996; Sampson, 2004; Thompson, 2002; Greenstreet, 2020). The CoSP involves creating both a high-level and a low-level conference timetable. A high-level schedule involves organising tracks into sessions and allocating rooms, whereas a low-level schedule involves assigning submissions to specific time slots within sessions. The problem requires the consideration of both *hard* and *soft* constraints, and the objective aims to minimise the total penalty cost deriving from violations of the soft constraints. The problem was proved to be $\mathcal{NP}$-hard by Quesnelle and Steffy (2015) and Vangerven et al. (2018).

The operations research literature has several studies tackling the CoSP including both exact and heuristic methods. However, it is challenging to compare and evaluate the developed methods due to several reasons. The primary issue is that many conferences have different scheduling requirements, objectives, and constraints, some of which are case-specific. As a result, there are various problem descriptions, objective functions, and developed methods in the literature which depend on the need of the particular conference. Therefore, benchmark data are needed to provide a fair comparison between the developed methods. Moreover, benchmark instances will create a competitive environment for the development of advanced algorithms to solve CoSP, and raise awareness of the problem which is a real-world problem that has not been studied as much as related problems. Last but not least, researchers could contribute to the development of CoSPLib by submitting new instances, constraints, and solving methods.

In this paper, we are describing what we believe to be the first conference scheduling benchmark. We also present a hyper-heuristic algorithm to create conference schedules. The paper has the following structure. Section 5.2 is a literature review. Section 5.3 describes the CoSPLib. Section 5.4 describes the hyper-heuristic method that we used and presents the computational results. Section 5.5 contains some concluding remarks.

5.2 Background

Thompson (2002) distinguished between two perspectives for CoSP, depending on constraints and objectives of the given problem: Presenter-Based Perspective (PBP) and Attender-Based Perspective (ABP). PBP focuses on optimising presenters' preferences, such as scheduling presenters on their preferred day or time, while ABP prioritises attendees' preferences, such as attending preferred sessions without conflicts or space-shortening problems. Thus, the quality of the schedule is subjective and sensitive to the perspective considered. Similarly to other studies, in this paper, we propose a variant that combines both perspectives in a single model.

In Potthoff and Munger (2003), the research focused on scheduling submissions into sessions in a balanced manner using an integer programming model. Potthoff and Brams (2007) refined this model by incorporating presenter availability, which successfully solved a CoSP that included more than 70 submissions. Edis and Edis (2013) addressed a similar CoSP, aiming to minimise similar tracks scheduled concurrently and

balance submissions across sessions, using an integer programming model for a conference with about two hundred submissions. Nicholls (2007) introduced a heuristic algorithm to facilitate organisers in scheduling and tested it on a conference with roughly three hundred submissions and attendees. Stidsen et al. (2018) addressed the CoSP of the EURO2016 Conference, which is one of the largest conferences within the field of operational research. This CoSP involved the assignment of areas into buildings and the allocation of tracks into sessions and rooms. The authors followed a lexicographic approach using a multi-objective mixed-integer programming model that consisted of five ranked objectives to solve the problem. The objectives, in order of priority, aimed to minimise the number of areas scheduled in different buildings, maximise the number of areas scheduled in the same building, minimise the number of rooms utilised by each track, minimise time gaps across scheduled tracks and maximise the residual room capacity. Constraints included room capacity and prevention of the same track running in parallel, however, presenters' conflicts were excluded and room capacity was limited by data scarcity. Their solution approach focuses on the high-level schedule and leaves the management of the low-level schedule to the track organisers. Vangerven et al. (2018) provided a solution to the scheduling of four conferences: MathSport 2013, MAPSP 2015 & 2017, and ORBEL 2017. The primary goal of the CoSP was to maximise attendees' satisfaction by aligning with their preferred submissions. Furthermore, the research aimed to minimise session hopping, resulting in attendees missing parts of their preferred sessions when switching rooms. The third objective included the maximisation of presenters' preferences. The authors successfully optimised the schedules of the four conferences with a hierarchical three-phase approach using integer programming, dynamic programming, and heuristics. Manda et al. (2019) utilised the dataset from Ecology 2013 for testing purposes to optimise the schedule of the Evolution 2014 conference. The aim of the CoSP was to schedule submissions into time slots to maximise consistency within sessions and minimise similarity across parallel sessions. The authors experimented with random, greedy and integer linear programming approaches to create initial solutions, which they optimised using hill climbing and simulated annealing algorithms. Bulhões et al. (2022) studied a different perspective from typical CoSP in which they mainly focused on grouped optimally submissions into tracks based on their similarity. However, they also scheduled submissions into sessions considering constraints such as presenter conflicts and parallel tracks with the purpose of maximising the total benefit and defined the problem as clustering-based CoSP. Three formulations were presented and tested on artificial instances, each of the approaches suitable for different sizes of instances. They also obtained optimal solutions for two real instances which

were adopted by the organisers of the XV Latin American Robotics Symposium and the Brazilian Logic Conference. Patro et al. (2022) defined the virtual CoSP that required maximisation of efficiency and fairness objectives subject to attendees' preferences and their availability based on time zone information. The authors developed an exact and a heuristic method that were tested on both real and artificial datasets. The virtual CoSP was limited to single-track conferences where parallel sessions are ignored. Riquelme et al. (2022) generated artificial instances with similar characteristics to GECCO 2019 and solved them via integer programming and simulated annealing. The problem was defined by the authors as a track-based CoSP requiring the scheduling of tracks into sessions and rooms, and the scheduling of submissions into time slots of sessions with the purpose of minimising the number of missing seats. Rezaeinia et al. (2024) supported the scheduling of LOGMS, INFORMS TSL Workshop and ICSP by developing and testing three different approaches. Although the CoSP tackled was the same as described in Vangerven et al. (2018), the authors experimented by relaxing the problem and allowing submissions to be freely scheduled regardless of their assigned track. However, this resulted in many submissions being scheduled in tracks in which they did not fit well based on feedback from the organising committee. Finally, a generic approach for conference scheduling was presented by Pylyavskyy et al. (2024a) who developed two integer programming models and tested their performance on the instances presented in this study.

5.3 CoSPLib

5.3.1 Problem Description

The CoSP that we consider in this study includes a set of *tracks* (stream, subject area, topic) along with their corresponding *submissions* (e.g., papers, presentations, talks), a set of available *sessions* along with their corresponding *time slots*, and a set of available *rooms*. We assume that submissions are already assigned to their tracks, and we define "submission" as any type of formal event (e.g., research paper, keynote speech, workshop, etc.) that needs to be scheduled. We consider "sessions" to be periods of time between breaks where participants can switch rooms. For each session, a number of available time slots is assigned where submissions are scheduled. That is, a time slot is defined as a fixed amount of time for presentation. Even though time slots have fixed available time, sessions may consist of different number of time slots. While a submission typically

requires one time slot, some submissions (e.g., tutorial) may require additional time slots to be completely scheduled, satisfying the HC_2 constraint as defined in Section 5.3.1.1. The problem requires the scheduling of tracks into sessions and rooms, and the scheduling of submissions into time slots within sessions and rooms with respect to a number of hard and soft constraints. The objective is to generate a complete schedule by minimising the violations of the soft constraints and satisfying the hard constraints.

5.3.1.1 Hard Constraints

Constraints within this class must be satisfied in order to ensure a feasible schedule is achieved.

- HC_1 *Feasibility*: All submissions must be scheduled.
- HC_2 *Extended Submissions*: Submissions requiring more than one time slot must be scheduled within the same session.
- HC_3 *Presenters' Conflicts*: On the occasion of having two or more submissions which belong to the same presenter, such submissions should be either scheduled within the same room of a session or scheduled within different sessions. Note that it is possible to handle presenters' conflicts either on a session or a time slot level.
- HC_4 *Track Chairs Conflicts*: In case of a track chair being responsible for more than one track, such tracks should be scheduled within different sessions.

5.3.1.2 Soft Constraints

This class includes soft constraints which do not have to be necessarily satisfied to achieve a feasible schedule, but the quality of the schedule is determined based on how many and which of these constraints are satisfied.

- SC_1 *Track-Session Preference Matrix*: The aversion of conference organisers to schedule a specified track into a specified session.
- SC_2 *Track-Room Preference Matrix*: The aversion of conference organisers to schedule a specified track into a specified room.

- SC_3 *Session-Room Preference Matrix*: The aversion of conference organisers to utilise a specified room during a specified session. Sometimes, certain rooms might be unavailable during certain sessions.
- SC_4 *Submission-Session Preference Matrix*: The aversion of presenters to present during a specified session.
- SC_5 *Parallel Tracks*: Scheduling the same track in parallel should be avoided.
- SC_6 *Limit the Number of Rooms per Track*: Scheduling the same track into different rooms should be avoided. Having a track scheduled in multiple rooms is inconvenient for the participants as they would have to switch rooms frequently.
- SC_7 *Consecutive Tracks*: It is desirable to schedule tracks consecutively.
- SC_8 *Submission's Order*: Submissions should be scheduled with respect to their specified order.
- SC_9 *Track-Track Preference Matrix*: The aversion of conference organisers to schedule a pair of tracks in parallel. This is because some conferences may have a number of tracks which are similar with the potential of attracting the interest of the same audience.
- SC_{10} *Presenter's Time Zone*: On the occasion of an online or hybrid conference, the time zone of each presenter should be considered to schedule their submission within a suitable session.
- SC_{11} *Submission-Room Preference Matrix*: The aversion of presenters to present within a specified room. 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.
- SC_{12} *Attendees' Conflicts*: On the occasion of having two or more submissions for which an attendee has declared attending preference, such submissions should be either scheduled within the same room of a session or scheduled within different sessions. Note that it is possible to handle attendees' conflicts either on a session or a time slot level.

TABLE 5.1: Parameters sheet

Sessions			Weights	
Local time zone:	GMT+0		Tracks_Sessions—Penalty:	0
Suitable scheduling times			Tracks_Rooms—Penalty:	0
From:	09:30		Sessions_Rooms—Penalty:	0
To:	21:30		Similar Tracks:	1
Less suitable scheduling times			Number of Rooms per Track:	10
From:	07:00		Parallel Tracks:	1
To:	23:00		Consecutive Tracks:	1
Penalty:	1		Submissions_Timezones:	0
Unsuitable scheduling times			Submissions Order:	1000
Penalty:	10		Submissions_Sessions—Penalty:	10
			Submissions_Rooms—Penalty:	0
			Presenters Conflicts:	100000
			Attendees Conflicts:	0
			Chairs Conflicts:	0
			Presenters Conflicts Timeslot Level:	0
			Attendees Conflicts Timeslot Level:	0

5.3.1.3 Data Format Description

Each instance within the benchmark dataset is stored as an Excel file and follows a specific format. The file consists of the following sheets: parameters, submissions, tracks, sessions, rooms, tracks_sessions penalty, tracks_rooms penalty, similar tracks, and sessions_rooms penalty.

The parameters sheet includes time zone and weights settings (see Table 5.1). Under the "Sessions" column it is possible to set the local time zone of the location at which the conference takes place. In addition, it is possible to set the time range that is considered suitable for scheduling submissions of online presenters. For example, if the time of the presenter's location is outside the specified range for their scheduled submission, then a penalty is incurred. The "Weights" column allows the setting of weight values for all the soft constraints.

The submissions sheet contains information and constraints for each submission as shown in Table 5.2. The first column indicates the reference of each submission and the second column shows their assigned track. In the third column the number of time slots that each submission requires is indicated, followed by the fourth column where the scheduling order of each submission is specified (if irrelevant, zero value is used). For example, Sub 1 should be the first scheduled submission in Track 1, and Sub 2 should be scheduled second. The "Presenters" and "Attendees" columns indicate the presenters

Table 5.2: Submissions Sheet

Reference	Track	Required Time Slots	Order	Presenters	Attendees	Session 1	Session 2	Session 3	Room 1	Room 2
Sub 1	Track 1	1	1	P1	A1					
Sub 2	Track 1	1	2	P2	A2	1	10			
Sub 3	Track 2	1	0	P3	A3, A5					
Sub 4	Track 2	2	0	P4, P5	A4					10

Table 5.3: Tracks Sheet

Tracks	Chairs
Track 1	C1, C2
Track 2	C3
Track 3	C4
Track 4	C5

Table 5.4: Sessions Sheet

Session	# Time Slots	Date	Start Time	End Time
Session 1	4	28/07/2021	09:30	10:30
Session 2	2	28/07/2021	14:00	14:30
Session 3	2	29/07/2021	09:30	10:00
Session 4	3	29/07/2021	10:30	11:15

and attendees for each submission respectively from which presenters' and attendees' conflicts are defined. Note that it is possible for submissions to have multiple presenters and attendees. Then, the next number of columns is given by the total number of available sessions and form the submission-session preference matrix. For example, if Sub 2 is scheduled in Session 1 or Session 2, then a penalty of 1 or 10 will be incurred respectively. Similarly, the remaining number of columns is given by the total number of available rooms and form the submission-room preference matrix (e.g., a penalty of 10 will be incurred if Sub 4 is scheduled in Room 2).

The tracks sheet contains track names in the first column, followed by the track chairs in the second column as shown in Table 5.3.

The sessions sheet includes the name of each session in the first column (see Table 5.4). In the second column, the available number of time slots for each session are indicated. For instance, up to four submissions can be scheduled in Session 1, but in Session 2 at most two submissions can be scheduled assuming a duration of 15 minutes per time slot. The remaining columns indicate the date, the start time, and the end time of each session. The rooms sheet simply contains the name of each room in the first column.

Table 5.5: Tracks_Sessions Penalty Sheet

	Session 1	Session 2	Session 3	Session 4
Track 1	1			
Track 2				
Track 3				
Track 4				10

Table 5.6: Tracks_Rooms Penalty Sheet

	Room 1	Room 2	Room 3	Room 4
Track 1				
Track 2	10		10	1
Track 3				
Track 4				

Table 5.7: Tracks_Track Penalty Sheet

	Track 1	Track 2	Track 3	Track 4
Track 1	-	1	10	
Track 2	-	-		
Track 3	-	-	-	
Track 4	-	-	-	-

The tracks_sessions penalty sheet corresponds to the track-session preference matrix. In Table 5.5, an example is provided where a penalty of 1 incurs if Track 1 is scheduled in Session 1, while a penalty of 10 incurs if Track 4 is scheduled in Session 4.

The tracks_rooms penalty sheet is used to form the track-room preference matrix as shown in Table 5.6. In this example a penalty of 10 is incurred if Track 2 is scheduled in either Room 1 or Room 3, while a penalty of 1 is incurred if Track 2 is scheduled in Room 4.

The similar tracks sheet corresponds to the track-track preference matrix. An example is provided in Table 5.7 where a penalty of 1 is incurred if Track 1 is scheduled in parallel with Track 2, while a penalty of 10 is incurred if Track 1 is scheduled in parallel with Track 3.

Lastly, the sessions_rooms penalty sheet forms the session-room preference matrix as shown in Table 5.8. In this example, a penalty of 100 is incurred if any room except for Room 4 is utilised during Session 2.

TABLE 5.8: Sessions_Rooms Penalty Sheet

	Room 1	Room 2	Room 3	Room 4
Session 1				
Session 2	100	100	100	
Session 3				
Session 4				

5.3.2 The Library

Even though several studies have developed both exact and heuristic methods for the CoSP, it is very difficult to compare them as benchmark instances are not available. Most studies report results based on a real-life or artificial CoSP instance. However, the developed methods should be ideally tested on a dataset in order to measure their performance and to enable comparisons. With this in mind, we propose a benchmark dataset that is publicly available to the research community and can be downloaded from `https://github.com/ahmedkheiri/CoSPLib`. The dataset consists of sixteen instances from four conferences, namely the Genetic and Evolutionary Computation Conference (GECCO), the OR Society's 60th Annual Conference (OR60), the New to OR Conference (N2OR), and the International Symposium on Forecasting (ISF). We present the characteristics of each instance in Table 5.9. OR60F, OR60F2 and OR60F3 are artificially generated instances derived from OR60.

5.4 Hyper-Heuristic for CoSP

Hyper-heuristics are general-purpose problem-independent search methodologies that exploit the search space of low-level heuristics to solve computational optimisation problems. They are classified into two main types: *generation* hyper-heuristics which generate new heuristics and *selection* hyper-heuristics which select heuristics. A selection hyper-heuristic is further categorised into either a *constructive* or *perturbative* hyper-heuristic depending on the nature of its search structure. While the former type starts with an empty solution and constructs a complete solution by selecting and applying a low-level heuristic at each step, the latter type requires a complete solution where at each step it selects and applies a low-level heuristic to improve the solution. A detailed classification of selection hyper-heuristics can be found in the study of Drake et al. (2020).

Table 5.9: Benchmark Instances

Instance	Submissions	Tracks	Sessions	Rooms	Timeslots
GECCO2019	202	29	13	10	45
GECCO2020	158	24	7	8	28
GECCO2020 Poster	131	1	2	1	132
GECCO2020 Workshop	131	26	8	10	40
GECCO2021	138	27	6	8	24
GECCO2021 Workshop	203	28	8	10	56
GECCO2022	179	39	7	8	56
GECCO2022 Workshop	138	59	8	10	80
GECCO2023	207	26	6	9	60
GECCO2023 Workshop	233	55	8	9	80
ISF2022	311	49	11	10	36
N2OR	35	8	4	4	9
OR60	329	45	8	23	24
OR60F	279	45	8	23	24
OR60F2	556	72	16	23	49
OR60F3	1112	72	32	23	105

We have developed a selection perturbative hyper-heuristic to generate optimised conference schedules. The framework of our method involves a two-step iterative process during problem-solving as shown in Algorithm 2.

Algorithm 2: Simple Random - Improving or Equal

```
Let O represent the set of operators
Let S represent the current solution
Let S_new represent the new solution
S ← Initialise();
repeat
    O_i ← Select(O);
    S_new ← ApplyOperator(O_i, S);
    if S_new ≤ S then
        S ← S_new;
    end
until TimeLimit;
```

In the first step, a low-level heuristic is selected randomly and is applied to the solution. Then, in the second step, if the modified solution is not worse than the previous, it is accepted. Otherwise, it is rejected and the previous solution is restored.

TABLE 5.10: High-level Schedule Solution Example

Session \ Room	Room 1	Room 2	Room 3
Session 1	Track 1	Track 2	Track 3
Session 2	Track 1	Track 2	Track 5
Session 3	Track 1	Track 4	Track 6

TABLE 5.11: Low-level Schedule Solution Example

Timeslot \ Room	Room 1	Room 2	Room 3
Timeslot 1 (Session 1)	Sub 1	Sub 9	Sub 14
Timeslot 2 (Session 1)	Sub 2	Sub 10	Sub 15
Timeslot 3 (Session 1)	Sub 3	Sub 11	Sub 16
Timeslot 1 (Session 2)	Sub 4	Sub 12	Sub 20
Timeslot 2 (Session 2)	Sub 5	Sub 13	Sub 21
Timeslot 1 (Session 3)	Sub 6	Sub 17	Sub 22
Timeslot 2 (Session 3)	Sub 7	Sub 18	Sub 23
Timeslot 3 (Session 3)	Sub 8	Sub 19	Sub 24

The solution is represented by two matrices which correspond to a high-level and a low-level conference schedule. In the first matrix (high-level), rows represent sessions, columns represent rooms, and elements indicate the scheduled track as shown in Table 5.10.

In the second matrix (low-level), rows represent the time slots for each session, columns represent rooms, and elements show the scheduled submission as shown in Table 5.11.

Our hyper-heuristic uses three low-level heuristics, specifically two swap heuristics and a reverse heuristic. One of the swap heuristics is applied on the high-level schedule and swaps a track with another track randomly. The other swap heuristic is applied on the low-level schedule and swaps two submissions of the same track at random. The reverse heuristic is also applied on the low-level schedule where a randomly selected sequence of submissions of the same track are reversed. To enhance the exploration of solutions, we introduce random shuffling of the current solution at regular intervals during the algorithm's execution. Preliminary experiments showed that shuffling every 600 seconds balances exploration and exploitation.

A feasible solution is achieved by satisfying the hard constraints as described in Section 5.3.1.1, and the quality of the solution is given by evaluating the violations of the weighted soft constraints presented in Section 5.3.1.2. Each of the soft constraints, $SC = \{sc_1, sc_2, \ldots, sc_n\}$, is assigned a weight, $w = \{w_{sc_1}, w_{sc_2}, \ldots, w_{sc_n}\}$, according to

their subjective significance, where n is the total number of constraints. The goal is to minimise the following objective function which is a summation of the weighted soft constraints violations;

$$\min \sum_{n=1}^{n} w_{sc_n} \times V_{sc_n} \tag{5.1}$$

where w_{sc_n} indicates the corresponding weight of constraint SC_n, and V_{sc_n} indicates the corresponding violated amount of constraint SC_n.

An i7-11370H CPU Intel Processor with 8 cores at 3.30GHz with 16.00 GB RAM was used to generate the results and the hyper-heuristic algorithm was developed in Python 3.8.3. Each instance was solved in a single run by the hyper-heuristic within a 1 hour time limit. The weights used and the results are reported in Table 5.12, where "-" denotes that a constraint is not used. Note that, as we are using a heuristic method, the hard constraints are treated as soft constraints but with very high weights to ensure their satisfaction in the final solution. This approach resulted in feasible solutions in all instances, with the only exception being the GECCO20 instance, which resulted in 3 presenter conflicts.

5.5 Conclusion

In this work, we have provided a benchmark dataset, the CoSPLib, for the under-studied conference scheduling problem which consists of sixteen instances from the following conferences: the Operational Research Society's Annual Conference, the New to Operational Research Conference, the Genetic and Evolutionary Computation Conference, and the International Symposium on Forecasting. We believe that this library will facilitate the comparison and evaluation processes of different developed methods tackling CoSPs. We also hope that the research community will contribute to the further development of the CoSPLib by submitting new instances, constraints, and developed methods. This could potentially create competition in terms of developing new scheduling methods and raise awareness of the real-world problem of conference scheduling. We have also presented a hyper-heuristic algorithm which follows a weighted sum approach, and solved all instances from the benchmark dataset. Lastly, we have reported the computational results of our method to allow their comparison with the results of future works.

Acknowledgement: This work was supported by the UK Research and Innovation under Grant EP/V520214/1.

TABLE 5.12: Weights (upper table) and violations (lower table) of soft constraints and objective value per instance

Instance	w_{SC_1}	w_{SC_2}	w_{SC_3}	w_{SC_4}	w_{SC_5}	w_{SC_6}	w_{SC_7}	w_{SC_8}	w_{SC_9}	$w_{SC_{10}}$	$w_{SC_{11}}$	$w_{SC_{12}}$
GECCO19	10^2	1	-	10^2	1	10	1	-	1	-	-	-
GECCO20	10^2	1	10	10^2	1	10	1	10^6	1	-	-	-
GECCO20 Poster	-	-	-	1	-	-	-	-	-	-	-	-
GECCO20 Workshop	10^2	-	-	10^2	1	10^4	10^4	10^4	1	-	-	-
GECCO21	10^2	1	-	10^2	1	10	1	10^6	1	1	-	-
GECCO21 Workshop	10^2	1	-	10^2	1	10	10^4	10^4	1	1	-	-
GECCO22	10^2	1	10	10^2	1	10	1	10^6	1	1	-	-
GECCO22 Workshop	10^2	1	-	10^2	1	10^4	10^4	10^4	1	1	-	-
GECCO23	10^2	1	10	10^2	1	10	1	10^6	1	1	-	-
GECCO23 Workshop	10^2	-	-	10^2	1	10^4	10^4	10^4	1	1	-	-
ISF22	10^2	10^2	10^5	10^2	10^4	50	1	10^2	10^4	-	-	10
N2OR	-	-	-	10	1	10	1	10^3	1	-	-	-
OR60	10^2	1	10	10	1	10	1	10^3	1	-	-	-
OR60F	10^2	1	10	10	1	10	1	10^3	-	-	-	-
OR60F2	10^2	1	10	10	1	10	1	10^3	-	-	-	-
OR60F3	10^2	1	10	10	1	10	1	10^3	-	-	-	-

Instance	V_{SC_1}	V_{SC_2}	V_{SC_3}	V_{SC_4}	V_{SC_5}	V_{SC_6}	V_{SC_7}	V_{SC_8}	V_{SC_9}	$V_{SC_{10}}$	$V_{SC_{11}}$	$V_{SC_{12}}$	Objective
GECCO19	0	20	-	0	3	70	11	-	10	-	-	-	114
GECCO20	0	0	0	4,600	1	20	4	0	40	-	-	-	4,695
GECCO20 Poster	-	-	-	0	-	-	-	-	-	-	-	-	0
GECCO20 Workshop	2,100	-	-	6,000	26	70,000	0	0	0	-	-	-	78,126
GECCO21	0	30	-	0	1	10	1	0	0	111	-	-	153
GECCO21 Workshop	300	250	-	11,100	22	100	0	0	41	240	-	-	12,053
GECCO22	0	0	0	1,900	0	0	3	0	20	446	-	-	2,369
GECCO22 Workshop	3,100	100	-	400	0	0	0	0	20	425	-	-	4,045
GECCO23	0	10,112	0	900	1	10	5	5,000,000	0	58	-	-	5,011,086
GECCO23 Workshop	100	-	-	0	0	0	0	20,000	0	51	-	-	20,151
ISF22	600	100	100,000	0	0	150	13	0	10,000	-	-	0	110,863
N2OR	-	-	-	0	0	0	1	0	0	-	-	-	1
OR60	0	916	0	210	29	260	16	34,000	46	-	-	-	35,477
OR60F	0	811	0	220	13	250	16	9,000	-	-	-	-	10,310
OR60F2	0	1,016	10,000	620	5	670	53	39,000	-	-	-	-	51,364
OR60F3	0	6,232	0	730	30	2,260	71	123,000	-	-	-	-	132,323

Chapter 6

Exact and Hyper-heuristic Methods for Solving the Conference Scheduling Problem

6.1 Introduction

Conferences are crucial events for academic communities, offering numerous benefits to participants such as sharing the latest research, exchanging ideas, receiving feedback, and networking with peers from diverse backgrounds. To fully exploit these benefits, an effective schedule is essential. However, generating such a schedule is challenging due to numerous preferences and constraints. Traditionally, a group of organisers manually schedules conferences, which is a time-consuming and error-prone process often subject to last-minute changes. Previously, achieving any feasible schedule was sufficient (Sampson, 2004). Nevertheless, nowadays the focus has shifted towards optimising the schedule quality.

The conference scheduling problem (CSP) was introduced in Eglese and Rand (1987) and proved to be $\mathcal{NP}$-hard in Quesnelle and Steffy (2015). Even though the CSP was introduced a few decades ago, it has not been studied as much as related problems such as class and exam scheduling (Sampson and Weiss, 1996; Sampson, 2004; Thompson, 2002). There are two primary approaches to CSP based on constraints and objectives: the Presenter-Based Perspective (PBP) and the Attender-Based Perspective (ABP) (Thompson, 2002). The PBP considers specific requests from presenters, such as

presenting on a particular day or time. The ABP focuses on minimising attendee preference violations, ensuring attendees can attend their preferred sessions without conflicts or space shortages (Zulkipli et al., 2013). Some studies adopted a mixed approach, balancing both presenters' and attendees' preferences (Nicholls, 2007).

The main goal of this study is the investigation of different decision support tools for the creation of a generic conference scheduler applicable to many conferences. These tools are freely available at `https://github.com/ahmedkheiri/CoSPLib` and can be used to generate both high and low level optimised conference schedules in an autonomous and fully automated manner. A generic solution approach has been designed to allow the customisation of our scheduler to fit the needs of different conferences.

6.2 Problem Description

To clarify the terminology used in this paper due to the diverse conference terms in CSP literature, we define the following:

- **Submission:** A formal event that requires scheduling at a conference, replacing terms such as paper, presentation, talk, discussion, and panel.
- **Track:** A group of submissions with a similar subject, synonymous with terms such as stream, subject area, and topic.
- **Session:** A specific time period of the conference consisting of multiple time slots.
- **Time Slot:** A fixed, predefined duration available for the presentation of a submission.

In general, a typical CSP involves scheduling tracks into sessions and rooms to form the high-level schedule, and scheduling submissions into sessions, rooms, and time slots to form the low-level schedule, subject to multiple soft and hard constraints. Some studies in the literature generate both high and low-level schedules, whereas others only generate a high-level schedule, requiring organisers to generate the low-level schedule. Due to the diverse constraints and objectives of different conferences, various problem descriptions, objective functions, and methods have been developed to meet specific needs. As a result, different mathematical models and heuristic methods have been designed for particular conferences, and a method effective for one conference might be unsuitable for another.

A spreadsheet file is used to store input data, which follows a specific template with the purpose of providing a generic approach suitable for many conference scheduling problems. Our scheduler contains a pool of constraints to select from and allows weight assignment for each constraint based on their subjective significance. In addition, the scheduler is also suitable for hybrid and online conferences where submissions need to be scheduled in appropriate sessions considering timezone information. When a CSP is solved using the scheduler, an informative solution file is generated which provides insights regarding the solution quality. The decision maker 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.

6.3 Methodology

Different decision support tools, including integer programming, heuristics, and matheuristics are investigated to build the conference scheduler. All these developed optimisation methods are included in the conference scheduler allowing the decision-maker to select which one they wish to use as some methods may perform better than others depending on the given CSP.

The first tool extends the integer programming model described in Pylyavskyy et al. (2024a). In Pylyavskyy et al. (2024a), two integer programming models were developed to generate high and low-level schedules in a fully automated manner. The results on real data from five different conferences and on additional artificial instances demonstrated the success of the exact models in finding optimal solutions for almost all instances. The second tool is described in Pylyavskyy et al. (2024b) which is a 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, in phase two, the low-level schedule is created where submissions are allocated into sessions, rooms, and time slots. Then, a selection perturbative hyper-heuristic is used to further optimise both levels of the schedule. This solution approach was compared against an integrated mathematical model under different time limits on a set of real and artificial instances. The results showed that the matheuristic finds near-optimal solutions and finds solutions for instances where the mathematical model fails to provide solutions within the one hour time limit. The third tool is a selection hyper-heuristic algorithm, described in Kheiri et al. (2024). The hyper-heuristic method

was validated on GECCO 2019 data and has been used to generate effective schedules for GECCO conferences from 2020 onwards.

In this study, we present the required modifications in the formulations presented in Pylyavskyy et al. (2024a) to obtain the equivalent mathematical models with time slots and discuss their performance compared to the original mathematical models. We also present an approximation model with a simpler, relaxed objective function which is obtained through transformations and discuss its performance compared to the exact model with time slots. Additionally, we compare the performance of all these methods by solving the benchmark instances from Kheiri et al. (2023), and discuss the benefits and limitations of each method.

6.3.1 Mathematical Models with Time Slots for CSPs

Conference organisers may request some submissions to be scheduled in a specified order within their track. This is a constraint that has to be resolved on a time slot level and cannot be handled by formulations presented in Pylyavskyy et al. (2024a). To form the submissions ordering constraint, we first need to introduce a new subset $\mathcal{SU}_t^o$, a new parameter $id_{s,ts}$ and change $X_{s,r}^{(t,su)}$ decision variables to include time slots:

$$
\begin{aligned}
su \in \mathcal{SU}_t^o &: \text{The subset of submissions sorted by their specified scheduling order} \\
&\quad \text{belonging to track } t \\
id_{s,ts} &: \text{The chronological order of time slot } ts \text{ belonging to session } s \\
X_{s,r,ts}^{(t,su)} \in \{0,1\} &: 1 \text{ if submission } (t, su) \text{ is scheduled in session } s, \text{ room } r, \\
&\quad \text{and time slot } ts; \ 0 \text{ if not}
\end{aligned}
$$

With the above changes, we can now form the submissions ordering constraint, Eq. 6.1, which ensures that submissions of a given track are scheduled in the desired specified order.

$$
\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{ts \in \mathcal{TS}_s} id_{s,ts} \times X_{s,r,ts}^{(t,su)} + 1 \leq \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{ts \in \mathcal{TS}_s} id_{s,ts} \times X_{s,r,ts}^{(t,su+1)} \quad \forall\, t \in \mathcal{T}, \forall\, su \in \mathcal{SU}_t^o \setminus \{SU_t^o\} \tag{6.1}
$$

6.3.1.1 Exact Model

To obtain the equivalent exact model of Pylyavskyy et al. (2024a) with time slots, we need to proceed with the following modifications. Firstly, we need to add the set of constraints Eq. 6.2 which ensures that each time slot either gets assigned one submission or remains empty.

$$\sum_{(t,su)\in\mathcal{TSU}} X_{s,r,ts}^{(t,su)} \leq 1 \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, ts \in \mathcal{TS}_s \quad (6.2)$$

Additionally, we modify constraints Eq. 1, Eq. 2, Eq. 6, Eq. 7, and Eq. 10 of Pylyavskyy et al. (2024a) as follows:

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s} X_{s,r,ts}^{(t,su)} = 1 \qquad \forall\, (t,su) \in \mathcal{TSU}$$

$$M_s^p \sum_{ts\in\mathcal{TS}_s} X_{s,r,ts}^{(t,su)} + \sum_{r'\in\mathcal{R}\setminus\{r\}}\sum_{ts'\in\mathcal{TS}_s}\sum_{(t',su')\in\mathcal{TSU}^p} X_{s,r',ts'}^{(t',su')} \leq M_s^p \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, p \in \mathcal{P}, \forall\, (t,su) \in \mathcal{TSU}^p$$

$$\sum_{ts\in\mathcal{TS}_s}\sum_{su\in\mathcal{SU}_t} n^{(t,su)} X_{s,r,ts}^{(t,su)} - |\mathcal{TS}_s| Z_{s,r}^t \leq 0 \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T} \quad (6.3)$$

$$\sum_{ts\in\mathcal{TS}_s}\sum_{su\in\mathcal{SU}_t} X_{s,r,ts}^{(t,su)} - Z_{s,r}^t \geq 0 \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

$$X_{s,r,ts}^{(t,su)} \in \{0,1\} \qquad \forall\, t \in \mathcal{T}, \forall\, su \in \mathcal{SU}_t, \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, ts \in \mathcal{TS}_s$$

Then, we need to change the objective function of the exact model, Eq. 11 of Pylyavskyy et al. (2024a), as follows:

$$\begin{aligned} \min \quad & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}} (w_\alpha \alpha_s^t + w_\beta \beta_r^t + w_\gamma \gamma_{s,r}) Z_{s,r}^t \\ + & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s}\sum_{(t,su)\in\mathcal{TSU}} (w_\delta \delta_s^{(t,su)} + w_\epsilon \epsilon_s^{(t,su)} + w_\zeta \zeta_r^{(t,su)}) X_{s,r,ts}^{(t,su)} \end{aligned} \quad (6.4)$$

6.3.1.2 Extended Model

The equivalent extended model of Pylyavskyy et al. (2024a) with time slots is obtained by modifying constraints Eq. 14 and the objective function Eq. 20 as follows:

$$M_s^a \sum_{ts \in \mathcal{TS}_s} X_{s,r,ts}^{(t,su)} + \sum_{r' \in \mathcal{R} \setminus \{r\}} \sum_{ts' \in \mathcal{TS}_s} \sum_{(t',su') \in \mathcal{TSU}^a} X_{s,r',ts'}^{(t',su')} \leq M_s^a \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, a \in \mathcal{A}, \forall\, (t,su) \in \mathcal{TSU}^a$$

$$\begin{aligned} \min \quad & \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} (w_\alpha \alpha_s^t + w_\beta \beta_r^t + w_\gamma \gamma_{s,r}) Z_{s,r}^t \\ + & \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{ts \in \mathcal{TS}_s} \sum_{(t,su) \in \mathcal{TSU}} (w_\delta \delta_s^{(t,su)} + w_\epsilon \epsilon_s^{(t,su)} + w_\zeta \zeta_r^{(t,su)}) X_{s,r,ts}^{(t,su)} \\ & - \pi_K \times \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} K_{s,r}^t \end{aligned}$$

6.3.1.3 Approximation Model

We can relax the objective function of the exact model, Eq. (6.4), to create an approximation model by replacing $Z_{s,r}^t$ variables with $X_{s,r,ts}^{(t,su)}$ variables through transformations which we present next. To create the objective function of the approximation model, we get Eq. (6.3) and proceed with the following steps:

$$\sum_{ts \in \mathcal{TS}_s} \sum_{su \in \mathcal{SU}_t} n^{(t,su)} X_{s,r,ts}^{(t,su)} - |\mathcal{TS}_s| Z_{s,r}^t \leq 0 \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

$$\sum_{ts \in \mathcal{TS}_s} \sum_{su \in \mathcal{SU}_t} n^{(t,su)} X_{s,r,ts}^{(t,su)} \leq |\mathcal{TS}_s| Z_{s,r}^t \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

$$\sum_{ts \in \mathcal{TS}_s} \sum_{su \in \mathcal{SU}_t} \frac{n^{(t,su)}}{|\mathcal{TS}_s|} X_{s,r,ts}^{(t,su)} \leq Z_{s,r}^t \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

$$\sum_{ts \in \mathcal{TS}_s} \sum_{su \in \mathcal{SU}_t} \frac{n^{(t,su)(w_\alpha \alpha_s^t + w_\beta \beta_r^t + w_\gamma \gamma_{s,r})}}{|\mathcal{TS}_s|} X_{s,r,ts}^{(t,su)} \leq (w_\alpha \alpha_s^t + w_\beta \beta_r^t + w_\gamma \gamma_{s,r}) Z_{s,r}^t \qquad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

We sum over sessions, rooms, and tracks to get the following inequality:

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s}\sum_{t\in\mathcal{T}}\sum_{su\in\mathcal{SU}_t}\frac{X^{(t,su)}_{s,r,ts}}{|\mathcal{TS}_s|}\times n^{(t,su)}(w_\alpha\alpha^t_s+w_\beta\beta^t_r+w_\gamma\gamma_{s,r})\leq\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s+w_\beta\beta^t_r+w_\gamma\gamma_{s,r})Z^t_{s,r}$$

Lastly, we replace $\sum_{t\in\mathcal{T}}\sum_{su\in\mathcal{SU}_t}$ with $\sum_{(t,su)\in\mathcal{TSU}}$ in the left hand side of the inequality:

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s}\sum_{(t,su)\in\mathcal{TSU}}\frac{X^{(t,su)}_{s,r,ts}}{|\mathcal{TS}_s|}\times n^{(t,su)}(w_\alpha\alpha^t_s+w_\beta\beta^t_r+w_\gamma\gamma_{s,r})\leq\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s+w_\beta\beta^t_r+w_\gamma\gamma_{s,r})Z^t_{s,r} \tag{6.5}$$

From Eq. (6.5), we replace the $Z^t_{s,r}$ variables in Eq. (6.4) and obtain the following objective for the approximation model:

$$\min\quad\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s}\sum_{(t,su)\in\mathcal{TSU}}\omega^{(t,su)}_{s,r,ts}X^{(t,su)}_{s,r,ts}$$

where $\omega^{(t,su)}_{s,r,ts}=\frac{n^{(t,su)}(w_\alpha\alpha^t_s+w_\beta\beta^t_r+w_\gamma\gamma_{s,r})}{|\mathcal{TS}_s|}+w_\delta\delta^{(t,su)}_s+w_\epsilon\epsilon^{(t,su)}_s+w_\zeta\zeta^{(t,su)}_r$.

6.4 Computational Results

The results in this section were generated on an i7-11370H CPU Intel Processor with 8 cores at 3.30 GHz with 16.00 GB RAM. We used Python 3.8.3 and Gurobi 9.5.0. We present the computational results in Table 6.1, where Objective indicates the aggregation of penalties caused by violations of soft constraints, Gap indicates the relative gap between the two objective bounds, and Time indicates the required time for the solver to terminate in seconds. N/A indicates the value is not available. We observe that the size of the exact model is significantly larger compared to Table 3 of Pylyavskyy et al. (2024a). The exact model with time slots replicated the results of the exact model in Pylyavskyy et al. (2024a) for some instances but with significantly worse computational times, and was infeasible for the OR60F instance due to the submissions ordering constraints, Eq. 6.1. Additionally, the exact model with time slots failed to find solutions within time limit for instances OR60F2 and OR60F3.

We observe that the extended model replicated the results of the extended model presented in Pylyavskyy et al. (2024a) for N2OR, GECCO20, and GECCO21 instances at worse computational times compared to Table 6 of Pylyavskyy et al. (2024a). It also

TABLE 6.1: Exact, Extended and Approximation Models with Time Slots Results

	Exact Model					Extended Model					Approximation Model			
Instance	Variables	Constraints	Objective	Gap (%)	Time (s)	Variables	Constraints	Objective	Gap (%)	Time (s)	Variables	Constraints	Objective	Time (s)
N2OR	1,420	387	0	0.000	0.3	1,516	691	1	0.000	1.2	1,420	387	0	0.3
GECCO19	94,960	27,361	1,000,010	0.001	3,600.0	98,440	39,400	2,000,080	0.001	3,600.0	94,960	27,361	1,000,020	3,600.0
GECCO20	36,928	6,604	6,110	0.000	420.0	38,080	10,333	7,750	0.000	3,274.6	36,928	6,604	6,110	159.5
GECCO21	28,008	4,993	11,130	0.000	98.0	29,088	9,805	11,130	0.000	292.0	28,008	4,993	11,130	56.8
OR60	190,923	41,540	Infeasible	N/A	6.6	198,168	66,602	Infeasible	N/A	6.0	190,923	41,540	Infeasible	6.6
OR60F	163,323	31,707	Infeasible	N/A	2,289.2	170,568	53,442	Infeasible	N/A	3,600.0	163,323	31,707	Infeasible	2,224.0
OR60F2	654,764	79,023	N/A	N/A	3,600.0	679,604	174,776	N/A	N/A	3,600.0	654,764	79,023	N/A	3,600.0
OR60F3	2,740,128	176,470	N/A	N/A	3,600.0	2,791,464	353,271	N/A	N/A	3,600.0	2,740,128	176,470	N/A	3,600.0

found a slightly worse objective for GECCO19 instance and failed to find solutions for OR60F2 and OR60F3 within the time limit.

Similar to the extended model, the objective values of the approximation model have been computed through evaluation functions in order to present the objective value of the exact model. The results of approximation model compared to the exact model reveal that the objective of the exact model was replicated in all instances except for GECCO19 for which a slightly worse objective was found. Moreover, the approximation model achieved a better computational time for GECCO20 and GECCO21 instances but also failed to find solutions for OR60F2 and OR60F3.

Overall, the presented models with time slots managed to replicate the results of the models presented in Pylyavskyy et al. (2024a) for some instances. The computational times were significantly worse, compared to the models in Pylyavskyy et al. (2024a), due to the increased size of the models, which makes them impractical for larger instances such as OR60F2 and OR60F3. However, they seem to be suitable for conferences that have similar size to N2OR and GECCO which involve constraints that need to be resolved on a time slot level. Lastly, it should be noted that the models presented in Pylyavskyy et al. (2024a) should be preferred over the models with time slots for conference scheduling problems including only constraints that need to be addressed on a session level.

6.4.1 Performance Comparison of Different Methods for CSPs

In this section, we compare the performance of different methods by solving the benchmark instances and using the weights from Kheiri et al. (2023). To make the methods comparable, we considered the extended model with time slots from Section 6.3.1.2 and relaxed certain hard constraints as in Pylyavskyy et al. (2024b). Specifically, the relaxed constraints are the following: submissions ordering, parallel tracks, number of rooms per track, and similar tracks. For the remaining two methods, the matheuristic

TABLE 6.2: The Performance of Integer Programming (IP), Matheuristic (MH), and Hyper-heuristic (HH). Best solutions are highlighted in bold.

Instance	IP	MH	HH
GECCO19	98	**49**	114
GECCO20	5,784	**3,895**	4,695
GECCO20 Poster	0	0	0
GECCO20 Workshop	**72,825**	76,627	78,126
GECCO21	221	**146**	153
GECCO21 Workshop	41,774	**11,519**	12,053
GECCO22	14,011,954	5,681	**2,369**
GECCO22 Workshop	22,974	**1,920**	4,045
GECCO23	21,030,737	**3,001,684**	5,011,086
GECCO23 Workshop	1,081,310	**145**	20,151
ISF22	N/A	**572**	110,863
N2OR	1	1	1
OR60	104,210	43,577	**35,477**
OR60F	34,069	13,719	**10,310**
OR60F2	78,633	**34,267**	51,364
OR60F3	N/A	**83,136**	132,323

and the hyper-heuristic, no modifications were needed. Each instance was solved with a time limit of one hour for each method and the results are presented in Table 6.2.

By observing Table 6.2, we notice that the matheuristic method had the best performance overall finding the best solutions in 10 out of 16 instances. The next best performance was achieved by the hyper-heuristic method which found the best solutions in 3 out of 16 instances, followed by the integer programming method which only found the best solution for the GECCO20 Workshop instance, but it failed to find a solution within the time limit for ISF22 and OR60F3. All methods successfully found the optimal solution for GECCO20 Poster and N2OR instances. In the next paragraphs, some key takeaways from this experiment are discussed including benefits and limitations of each method.

The integer programming method has several benefits compared to the other methods. First of all, it is mostly appropriate for instances involving only constraints that need to be resolved on a session level and for instances where hard constraints can be satisfied (e.g., GECCO20 Poster, N2OR). In this case, the models presented in Pylyavskyy et al. (2024a) can be used which are significantly smaller in size compared to mathematical models with time slots and, thus, much faster to solve. In addition, this method is ideal for small conferences with few constraints and it may achieve proven optimal solutions

given that the model does not need to be relaxed. On the other hand, the integer programming has several limitations too. It is slow for instances involving constraints that need to be resolved on a time slot level and, sometimes, it may fail to return a solution (e.g., ISF22). Another limitation is that for instances where hard constraints cannot be satisfied, the model needs to be relaxed and, hence, the final solution is not guaranteed to be optimal. Lastly, this method is unsuitable for large scale instances due to the increased size of the model (e.g., OR60F3).

The matheuristic and hyper-heuristic methods have common benefits over the mathematical models. Both methods achieve decent solutions and can handle numerous constraints of both types, time slot and session level. Additionally, both methods always return a solution and they are suitable for conferences of all sizes including large scale instances. Moreover, the matheuristic method finds good solutions faster than the hyper-heuristic, but both methods due to the heuristic nature are not guaranteed to find optimal solutions.

Overall, the integer programming method is suitable for small to medium size conferences where hard constraints can be satisfied and need to be resolved on a session level. The matheuristic and the hyper-heuristic methods are suitable for all conferences of any size and for any type of constraints including both time slot and session level. Lastly, the matheuristic method is faster in finding decent solutions, compared to the hyper-heuristic, which allows the exploration of additional alternative solutions within a short amount of time.

6.5 Conclusion

This work is concerned with the optimisation of the conference scheduling problem. In this study, we presented extended formulations of mathematical models which are suitable for constraints that need to be resolved on time slot level. Moreover, we presented an approximation model with a simpler, relaxed objective function which is obtained through transformations, and tested all the mathematical models on a set of real and artificial instances. The mathematical models with time slots managed to replicate the results of the models presented in Pylyavskyy et al. (2024a) for some instances, but at significantly worse computational times due to their increased size. In addition, we compared the performance of three different methods based on exact, matheuristic and hyper-heuristic techniques by solving the benchmark instances (available at Kheiri et al.

(2023)) to explore the benefits and limitations of each method. We observed that the matheuristic had the best performance overall finding better solutions than the other methods in 10 out of 16 instances. Overall, the integer programming method is suitable for certain conferences with specific characteristics, while the matheuristic and the hyper-heuristic methods are suitable for all conferences of any size and for any type of constraints including both time slot and session level. A potential future work could include the exploration of the characteristics that make an instance harder to solve compared to others. For example, ISF22 instance is much harder to solve exactly in comparison to other instances of similar size. This might be caused by the limited number of available time slots, however, other characteristics could also play a vital role.

Acknowledgements

This work was supported by the UK Research and Innovation under Grant EP/V520214/1.

Chapter 7

Conclusion

In this chapter, the contributions of this thesis are summarised in section 7.1, new knowledge and the significance of this research are presented in section 7.2, and potential directions for future work are suggested in section 7.3.

7.1 Summary of Work

This thesis has successfully provided answers to the research questions that were formed in chapter 1. Firstly, the approach followed in this research revealed the feasibility of a generic conference scheduler through the proposal of various modelling formulations considering common objectives and constraints encountered in conference scheduling. Secondly, different operations research techniques were investigated to identify their benefits, drawbacks, and suitability to solve CoSPs, including online and hybrid modes. Thirdly, the code base of the scheduler has been open-sourced, resulting in the first open-source solution for conference scheduling. Lastly, the scheduler has already been used to schedule several conferences, which emphasises the impact of this research and increases the potential for further impact in the future.

In this paragraph, the main contributions to new knowledge and significance of this research are outlined. First of all, this research proposed the first generic conference scheduler that offers a complete solution to many conferences due to its flexible approach. In addition, this is the first work in which hyper-heuristics have been used to solve CoSPs, demonstrating their efficiency in solving challenging problems and their suitability in tackling a wide range of problems. Moreover, this research has proposed a matheuristic

method that achieves near-optimal solutions for CoSPs within a short amount of time. Furthermore, the code base of this research has been open-sourced, making it the first open-source code base for conference scheduling that allows its free usage to optimise schedules for conferences and allows other researchers to contribute to it. Last but not least, the scheduler has already been used to successfully schedule several conferences, highlighting the significance and impact of this research. In the following paragraphs, the contributions per chapter are presented.

In chapter 3, a generic solution approach was presented in which preferences and constraints are considered for both low-level and high-level schedules. Additionally, this was the first work to consider timezone differences when scheduling hybrid and online conferences. We also developed a penalty system to accommodate scheduling preferences and combined it with the weighted sum method to form the objective function. Lastly, we considered submissions with multiple time slots in our solution approach.

Next, in chapter 4, an alternative solution approach was proposed, namely a matheuristic method, to enhance our generic conference scheduler. The proposed method can be used for cases where the integer programming models fail to return a solution within a reasonable amount of time. Moreover, this method provides additional flexibility when handling constraints, allowing easy prioritisation of constraints without requiring any reformulations.

In chapter 5, a hyper-heuristic algorithm was added to the scheduler to overcome the need for a commercial solver, which is usually required by the developed mathematical models for larger CoSPs. In addition, we provided a benchmark dataset along with benchmarking results that were produced by the hyper-heuristic algorithm.

In chapter 6, we presented extended formulations of mathematical models suitable for constraints that need to be resolved on time slot level. Moreover, we presented an approximation model with a simpler, relaxed objective function obtained through transformations. In addition, we provided a performance comparison of the developed methods in this thesis by solving the benchmark instances to explore the benefits and limitations of each method.

Overall, different operations research tools were investigated in this thesis which were used to create a generic conference scheduler applicable to many conferences. The conference scheduler is freely available at `https://github.com/ahmedkheiri/CoSPLib` and is suitable to generate both high and low level optimised conference schedules in an effortless, autonomous, and fully automated manner. A generic solution approach has

been designed to allow the customisation of the scheduler to fit the needs of different conferences, and a spreadsheet file is used to store input data, which follows a specific template with the purpose of providing a generic approach suitable for many conference scheduling problems. The scheduler is also suitable for hybrid and online conferences where submissions need to be scheduled in appropriate sessions considering timezone information. It also allows one to select the constraints they are interested in from an available pool of constraints, and assign a weight for each selected constraint based on their subjective significance. In addition, it is also possible to select the optimisation method from a number of different available optimisation methods that will be applied to the problem at hand. Lastly, upon termination, the scheduler generates an informative solution file which provides insights to decision-makers regarding the solution quality, and allows them to manually edit the solution and observe the impact of their changes on solution quality.

7.2 Lessons learned

In this section, the significance of this research and the newly acquired knowledge, which may be applicable to other problems and future investigations, are further discussed. An essential insight gained is the importance of attempting to formulate exactly the problem at hand, even when it appears intractable. Some researchers bypass this step and directly attempt a heuristic solution. However, an initial attempt to resolve the problem exactly provides a deeper comprehension of the problem and associated complexities. This understanding can facilitate the innovation of more effective solutions. In this research, I not only managed to formulate the problem exactly, but also solved some instances exactly. This process inspired me to explore the decomposition of the problem following a two-phase matheuristic approach, where constraints that are less challenging to satisfy are solved exactly, while the more complex constraints are addressed heuristically. Additionally, I enhanced the hyper-heuristic algorithm by incorporating "repair" low-level heuristics, which focus on refining the most constraint-violated segments of the solution. As a result of these efforts, I successfully developed a matheuristic approach that outperforms the other methods in performance across most tested instances. Another important lesson is to obtain a deep understanding of the problem. Before starting developing solutions, it is significant to identify key constraints, objectives, trade-offs and other specific challenges of a problem. Exploring multiple methods is another key lesson as no single method would be efficient for every problem. Exact methods are powerful,

but when facing computational limits, then heuristics are equally powerful and important. Testing developed methods on real data is crucial too, as it provides validation of the developed solutions and might help in identifying hidden or unknown challenges. Lastly, mathematical models should be preferred as an ideal solution method, however, developing a generic solution approach for a real-world problem requires scalability and flexibility to accommodate diverse user needs, which makes heuristic methods more appropriate.

7.3 Future Work

In the next paragraphs some potential directions for future work are discussed. Even though multiple constraints have been considered in this thesis, there are additional constraints that could be included to the conference scheduler which are described next:

- *Submission's Relative Order*: Submissions should be scheduled with respect to their specified relative order. The relative order does not imply time slot order. The submission order currently considered implies time slot order.
- *Track's Relative Order*: Tracks should be scheduled with respect to their specified relative order. This constraint can facilitate requests where some tracks should not start before some other tracks have been completed.
- *Track's Building*: Schedule a track in one building. In some cases (e.g., EURO), there may be a number of buildings used during the conference. Hence, this constraint could prevent participants interested in a specific track to move from one building to another.
- *Balance*: Submissions should be scheduled in a balanced manner within rooms across sessions. This constraint could prevent some parallel sessions having different number of submissions scheduled (e.g., a parallel session where 4 submissions are scheduled in one room and 2 submissions scheduled in another room).
- *Open Session*: A submission is specified whether it can open a session or not.
- *Close Session*: A submission is specified whether it can close a session or not.
- *Different Session*: Two or more submissions should be scheduled in different sessions.

- *Same Session*: Two or more submissions should be scheduled in the same session.
- *Track's Maximum Number of Days*: The maximum number of days that a track should be scheduled in.
- *Tracks Same Room*: Some tracks should be scheduled in the same room.
- *Tracks Same Building*: Some tracks should be scheduled in the same building.
- *Preferred Number of Time slots*: The preferred number of time slots that should be scheduled in a session.

Another potential future work could include the exploration of the characteristics that make an instance harder to solve compared to others. For example, ISF22 instance is much harder to solve exactly in comparison to other instances of similar size. This might be caused by the limited number of available time slots, however, other characteristics could also play a vital role. Furthermore, it could be worth exploring the conference scheduling problem with more scheduling freedom where conference organisers could allow certain submissions to be flexible and provide alternative eligible tracks for that submission. Additionally, conference organisers could allow for changes in the structure of the schedule by providing different options regarding the number of sessions, and a range of minimum and maximum number of time slots for sessions. Moreover, sometimes last-minute changes are required on a schedule that has already been published, which allows very limited alterations on the schedule. Hence, it could be worth exploring optimisation techniques that have been successfully applied to minimal perturbation problems, such as Barták et al. (2003) and Phillips et al. (2017).

Finally, another potential direction is to explore multi-objective approaches other than the weighted sum approach. Another approach that could be worth exploring is the lexicographic approach, where different objectives are categorised into an order of priority levels. Objectives at lower levels are considered infinitely more important than those at higher levels. Consequently, the algorithm seeks the optimal solution for objectives at lower levels before addressing those at higher levels. In the weighted sum approach, the set of objectives is combined into a single objective by multiplying each objective by a user-supplied weight. This method is widely used, but determining the appropriate weights for each objective can be challenging. Typically, weights are assigned in proportion to the relative importance of each objective in the problem. However, one of the main challenges with the weighted sum method arises in non-convex multi-objective

problems, where certain Pareto-optimal solutions may be missed. Consequently, further research could explore alternative methods that may perform better in such cases.

Bibliography

Leena N. Ahmed, Ender Özcan, and Ahmed Kheiri. Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications*, 42(13):5463 – 5471, 2015.

Alhanof Almutairi, Ender Özcan, Ahmed Kheiri, and Warren G. Jackson. Performance of selection hyper-heuristics on the extended hyflex domains. In Tadeusz Czachórski, Erol Gelenbe, Krzysztof Grochla, and Ricardo Lent, editors, *Computer and Information Sciences*, pages 154–162, Cham, 2016. Springer International Publishing.

Claudia Archetti and M.Grazia Speranza. A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246, 2014. ISSN 2192-4406. doi: https://doi.org/10.1007/s13675-014-0030-7.

Michael O Ball. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16(1):21–38, 2011.

Roman Barták, T Muller, and Hana Rudová. Minimal perturbation problem - a formal view. *Neural Network World*, 13(5):501–512, 2003.

Dimitri P Bertsekas. Convexification procedures and decomposition methods for nonconvex optimization problems. *Journal of Optimization Theory and Applications*, 29 (2):169–197, 1979.

Anant Bhardwaj, Juho Kim, Steven Dow, David Karger, Sam Madden, Rob Miller, and Haoqi Zhang. Attendee-sourcing: exploring the design space of community-informed conference scheduling, 2014.

Alain Billionnet and Éric Soutif. Using a mixed integer programming tool for solving the 0–1 quadratic knapsack problem. *INFORMS Journal on Computing*, 16(2):188–197, 2004.

Teobaldo Bulhões, Rubens Correia, and Anand Subramanian. Conference scheduling: a clustering-based approach. *European Journal of Operational Research*, 297(1):15–26, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.04.042.

Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. *Handbook of metaheuristics*, pages 457–474, 2003.

Edmund K. Burke and Yuri Bykov. A late acceptance strategy in hill-climbing for examination timetabling problems. In *In Proceedings of the conference on the Practice and Theory of Automated Timetabling(PATAT*, 2008.

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, Dec 2013.

Cem Canel and Basheer M Khumawala. A mixed-integer programming approach for the international facilities location problem. *International Journal of Operations & Production Management*, 16(4):49–68, 1996.

Konstantin Chakhlevitch and Peter Cowling. Hyperheuristics: recent developments. In *Adaptive and multilevel metaheuristics*, pages 3–29. Springer, 2008.

Alexander Chernyavsky, Jason J Bramburger, Giovanni Fantuzzi, and David Goluskin. Convex relaxations of integral variational problems: pointwise dual relaxation and sum-of-squares optimization. *SIAM Journal on Optimization*, 33(2):481–512, 2023.

Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming models*. Springer, 2014.

Gianni Di Pillo. Exact penalty methods. *Algorithms for continuous optimization: the state of the art*, pages 209–253, 1994.

John H. Drake, Ender Özcan, and Edmund K. Burke. An improved choice function heuristic selection for cross domain heuristic search. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII*, pages 307–316, Berlin, Heidelberg, 2012a. Springer Berlin Heidelberg.

John H Drake, Ender Özcan, and Edmund K Burke. An improved choice function heuristic selection for cross domain heuristic search. In *International Conference on Parallel Problem Solving from Nature*, pages 307–316. Springer, 2012b.

John H Drake, Ahmed Kheiri, Ender Özcan, and Edmund K Burke. Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405–428, 2020.

Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86 – 92, 1993.

Emrah Edis and Rahime Sancar Edis. An integer programming model for the conference timetabling problem. *Celal Bayar Üniversitesi Fen Bilimleri Dergisi*, 9(2):55–62, 2013.

R. W. Eglese and G. K. Rand. Conference seminar timetabling. *Journal of the Operational Research Society*, 38(7):591–598, 1987.

Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

The Conference Exchange. Confex, 1996. URL `https://confex.com/`.

Patrik Flisberg, Bertil Lidén, and Mikael Rönnqvist. A hybrid method based on linear programming and tabu search for routing of logging trucks. *Computers & Operations Research*, 36(4):1122–1144, 2009.

Christodoulos A Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139:131–162, 2005.

Michael R Garey, David S Johnson, et al. A guide to the theory of np-completeness. *Computers and intractability*, pages 37–79, 1990.

P Greenstreet. A new hyper-heuristic with weight sum approach to conference scheduling problem. Master's thesis, Lancaster University, 2020.

Esra Gündoğan and Mehmet Kaya. Deep learning based conference program organization system from determining articles in session to scheduling. *Information Processing & Management*, 59(6):103107, 2022. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2022.103107.

Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449 – 467, 2001.

Thamarassery Abduljaleel Jessin, Sakthivel Madankumar, and Chandrasekharan Rajendran. Permutation flowshop scheduling to obtain the optimal solution/a lower bound with the makespan objective. *Sādhanā*, 45(1):228, 2020.

David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79 – 100, 1988.

Murat Kalender, Ahmed Kheiri, Ender Özcan, and Edmund K Burke. A greedy gradient-simulated annealing selection hyper-heuristic. *Soft Computing*, 17:2279–2292, 2013.

G. Kendall and M. Mohamad. Channel assignment optimisation using a hyper-heuristic. In *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, volume 2, pages 791–796, Dec 2004.

Graham Kendall and Naimah Mohd Hussin. A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 270–293. Springer, 2004.

JW Kendall. Hard and soft constraints in linear programming. *Omega*, 3(6):709–715, 1975.

Ahmed Kheiri, Ender Özcan, and Andrew J Parkes. Hysst: hyper-heuristic search strategies and timetabling. In *Proceedings of the 9th international conference on the practice and theory of automated timetabling (PATAT '12)*, 2012.

Ahmed Kheiri, Yaroslav Pylyavskyy, and Peter Jacko. Csplib - a benchmark library for conference scheduling problems. Manuscript in preparation, 2023.

Ahmed Kheiri, Yaroslav Pylyavskyy, and Peter Jacko. Efficient scheduling of gecco conferences using hyper-heuristic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '24 Companion, pages 1732–1737, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704956. doi: 10.1145/3638530.3664186.

Juho Kim, Haoqi Zhang, Paul André, Lydia B Chilton, Wendy Mackay, Michel Beaudouin-Lafon, Robert C Miller, and Steven P Dow. Cobi: a community-informed conference scheduling tool. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 173–182, 2013.

Jozef Kratica, Djordje Dugošija, and Aleksandar Savić. A new mixed integer linear programming model for the multi level uncapacitated facility location problem. *Applied Mathematical Modelling*, 38(7-8):2118–2129, 2014.

Y. Le Page. Optimized schedule for large crystallography meetings. *Journal of Applied Crystallography*, 29(3):291–295, 1996.

Claude Lemaréchal. Lagrangian relaxation. *Computational combinatorial optimization: optimal or provably near-optimal solutions*, pages 112–156, 2001.

Igor Litvinchev, Luis Infante, and Lucero Ozuna. Approximate packing: integer programming models, valid inequalities and nesting. *Optimized Packings with Applications*, pages 187–205, 2015.

Andrea Lodi and Michele Monaci. Integer linear programming models for 2-staged two-dimensional knapsack problems. *Mathematical Programming*, 94(2):257–278, 2003.

Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated local search. In *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers, 2002.

Zhi-Quan Luo, Wing-Kin Ma, Anthony Man-Cho So, Yinyu Ye, and Shuzhong Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, 2010.

Bjoern-Elmar Macek, Christoph Scholz, Martin Atzmueller, and Gerd Stumme. Anatomy of a conference. In *Proceedings of the 23rd ACM conference on Hypertext and Social Media*, pages 245–254, 2012.

Prashanti Manda, Alexander Hahn, Katherine Beekman, and Todd J Vision. Avoiding "conflicts of interest": a computational approach to scheduling parallel conference tracks and its human evaluation. *PeerJ Computer Science*, 5:e234, 2019.

Renata Mansini, Wlodzimierz Ogryczak, M Grazia Speranza, and EURO: The Association of European Operational Research Societies. *Linear and mixed integer programming for portfolio optimization*, volume 21. Springer, 2015.

Helga Meier, Nicos Christofides, and Gerry Salkin. Capital budgeting under uncertainty—an integrated approach using contingent claims analysis and integer programming. *Operations Research*, 49(2):196–206, 2001.

Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

Agustín Montero, Isabel Méndez-Díaz, and Juan José Miranda-Bront. An integer programming approach for the time-dependent traveling salesman problem with time windows. *Computers & Operations Research*, 88:280–289, 2017.

Marcelo G Narciso and Luiz Antonio N Lorena. Lagrangean/surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, 114(1): 165–177, 1999.

M G Nicholls. A small-to-medium-sized conference scheduling heuristic incorporating presenter and limited attendee preferences. *Journal of the Operational Research Society*, 58(3):301–308, 2007.

Ralf Niemann and Peter Marwedel. An algorithm for hardware/software partitioning using mixed integer linear programming. *Design Automation for Embedded Systems*, 2:165–193, 1997.

Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. A comprehensive analysis of hyper-heuristics. *Intell. Data Anal.*, 12(1):3–23, January 2008.

Ender Ozcan, Yuri Bykov, Murat Birben, and Edmund K Burke. Examination timetabling using late acceptance hyper-heuristics. In *2009 IEEE Congress on Evolutionary Computation*, pages 997–1004. IEEE, 2009.

Ender Özcan, Mustafa Misir, Gabriela Ochoa, and Edmund K Burke. A reinforcement learning: great-deluge hyper-heuristic for examination timetabling. In *Modeling, analysis, and applications in metaheuristic computing: advancements and trends*, pages 34–55. IGI Global, 2012.

Christos H Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

Kyungchul Park, Seokhoon Kang, and Sungsoo Park. An integer programming approach to the bandwidth packing problem. *Management science*, 42(9):1277–1291, 1996.

Gourab K Patro, Prithwish Jana, Abhijnan Chakraborty, Krishna P Gummadi, and Niloy Ganguly. Scheduling virtual conferences fairly: Achieving equitable participant and speaker satisfaction. In *Proceedings of the ACM Web Conference 2022*, pages 2646–2656, 2022.

Antony E Phillips, Cameron G Walker, Matthias Ehrgott, and David M Ryan. Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research*, 252:283–304, 2017.

Nelishia Pillay. A review of hyper-heuristics for educational timetabling. *Annals of Operations Research*, 239:3–38, 2016.

Marc Pirlot. General local search methods. *European Journal of Operational Research*, 92(3):493 – 511, 1996.

Richard F. Potthoff and Steven J. Brams. Scheduling of panels by integer programming: results for the 2005 and 2006 New Orleans meetings. *Public Choice*, 131(3-4):465–468, 2007.

Richard F Potthoff and Michael C Munger. Use of integer programming to optimize the scheduling of panels at annual meetings of the Public Choice Society. *Public Choice*, 117(1-2):163–175, 2003.

Jakob Puchinger and Günther R. Raidl. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. In José R. Álvarez José Mira, editor, *First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005*, volume 3562, pages 41–53, Las Palmas, Spain, June 2005. Springer-Verlag.

Yaroslav Pylyavskyy, Ahmed Kheiri, and Leena Ahmed. A reinforcement learning hyperheuristic for the optimisation of flight connections. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

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, 2024a. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2024.04.001.

Yaroslav Pylyavskyy, Ahmed Kheiri, and Peter Jacko. A two-phase matheuristic approach to conference scheduling problems. In review, 2024b.

Jeffrey Quesnelle and Daniel Steffy. Scheduling a conference to minimize attendee preference conflicts. In *Proceedings of the 7th multidisciplinary international conference on scheduling: theory and applications (MISTA)*, pages 379–392, 2015.

Rushil Raghavjee and Nelishia Pillay. A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. In *2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 98–103. IEEE, 2012.

Nahid Rezaeinia, Julio C Góez, and Mario Guajardo. Scheduling conferences using data on attendees' preferences. *Journal of the Operational Research Society*, pages 1–14, 2024.

Fabian Riquelme, Elizabeth Montero, Leslie Pérez-Cáceres, and Nicolás Rojas-Morales. A track-based conference scheduling problem. *Mathematics*, 10(21), 2022. ISSN 2227-7390. doi: 10.3390/math10213976.

David M Ryan and Brian A Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.

Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. Grammatical evolution hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 17(6):840–861, 2013.

Scott E. Sampson. Practical implications of preference-based conference scheduling. *Production and Operations Management*, 13(3):205–215, 2004.

Scott E Sampson and Elliott N Weiss. Increasing service levels in conference and educational scheduling: a heuristic approach. *Management Science*, 41(11):1816–1825, 1995.

Scott E. Sampson and Elliott N. Weiss. Designing conferences to improve resource utilization and participant satisfaction. *Journal of the Operational Research Society*, 47(2):297–314, 1996.

Patrick Schittekat, Joris Kinable, Kenneth Sörensen, Marc Sevaux, Frits Spieksma, and Johan Springael. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, 229(2):518–528, 2013.

Alexander Schrijver et al. On cutting planes. *Combinatorics*, 79:291–296, 1980.

Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

Jorge A Soria-Alcaraz, Gabriela Ochoa, Jerry Swan, Martin Carpio, Hector Puga, and Edmund K Burke. Effective learning hyper-heuristics for the course timetabling problem. *European Journal of Operational Research*, 238(1):77–86, 2014.

Thomas Stidsen, David Pisinger, and Daniele Vigo. Scheduling EURO-k conferences. *European Journal of Operational Research*, 270(3):1138–1147, 2018. ISSN 0377-2217.

Gary M. Thompson. Improving conferences through session scheduling. *Cornell Hotel and Restaurant Administration Quarterly*, 43(3):71–76, 2002.

Robert J Vanderbei. Linear programming: foundations and extensions. *Journal of the Operational Research Society*, 49(1):94–94, 1998.

Bart Vangerven, Annette M.C. Ficker, Dries R. Goossens, Ward Passchyn, Frits C.R. Spieksma, and Gerhard J. Woeginger. Conference scheduling — a personalized approach. *Omega*, 81:38–47, 2018. ISSN 0305-0483.

Alexander Veremyev, Oleg A Prokopyev, and Eduardo L Pasiliao. An integer programming framework for critical elements detection in graphs. *Journal of Combinatorial Optimization*, 28:233–273, 2014.

H Paul Williams. Integer programming. In *Logic and Integer Programming*, pages 25–70. Springer, 2009.

Chirayu Wongchokprasitti, Peter Brusilovsky, and Denis Parra-Santander. Conference navigator 2.0: community-based recommendation for academic conferences. 2010.

Wei Yi and Linet Özdamar. A dynamic logistics coordination model for evacuation and support in disaster response activities. *European journal of operational research*, 179 (3):1177–1193, 2007.

F. Zulkipli, H. Ibrahim, and A. M. Benjamin. Optimization capacity planning problem on conference scheduling. In *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, pages 911–915, 2013.

Appendix A

Conference Scheduler - User Guides

A.1 Project Description

A.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.

A.1.2 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 of 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.

A.1.3 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.

A.1.4 How It Works

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

2. **Optimisation Process:** The Excel data is imported into the system and the user selects their preferred algorithm for scheduling optimisation (see Section A.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 subsubsection A.3.4.2).

A.1.5 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

A.1.6 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).

A.1.7 Constraints Available

Users are able to select the constraints to include during the schedule optimisation by assigning a weight value to each constraint (see subsection A.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 subsection A.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 subsection A.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 subsection A.2.1 & subsection A.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 subsection A.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 subsection A.2.1)
- **Rooms capacities:** Users can express preferences regarding the scheduling of tracks into rooms by setting appropriate penalty values. (see subsection A.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 subsection A.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 subsection A.2.2)

- **Tracks' scheduling preferences:** Users can express preferences regarding the scheduling of tracks into sessions by setting appropriate penalty values. (see subsection A.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 subsection A.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.

A.1.8 Optimisation Methods Available

Users are able to select which optimisation method they prefer to optimise the conference schedule (see Section A.3). Each optimisation method has its benefits and limitations which are summarised in Table A.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 A.1: Benefits and Limitations of each Optimisation Method

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

A.1.9 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.

A.1.10 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.

A.1.11 Acknowledgements

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

A.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.

A.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 A.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 A.1: Submissions sheet

A.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.

A.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.

A.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.

A.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 A.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 A.2: Tracks-Sessions Penalty sheet

A.2.6 Tracks-Rooms Penalty

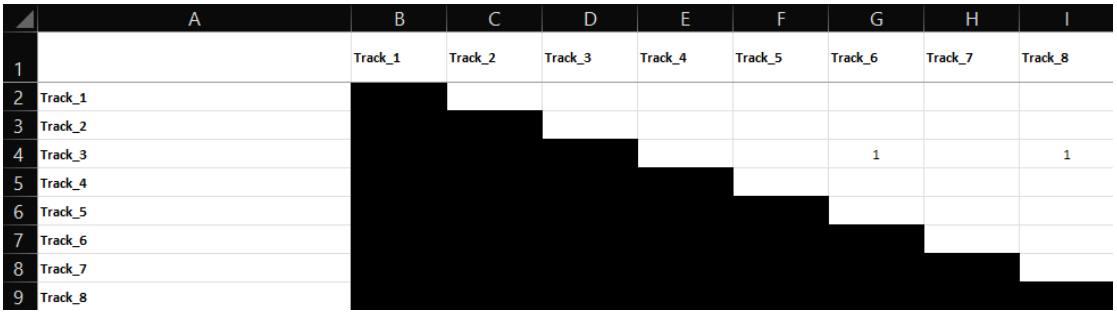
The Tracks-Rooms Penalty sheet is used to control the scheduling process of tracks into rooms as displayed in Figure A.3. Column A contains all tracks, and the number of next columns is given by the total number of rooms available, where each column corresponds to a room (from column B to column E in 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 A.3: Tracks-Rooms Penalty sheet

A.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 A.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 A.4: Similar Tracks sheet

A.2.8 Sessions-Rooms Penalty

The Sessions-Rooms Penalty sheet is used to define unavailability of rooms for certain sessions as presented in Figure A.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 A.5: Sessions-Rooms Penalty sheet

A.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 A.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 A.6: Parameters sheet

A.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.

A.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 subsubsection A.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.

A.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()
```

A.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")
```

A.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 subsubsection A.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.

A.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")
```

A.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)
```

A.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.

A.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()
```

A.3.4 Additional Use Cases

A.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 188 and comment it out, then uncomment line 189 as shown next for the exact model. Similarly, comment out line 492 and uncomment line 493 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`

A.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 A.7).

	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 A.7: 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 A.8.

	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 A.8: 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")
```

Appendix B

Scheduling of GECCO2023

Since 1999, the Genetic and Evolutionary Computation Conference (GECCO) has showcased the latest high-quality results in genetic and evolutionary computation, covering topics that range from genetic algorithms to evolutionary machine learning and their application in real-world problems.

GECCO2023 was held at the Altis Grand Hotel, a landmark venue in Lisbon renowned for hosting numerous notable events over the years. The conference took place from July 15 to July 19, 2023, in a hybrid format, and involved 883 registrants (272 online and 611 onsite). The first two days (Saturday and Sunday) focused on Workshops, Tutorials, Competitions, and Women+@GECCO, while the remaining three days (Monday to Wednesday) featured the Opening and Closing ceremonies, Main Tracks, Keynotes, Poster Sessions, ECiP (Evolutionary Computation in Practice), HOP (Hot Off the Press), HUMIES (Human-Competitive Awards), and Job Market. The Poster Sessions included a variety of submissions, such as poster submissions, Late-Breaking Abstracts (LBAs), and entries from the student workshop and competitions.

The hotel featured a total of 8 rooms spread across four different floors. On the ground floor, there was 1 room with a capacity of 90. The first floor had 3 rooms, each accommodating between 50 and 150 attendees. On the 12th floor, there was another room with a capacity of 70. The 13th floor housed 3 rooms, with capacities ranging from 60 to 150 attendees. Additionally, the Plenary Room, located on floor -1, was available exclusively from Monday to Wednesday for events such as Invited Keynotes.

During Saturday and Sunday, a total of 28 sessions were allocated for workshop tracks, with each workshop lasting between 0.5 to 3 sessions. There were also two additional

sessions reserved for the student workshop. Detailed information for each workshop included the mode of attendance (online vs. in-person) for speakers and organisers, time zones for online presentations, constraints for workshops and individual talks, as well as the preferred order and duration of talks. For tutorial tracks, there were 32 sessions available. Additionally, two sessions were designated for competition tracks.

From Monday to Wednesday, approximately 192 time slots were available for talks. This calculation was based on having 6 sessions per day over 3 days, with 8 rooms available and 4 talks per session. Out of 209 submissions, 29 were designated as HOP submissions. Four sessions were specifically allocated for HOP, with 7-8 time slots per session. Additionally, there were 26 Best Papers (BPs) to be scheduled.

To generate an efficient schedule for the conference, multiple requirements had to be considered, which are presented next:

1. Accommodation of speakers who were unavailable during specific sessions, including time zone consideration for online speakers. (42 in total)
2. Avoid speakers' clashes. Each talk could potentially clash with a number of talks, ranging from 0 to 10.
3. Some talks had to be scheduled in a specific order within their corresponding track.
4. Some tracks were not allowed to be scheduled in specific sessions. (e.g., BP sessions could not be scheduled on Wednesday to allow time for winners certificates preparation)
5. Allocation of tracks into appropriate rooms by considering expected attendance. (e.g., BP sessions required large rooms)
6. Avoid scheduling the same track in parallel to ensure participants do not miss their preferred talk.
7. Some rooms had to be unused during specific sessions. (e.g., During the Job Market session, the Plenary Room had to be free)
8. Accommodation of speakers who could not present in specific rooms due to accessibility or facility issues.
9. Avoid scheduling some tracks in parallel due to similarity. (e.g., Workshop organisers requested certain workshop and tutorials to not be scheduled in parallel)

10. Consideration of the number of time slots that were available in each session. Some flexibility was allowed for certain sessions but the majority of sessions had to consist of 4 time slots.

11. Sessions that were scheduled at the upper floors had to start later, allowing adequate time for participants to reach the floor due to limited elevators capacity.

12. Onsite and online talks had to be grouped separately to offer a smooth transition between the two modes to participants.

13. The sessions on Tuesday before lunch had to end at different times to avoid overcrowding during lunch.

14. Consideration of historical practices and consecutive scheduling of tracks. (e.g., Introductory Tutorials had to be scheduled on Saturday before Specialised Tutorials which had to be scheduled on Sunday)

15. Workshops and Tutorials that were scheduled as first sessions last year were not allowed to be scheduled again first.

16. Limitation of number of rooms assigned per track to prevent participants from frequently switching rooms.

17. Consideration of track chair duties to avoid clashes. (e.g., A person being simultaneously a track chair in two tracks or being a speaker in another track)

18. Some talks required the usage of multiple time slots to be completely scheduled. (e.g., Workshops, Tutorials, etc.)

Scheduling effectively the GECCO2023 conference by considering the above requirements is not a trivial task, and doing so manually would pose an arduous challenge for the organisers. Therefore, to facilitate the scheduling process, we used our conference scheduler which scheduled the conference in an effortless and autonomous manner. Our algorithm only required an Excel file in which we set all the preferences and requirements of the conference. Then, we executed our algorithm to obtain multiple candidate schedules within a few hours. The algorithm itself uses a selection hyper-heuristic that incorporates a random selection method with an improve-or-equal move acceptance criterion, which only accepts moves that do not deteriorate the current solution. Our approach involves simple low-level heuristics, such as: (1) swapping a random talk with another randomly chosen talk within the same track, (2) exchanging one track with

another randomly chosen track, and (3) inserting a random talk into a different position within the same track. Each low-level heuristic is equally likely to be chosen, with a probability of 1/3. To enhance the exploration of solutions, the algorithm periodically shuffles the current solution at regular intervals during execution. Thus, our approach involves running the hyper-heuristic algorithm to support exploitation, followed by shuffling the solution to support exploration. This iterative process continues until a termination criterion is met, with the hyper-heuristic running for another 30 minutes after each shuffle, ensuring continuous exploration and exploitation of solutions. To handle the numerous requirements, we treat each requirement and preference as a soft constraint and assign a weight to it according to its subjective significance. This results in the following objective function, a summation of the weighted soft constraints, which we minimise:

$$\min \sum w_{sc_i} \times V_{sc_i} \tag{B.1}$$

where w_{sc_i} indicates the corresponding weight of constraint SC_i, and V_{sc_i} is the corresponding violated amount of constraint SC_i.

We ran the algorithm for a duration of 4 hours, resulting in 8 intervals as we shuffle the solution every 30 minutes. Note that we scheduled the GECCO2023 Workshop (schedule for the first two days of the conference) and Main conference (schedule for the last three days) separately.

The following are the only constraint violations that were identified:

1. Three main tracks were not scheduled in consecutive order. Although this constraint was low priority for the main tracks, it was considered high priority for the Saturday and Sunday schedules.

2. Seven talks were scheduled on days they specified as unavailable. A closer look revealed that these talks indicated unavailability for the entire days.

3. The Theory track was allocated to two different rooms.

4. The GECH track was assigned to a room with limited seating capacity for its size. Additionally, the ACO-SI and CS tracks were placed in a room with somewhat limited seating capacity for tracks of their size.

5. Two online talks were not scheduled during their most preferred time slots based on their time zones. One was scheduled from 06:00 to 07:30, and another from 21:30 to 23:00.

6. An online workshop submission was scheduled from 22:00 to 23:50, while the optimal slots considering the speaker's time zone would have been 16:30-18:20 or 18:40-20:30. However, scheduling it in any of these ideal slots would require another online talk in the same workshop to be scheduled from either 03:30-05:20 or 05:40-07:30, which is not ideal.

7. An introductory tutorial is scheduled for the first time slot on Sunday rather than Saturday. However, the speaker requested their tutorial be scheduled for Sunday. Additionally, a specialised tutorial is scheduled for the last time slot on Saturday, but the speakers requested it be scheduled for Saturday.

8. The final constraint violation stems from conflicting availability preferences of authors within the same session. One author for an onsite submission in a particular workshop expressed unavailability on Saturday, yet the session was scheduled for that day. Conversely, another author from a different submission in the same session is only available on Saturday. This creates a dilemma where satisfying the constraint for one submission would inevitably violate the constraint for the other.

Overall, our conference scheduler successfully scheduled the GECCO2023 conference in an effortless and autonomous manner, saving a huge amount of time and effort that organisers would otherwise had invested if scheduled manually. The conference scheduler generates a high-level schedule, which determines the time and room allocations for each track, and a low-level schedule, which assigns individual talks to sessions, rooms, and time slots. It considers numerous requirements and preferences such as presenter and attendee preferences, room capacity, room accessibility, and minimising session hopping (i.e., reducing travel between rooms). In addition, it aims to promote inclusive, accessible, and sustainable events. Examples include scheduling specific talks in rooms that support advanced accessibility, minimising changeover times for tracks involving sessions at different locations, and offering flexible options for participants with caring responsibilities or special requirements, such as religious needs, to attend all or part of the conference. Our conference scheduler is also suitable for other conferences with different formats, and has been used in the past to schedule the OR Society's 60th Annual Conference, the New to OR Conference, and the International Symposium on Forecasting 2022, that in addition to GECCOs 2020-2023. Ultimately, we hope that our conference scheduler will enable academics to maximise their conference experience and alleviate the scheduling burden on organisers.

URL

The conference scheduling and various algorithms are available for the research community, which can be accessed on the following GitHub repository: `https://github.com/ahmedkheiri/CSPLib`.

Images

We may use screenshots from the GECCO Sigevo video, but we do not hold copyright for the video content: `https://www.youtube.com/watch?v=wtTpY9th8HY`. Alternatively, we suggest screenshots of the input Excel file, but these are already available on the GitHub repository.

Appendix C

Appendix C

In this section, we provide a sample spreadsheet file to demonstrate its usage. The spreadsheet file consists of the following sheets; Submissions, Tracks, Sessions, Rooms, Parameters, Tracks-Sessions Penalty, Tracks-Rooms Penalty, Similar Tracks, and Sessions-Rooms Penalty.

The Submissions sheet includes all the necessary information regarding submissions as well as submissions related penalties as shown in Figure C.1. Column A contains the reference name or ID of each submission. Column B indicates the track name of the corresponding submission. Column C is used to indicate the number of time slots that each submission requires. Column D is ignored in our model. Column E refers to the timezone in which the presenters of the corresponding submission are located. Column F is used to list the presenter or multiple presenters of the corresponding submission. Similarly, column G is used to list the attendees. This column may include non-presenters, such as co-authors, but it may also include presenters. The latter case will be considered as an attendee conflict during optimisation. 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

	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 C.1: Submissions sheet

	A	B
1	**Tracks**	**Chairs**
2	Track_1	Name_Surname_1
3	Track_2	Name_Surname_2
4	Track_3	Name_Surname_3
5	Track_4	Name_Surname_4
6	Track_5	Name_Surname_5
7	Track_6	Name_Surname_6
8	Track_7	Name_Surname_7
9	Track_8	Name_Surname_8

FIGURE C.2: Tracks sheet

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.

The Tracks sheet contains information regarding track names and the list of track chair names as presented in Figure C.2. In column A, the tracks are listed, and column B contains the names of track chairs for each track. Each track is not limited to only one track chair, it may have multiple track chairs. A chair conflict is created in case there are two tracks with the same person as track chair. If any of the track chairs is also a presenter in another track, then this will be considered as a presenter conflict. Lastly,

	A	B	C	D	E
1	Sessions	Max Number of Timeslots	Date	Start Time	End Time
2	Session_1	2	7/28/2021	9:30	10:30
3	Session_2	4	7/28/2021	10:45	11:45
4	Session_3	2	7/29/2021	9:30	10:30
5	Session_4	3	7/29/2021	10:45	11:45

FIGURE C.3: Sessions sheet

an attendee conflict is created when the same person is a track chair and at the same time is an attendee at a submission that belongs to another track.

Next, the Sessions sheet contains all the necessary information regarding available sessions as displayed in Figure C.3. Column A includes the names of the sessions. Column B refers to the total number of available time slots per session. Column C indicates the date for each session, while column D and E are used to set the starting and ending time for each session respectively.

We skip the Rooms sheet as it simply contains the names of the available rooms. The Parameters sheet includes settings for hybrid or online conferences, and allows to set weight values for penalties as shown in Figure C.4. 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 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.

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 C.5. Column A includes all tracks, and the number of next columns is given by the total number of sessions

	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 C.4: Parameters sheet

	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 C.5: Tracks-Sessions Penalty sheet

available, where each column corresponds to a session (from column B to column E in this example). 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.

We use Tracks-Rooms Penalty sheet to control the scheduling process of tracks into rooms as displayed in Figure C.6. Column A contains all tracks, and the number of next columns is given by the total number of rooms available, where each column corresponds

	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 C.6: Tracks-Rooms Penalty sheet

	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 C.7: Similar Tracks sheet

to a room (from column B to column E in this example). 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.

The Similar Tracks sheet allows to define which pair of tracks should not be scheduled in parallel as shown in Figure C.7. 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). 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. Notice that the value of the penalty does not support preferences here among pairs of tracks because we include this in the model as a hard constraint.

Lastly, we use Sessions-Rooms Penalty sheet to define unavailability of rooms for certain sessions as presented in Figure C.8. 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). For instance, if

	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 C.8: Sessions-Rooms Penalty sheet

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

FIGURE C.9: Solution example

	A	B	C
1	Obj	Evaluate Submissions|Sessions	
2	Final Objective = 1	NEW19A3728 - Thu2	1
3		Total	1

FIGURE C.10: Violations report example

Room_3 is unavailable during Session_4, then we add a penalty value for that session-room pair.

After the completion of the optimisation, we generate a new spreadsheet file that contains the optimised schedule and the violations report. In Figure C.9, we present a solution example by solving the N2OR instance with the extended model where the upper timetable refers to the tracks solution and the lower timetable refers to the submissions solution. The violations report is presented in Figure C.10, where we report the objective value along with details of the violations.

Appendix D

Appendix D

In Table D.1, we present the time required to build the mathematical models for each instance, where t_b indicates the time required to build the model, t_s indicates the required time for the solver to terminate, and t_t indicates the total time required. All times are in seconds.

Table D.1: Overall computing time per instance

	Exact Model			Extended Model		
Instance	t_b	t_s	t_t	t_b	t_s	t_t
N2OR	0.1	0.1	0.2	0.1	0.8	0.9
GECCO19	158.9	3,600.0	3,758.9	152.2	57.5	209.7
GECCO20	4.9	8.3	13.2	5.7	51.8	57.5
GECCO21	1.8	19.7	21.5	2.2	20.5	22.7
OR60	143.6	3.6	147.2	151.8	4.8	156.6
OR60F	79.0	13.9	92.9	99.2	3,600.0	3,699.2
OR60F2	272.6	88.9	361.5	344.5	3,600.0	3,944.5
OR60F3	1,072.7	137.4	1,210.1	1,211.8	3,600.0	4,811.8