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

Yaroslav Pylyavskyy*, Ahmed Kheiri*, Peter Jacko*†
*Lancaster University, Department of Management Science, Lancaster LA1 4YX, UK
{y.pylyavskyy1, a.kheiri, p.jacko}@lancaster.ac.uk
†Berry Consultants, 5 East Saint Helen Street, Abingdon OX14 5EG, UK

***Abstract*—Academic conferences offer notable benefits to participants and promote knowledge advancement. To maximise these benefits, an efficient schedule is crucial. Due to diverse constraints and objectives, various mathematical models and heuristic methods have been developed to meet the unique needs of different conferences. This paper investigates different decision-making tools for creating a generic conference scheduler applicable to a wide range of conferences. We present extended formulations of previously proposed mathematical models to handle constraints at the time slot level, and introduce an approximation model, which is a relaxed version of the previously proposed exact model, obtained through transformations. Additionally, we compare the performance of three methods: a relaxed integer programming model, a decomposed two-phase matheuristic solution approach, and a selection hyper-heuristic algorithm on 16 instances. We highlight the benefits and limitations of each method, providing comprehensive computational results.**

***Index Terms*—scheduling, optimisation, metaheuristic, hyper-heuristic**

I. 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 [1]. Nevertheless, nowadays the focus has shifted towards optimising the schedule quality.

The conference scheduling problem (CSP) was introduced in [2] and proved to be $\mathcal{NP}$-hard in [3]. 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 [1], [4], [5]. There are two primary approaches to CSP based on constraints and objectives: the Presenter-Based Perspective (PBP) and the Attender-Based Perspective (ABP) [5]. 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 [6]. Some studies adopted a mixed approach, balancing both presenter and attendee preferences [7]–[10]. A detailed literature review on conference scheduling can be found in [11].

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. In addition, we compare the performance of the developed decision support tools and report their benefits and limitations in Table III. These tools are freely available at https://github.com/ahmedkheiri/CSPLib 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.

II. 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.

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

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.

III. 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 [11]. In [11], 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 [12] 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 [13]. 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 [11] 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 [14], and discuss the benefits and limitations of each method.

A. 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 [11]. To form the submissions ordering constraint, we first need to introduce a new subset $\mathcal{SU}_t^o$ and a new parameter $id_{s,ts}$:

$su \in \mathcal{SU}_t^o$:The subset of submissions sorted by their specified scheduling order belonging to track t

$id_{s,ts}$:The chronological order of time slot ts belonging to session s

With the new subset and parameter introduced, we can now form the submissions ordering constraint, Eq. 1, which ensures that submissions of a given track are scheduled in the desired specified order.

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

1) Exact Model (IP1): To obtain the equivalent exact model of [11] with time slots, we need to proceed with the following modifications. Firstly, we need to change $X_{s,r}^{(t,su)}$ decision variable as follows:

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

Next, we need to add the set of constraints Eq. 2 which ensures that each time slot either gets assigned one submission or remains empty.

$$\sum_{(t,su)\in\mathcal{TSU}} X_{s,r,ts}^{(t,su)}\leq 1 \qquad \forall\, s\in\mathcal{S},\forall\, r\in\mathcal{R},\forall\, ts\in\mathcal{TS}_s \tag{2}$$

Additionally, we modify constraints Eq. 1, Eq. 2, Eq. 6, Eq. 7, and Eq. 10 of [11] as follows:

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

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

$$\sum_{ts\in\mathcal{TS}_s}\sum_{su\in\mathcal{SU}_t} n^{(t,su)}X_{s,r,ts}^{(t,su)}-|\mathcal{TS}_s|Z_{s,r}^t\leq 0 \quad \forall\, s\in\mathcal{S},\forall\, r\in\mathcal{R},\forall\, t\in\mathcal{T} \tag{3}$$

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

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

Then, we need to change the objective function of the exact model, Eq. 11 of [11], as follows:

$$\begin{aligned}\min \quad & \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})Z^t_{s,r} \\ & + \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s}\sum_{(t,su)\in\mathcal{TSU}}(w_\delta\delta^{(t,su)}_s \\ & + w_\epsilon\epsilon^{(t,su)}_s + w_\zeta\zeta^{(t,su)}_r)X^{(t,su)}_{s,r,ts} \end{aligned} \tag{4}$$

2) Extended Model (IP2): The equivalent extended model of [11] with time slots is obtained by modifying constraints Eq. 14 and the objective function Eq. 20 of [11] as follows:

$$M^a_s \sum_{ts\in\mathcal{TS}_s} X^{(t,su)}_{s,r,ts} + \sum_{r'\in\mathcal{R}\setminus\{r\}}\sum_{ts'\in\mathcal{TS}_s}\sum_{(t',su')\in\mathcal{TSU}^a} X^{(t',su')}_{s,r',ts'} \leq M^a_s \quad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, a \in \mathcal{A}, \forall\, (t,su) \in \mathcal{TSU}^a$$

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

3) Approximation Model (IP3): We can relax the objective function of the exact model, Eq. (4), to create an approximation model by replacing $Z^t_{s,r}$ variables with $X^{(t,su)}_{s,r,ts}$ variables through transformations which we present next. To create the objective function of the approximation model, we get Eq. (3) and proceed with the following steps:

$$\sum_{ts\in\mathcal{TS}_s}\sum_{su\in\mathcal{SU}_t} n^{(t,su)}X^{(t,su)}_{s,r,ts} - |\mathcal{TS}_s|Z^t_{s,r} \leq 0 \quad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

$$\sum_{ts\in\mathcal{TS}_s}\sum_{su\in\mathcal{SU}_t} n^{(t,su)}X^{(t,su)}_{s,r,ts} \leq |\mathcal{TS}_s|Z^t_{s,r} \quad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

$$\sum_{ts\in\mathcal{TS}_s}\sum_{su\in\mathcal{SU}_t} \frac{n^{(t,su)}}{|\mathcal{TS}_s|}X^{(t,su)}_{s,r,ts} \leq Z^t_{s,r} \quad \forall\, s \in \mathcal{S}, \forall\, r \in \mathcal{R}, \forall\, t \in \mathcal{T}$$

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

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

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

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:

$$\begin{aligned}&\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{ts\in\mathcal{TS}_s}\sum_{(t,su)\in\mathcal{TSU}}\frac{X^{(t,su)}_{s,r,ts}}{|\mathcal{TS}_s|}\times \\ &\qquad n^{(t,su)}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r}) \\ &\leq \sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})Z^t_{s,r}\end{aligned} \tag{5}$$

From Eq. (5), we replace the $Z^t_{s,r}$ variables in Eq. (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 $\omega^{(t,su)}_{s,r,ts} = \frac{n^{(t,su)}(w_\alpha\alpha^t_s + w_\beta\beta^t_r + w_\gamma\gamma_{s,r})}{|\mathcal{TS}_s|} + w_\delta\delta^{(t,su)}_s + w_\epsilon\epsilon^{(t,su)}_s + w_\zeta\zeta^{(t,su)}_r$.

IV. 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. In Table I, we present the computational results and observe that the size of IP1 is significantly larger compared to Table 3 of [11]. The IP1 replicated the results of the exact model in [11] for some instances but with significantly worse computational times, and was infeasible for the OR60F instance due to the submissions ordering constraints, Eq. 1. Additionally, IP1 failed to find solutions within time limit for instances OR60F2 and OR60F3.

We observe that the IP2 replicated the results of the extended model presented in [11] for N2OR, GECCO20, and GECCO21 instances at worse computational times compared to Table 6 of [11]. It also found a slightly worse objective for GECCO19 instance and failed to find solutions for OR60F2 and OR60F3 within the time limit.

Similar to IP2, the objective values of IP3 have been computed through evaluation functions in order to present the objective value of IP1. The results of IP3 compared to IP1 reveal that the objective of IP1 was replicated in all instances except for GECCO19 for which a slightly worse objective was found. Moreover, IP3 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 [11] for some instances. The computational times were significantly worse,

TABLE I

IP1, IP2 AND IP3 RESULTS: 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.

	IP1					IP2					IP3			
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

TABLE II

THE PERFORMANCE OF IP4, MATHEURISTIC (MH), AND HYPER-HEURISTIC (HH). BEST SOLUTIONS ARE HIGHLIGHTED IN BOLD.

Instance	IP4	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

compared to the models in [11], 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 [11] 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.

A. 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 [14]. To make the methods comparable, we considered IP2 from Section III-A2 and relaxed certain hard constraints as in [12], which we refer to as IP4. 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 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 II.

By observing Table II, 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 IP4 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 needs 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 [11] 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.

A summary of the benefits and drawbacks of each method is presented in Table III. 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

TABLE III
BENEFITS AND DRAWBACKS OF EACH METHOD

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

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.

V. 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 (IP1, IP2). Moreover, we presented an approximation model (IP3) 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 [11] 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 (IP4), matheuristic and hyper-heuristic techniques by solving the benchmark instances (available at [14]) 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. Another future direction could explore 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 submissions. Lastly, 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 time slots for sessions.

REFERENCES

[1] S. E. Sampson, "Practical implications of preference-based conference scheduling," *Production and Operations Management*, vol. 13, no. 3, pp. 205–215, 2004.

[2] R. W. Eglese and G. K. Rand, "Conference seminar timetabling," *Journal of the Operational Research Society*, vol. 38, no. 7, pp. 591–598, 1987.

[3] J. Quesnelle and D. Steffy, "Scheduling a conference to minimize attendee preference conflicts," in *Proceedings of the 7th multidisciplinary international conference on scheduling: theory and applications (MISTA)*, 2015, pp. 379–392.

[4] S. E. Sampson and E. N. Weiss, "Designing conferences to improve resource utilization and participant satisfaction," *Journal of the Operational Research Society*, vol. 47, no. 2, pp. 297–314, 1996.

[5] G. M. Thompson, "Improving conferences through session scheduling," *Cornell Hotel and Restaurant Administration Quarterly*, vol. 43, no. 3, pp. 71–76, 2002.

[6] 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)*, 2013, pp. 911–915.

[7] M. G. Nicholls, "A small-to-medium-sized conference scheduling heuristic incorporating presenter and limited attendee preferences," *Journal of the Operational Research Society*, vol. 58, no. 3, pp. 301–308, 2007.

[8] T. Stidsen, D. Pisinger, and D. Vigo, "Scheduling EURO-k conferences," *European Journal of Operational Research*, vol. 270, no. 3, pp. 1138–1147, 2018.

[9] B. Vangerven, A. M. Ficker, D. R. Goossens, W. Passchyn, F. C. Spieksma, and G. J. Woeginger, "Conference scheduling — a personalized approach," *Omega*, vol. 81, pp. 38–47, 2018.

[10] F. Riquelme, E. Montero, L. Pérez-Cáceres, and N. Rojas-Morales, "A track-based conference scheduling problem," *Mathematics*, vol. 10, no. 21, 2022.

[11] Y. Pylyavskyy, P. Jacko, and A. Kheiri, "A generic approach to conference scheduling with integer programming," *European Journal of Operational Research*, vol. 317, no. 2, pp. 487–499, 2024.

[12] Y. Pylyavskyy, A. Kheiri, and P. Jacko, "A two-phase matheuristic approach to conference scheduling problems," 2024, in review.

[13] A. Kheiri, Y. Pylyavskyy, and P. Jacko, "Efficient scheduling of gecco conferences using hyper-heuristic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '24 Companion. New York, NY, USA: Association for Computing Machinery, 2024, pp. 1732–1737.

[14] ——, "Csplib - a benchmark library for conference scheduling problems," 2024, in review.