Computers & Industrial Engineering 157 (2021) 107279

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

An adaptive large neighborhood search for the multiple-day music rehearsal problems

Pisit Jarumaneeroj[a,b,*], Noppadon Sakulsom[c]

[a] Department of Industrial Engineering, Chulalongkorn University, Thailand
[b] Regional Centre for Manufacturing Systems Engineering, Chulalongkorn University, Thailand
[c] Department of Logistics Engineering, University of the Thai Chamber of Commerce, Thailand

ARTICLE INFO

Keywords:
Adaptive large neighborhood search
Music rehearsal problem
Capacitated vehicle routing problem
Scheduling

ABSTRACT

This paper presents an Adaptive Large Neighborhood Search (ALNS) framework to solve the Multiple-Day Music Rehearsal Problem (MMRP), where music pieces with different player sets and durations are arranged in a predefined number of rehearsal days so that the total days of attendance and waiting times experienced by all players are minimized. Two variants of the MMRP, namely the MMRP without setup times (MMRP-0) and the MMRP with setup times (MMRP-1), are herein explored based on mathematical formulations of the Capacitated Vehicle Routing Problem (CVRP) and the Music Rehearsal Problem (MRP). Extensive computational results on 120 generated instances and 78 benchmark instances indicate that the ALNS is greatly efficient as it can provide equivalent or better solutions than the exact method and a benchmark heuristic from the literature, with much less computational time. We also find that the ALNS tends to perform better in large and complicated MMRP settings, considering that it outperforms the time-restricted CPLEX in 34 out of 120 generated instances and successfully finds 4 new best-known solutions to 8 large benchmark instances.

1. Introduction

Music rehearsal is crucially important for the success of any concerts and orchestral performances, where a collection of both musical instruments and musicians must be present at the rehearsal place on the period at which a particular music piece is rehearsed. Since different music pieces may require different sets of players, and there are typically a number of music pieces to be rehearsed for each event, inattentive music piece arrangement may result in a delay — and so the total rehearsal cost due to an increase in both numbers of rehearsal and show-up days experienced by all players. In order to reduce such cost, music piece scheduling must be adequately and efficiently administered, where we will refer to such a problem as the Music Rehearsal Problem (MRP).

The very first MRP was presented by Adelson, Norman, and Laporte (1976), whose concern lay with the arrangement of music pieces that minimized total man-hours spent by all players. In their MRP model, a player must arrive at the rehearsal place by the time of the first piece he plays and leaves immediately after his last performance, which may not necessarily be the last of the day. While the authors could adroitly solve the underlying MRP by a Dynamic Programming (DP) approach, unfortunately, due to the curse of dimensionality (Bellman, 1957), the applicability of such an approach was deemed limited, especially for practical MRP instances with a large number of players. Smith (2003), later on, explored the MRP with a more specific objective, that is, to minimize total idling (waiting) times experienced by all players at the rehearsal place. In order to visualize this issue, let us consider an example of a simple rehearsal plan with nine music pieces and five players as shown in Fig. 1. If player *i* is involved with music piece *j*, the element (*i*, *j*) in the figure will be 1; and, 0, otherwise. Based on this music piece sequence (1–2–3–4–5–6–7–8–9), Player 1 must stay at the rehearsal place all day, and the total waiting time for such a player can be determined by the sum of the durations of music pieces 2, 4, 5, and 7 — which is 11 time units. As opposed to Player 1, Player 2 may arrive at the beginning of the third performance, *i.e.* music piece 3, and he may leave at the end of the eighth performance, as he is not involved with the last music piece of the day. The waiting time experienced by Player 2 is, therefore, the duration of the fifth music piece, which is four time units. In sum, this sequence yields a total of 39 time units of waiting. However, if we rearrange the sequence of music pieces into 8–4–1–7–6–3–9–2–5 as shown in Fig. 2, the total waiting time could be significantly reduced to nine, which is equivalent to a 77% reduction in waiting time. Although,

* Corresponding author at: Department of Industrial Engineering, Chulalongkorn University, Thailand.
E-mail address: pisit.ja@chula.ac.th (P. Jarumaneeroj).

https://doi.org/10.1016/j.cie.2021.107279
Received 13 April 2020; Received in revised form 25 August 2020; Accepted 25 March 2021
Available online 30 March 2021

Music Piece	1	2	3	4	5	6	7	8	9
Player 1	1	0	1	0	0	1	0	1	1
Player 2	0	0	1	1	0	1	1	1	0
Player 3	1	0	1	1	1	1	1	0	1
Player 4	1	1	1	0	1	1	1	0	1
Player 5	0	1	1	0	0	1	0	0	1
Duration	2	3	2	1	4	5	3	4	3

Fig. 1. A solution to a single-day music rehearsal problem (waiting slots in grey).

Music Piece	8	4	1	7	6	3	9	2	5
Player 1	1	0	1	0	1	1	1	0	0
Player 2	1	1	0	1	1	1	0	0	0
Player 3	0	1	1	1	1	1	1	0	1
Player 4	0	0	1	1	1	1	1	1	1
Player 5	0	0	0	0	1	1	1	1	0
Duration	4	1	2	3	5	2	3	3	4

Fig. 2. A modified solution to a single-day music rehearsal problem with the least waiting time (waiting slots in grey).

Constraint Programming (CP) was introduced as a solution approach for this MRP variant, due to its complexity and comparatively long computational time, Smith (2003) found that solving large MRP instances by the CP was less likely.

Another class of problems that most relates to the MRP is the Film Production Scheduling Problem (FPSP), first introduced by Cheng, Diamond, and Lin (1993), where a film producer needs to sequence a series of shooting scenes such that the total cost of actors — or the talent cost — is minimized. In the FPSP, talent cost is incurred based on hold days — the number of days at which an actor is present at the shooting place regardless of the scenes shot. Furthermore, the filming is divided into a number of shooting days, each with different numbers of predefined scenes and required actors. Observably, the underlying FPSP is equivalent to the single-day MRP defined by Smith (2003), where shooting days and hold days correspond to music pieces and waiting time slots in the context of MRP, respectively. Cheng et al. (1993) applied a Branch and Bound (BB) method to the FPSP as a primary solution strategy and later devised a two-phase heuristic capable of solving larger FPSP instances (with more than 15 shooting days) within an acceptable computational period. de la Banda, Stuckey, and Chu (2011) instead solved the FPSP by a modified DP with double-ended search, which resulted in a drastic improvement in terms of both solution quality and computational time when compared to that of Cheng et al. (1993). Recently, Qin, Zhang, Lim, and Liang (2016) have extended the concept of double-ended search, along with other accelerating techniques — including preprocessing, dominance rules, and caching search states — to enhance the performance of BB. They found that this enhanced branch-and-bound algorithm significantly outperformed those of Cheng et al. (1993) and de la Banda et al. (2011) in all aspects.

While attention has been paid mostly to the FPSP — or, equivalently, the single-day MRP — more advanced FPSP/MRP settings have also been investigated by Bomsdorf and Derigs (2008), Wang, Chuang, and Lin (2016) and Sakulsom and Tharmmaphornphilas (2014). More formally, Bomsdorf and Derigs (2008) studied and developed a decision support system for a more general FPSP, called the Movie Shoot Scheduling Problem (MSSP), that took into account several practical constraints pertaining to the required resources and filming atmosphere. Wang et al. (2016) generalized the FPSP by incorporating daily shooting capacity into the problem, implying that the total duration of shooting must not exceed the daily shooting limit. Based on this restriction, scene arrangement and talent scheduling were then combined and solved in a multi-phase manner, where shooting scenes were first allocated into each working day based on simple bin-packing heuristics; and, once completed, the resulting solutions were subsequently improved by a combination of Iterated Local Search (ILS) and Tabu Search (TS) in the latter phase.

Sakulsom and Tharmmaphornphilas (2014), on the other hand, focused more on a complicated MRP where music pieces were allowed to be rehearsed in multiple rehearsal days — each with a daily rehearsal limit like Wang et al. (2016). Fig. 3, for instance, shows an example of a simple two-day rehearsal plan with 14 music pieces, five players, and a daily rehearsal limit of 20 time units.

The objective of their study was to determine the sequence of music pieces that minimized the total number of show-up days, together with the resulting waiting times, experienced by all players. A two-phase method was devised to help solve the problem, where initial solutions were constructed based on the concept of cell formation (Sakulsom & Tharmmaphornphilas, 2011); and, once done, the sequences of music pieces that yielded the minimum waiting times were then determined by an Integer Programming (IP) model. While their approach is interesting in several aspects, the resulting solutions might be locally optimal as the whole MRP is disaggregated into sub-problems and solved sequentially.

In contrast to the previous literature, this paper attempts to provide single-stage mathematical formulations for this so-called Multiple-Day Music Rehearsal Problem (MMRP) and its practical variant, where additional setup time occurs whenever there is a change on player sets between two consecutively scheduled music pieces. For ease of discussion, the MMRP without setup times and the MMRP with setup times will be hereby referred to as the MMRP-0 and MMRP-1, respectively. In terms of problem settings, the MMRP-0 is equivalent to the MRP investigated by Sakulsom and Tharmmaphornphilas (2014), while the MMRP-1 differs slightly as we include setups between music pieces into consideration. The formulations of both MMRP-0 and MMRP-1 are based on the mathematical formulations of two well-known $\mathcal{NP}$-hard problems — namely the Capacitated Vehicle Routing Problem (CVRP) and the MRP (Lenstra & Rinnooy Kan, 1981; Cheng et al., 1993; Sakulsom & Tharmmaphornphilas, 2014) — where a thorough discussion of the underlying problems, along with their associated IP formulations, is provided in Section 2. As both MMRP-0 and MMRP-1 are $\mathcal{NP}$-hard — by a reduction from either the CVRP or the MRP — an Adaptive Large Neighborhood Search (ALNS) is therefore devised to help solve practically large MMRP-0 and MMRP-1 instances. The detailed implementation of the proposed ALNS is provided in Section 3, followed by intensive experimental results in Section 4. Lastly, Section 5 concludes our work and some future research directions.

2. Problem definition

The Multiple-Day Music Rehearsal Problem (MMRP) concerns the finding of an optimal sequence for both music pieces and players in a predefined number of rehearsal days so that the total cost associated with player attendance and idling is minimized. Two variants of the MMRP will be discussed in this paper, namely the MMRP without setup times (MMRP-0) and the MMRP with setup times (MMRP-1). Similar to Sakulsom and Tharmmaphornphilas (2014), the daily rehearsal limit is fixed over the planning horizon. Music pieces may also differ in terms of both music durations and required players. Players must be present at the rehearsal place by the times at which the first music pieces they perform start, and they may immediately leave once their last performances — not necessarily be the last of the day — have ended. Since the players are paid based on the days they show up and improper music piece scheduling would only create unnecessary waiting that leads to an increase commitment, the objective of our MMRPs is therefore defined as to minimize the total cost of show-up and waiting experienced by all players. Though, the objective of MMRP-1 differs slightly from that of

Day	Day 1							Day 2						
Music Piece	7	9	6	1	13	8	2	3	4	12	10	5	11	14
Player 1	0	0	1	1	1	1	1	0	1	1	1	1	0	0
Player 2	1	0	0	0	0	0	0	1	1	1	1	1	1	0
Player 3	0	1	1	0	1	1	1	0	0	0	1	1	1	1
Player 4	1	1	1	0	0	0	0	1	1	0	1	1	1	1
Player 5	1	1	1	1	0	0	0	0	0	1	0	0	0	0
Duration	3	4	4	1	2	2	4	4	3	1	4	2	3	3

Fig. 3. An example of a two-day rehearsal plan with 14 music pieces, five players, and a daily rehearsal limit of 20 time units (waiting slots in grey).

MMRP-0 as we also include setups between music pieces into consideration. The concept of setups in the MMRP-1 is quite similar to that of manufacturing systems where additional time is needed to clear, prepare, and set the stage for the next music pieces to be rehearsed. For simplicity, we assume that the setup time between two consecutively scheduled music pieces is proportional to the difference between player sets required by them, including both present and absent players.

2.1. *The Multiple-Day Music Rehearsal Problem without Setup Times (MMRP-0)*

2.1.1. *Sets and parameters*

- I is a set of music pieces, ranging from 1 to N.
- I_ϕ is a set of music pieces, including a fictitious node ϕ denoting the beginning of each rehearsal day.
- P is a set of players.
- D is a set of rehearsal days.
- K is a set of performance orders in each rehearsal day.
- Q is a daily available rehearsal limit.
- M is a large integer number.
- d_i denotes the duration of music piece $i \in I_\phi$, where $d_\phi = 0$.
- c_a denotes a player's daily wage.
- c_b denotes penalty cost from waiting, *i.e.* player's hourly wage.
- $play_{pi}$ is a parameter indicating whether player $p \in P$ is required for music piece $i \in I$; where,

$$play_{pi} = \begin{cases} 1 & \text{, player } p \in \text{P plays music piece } i \in \text{I}, \\ 0 & \text{, otherwise.} \end{cases}$$

2.1.2. *Decision variables*

- r_i^d is a nonnegative integer decision variable representing the order of music piece $i \in I$ on rehearsal day $d \in D$, *i.e.* $r_A^1 = 1$ indicates that music piece A is the first to be rehearsed on day 1.
- x_{ij}^d is a binary decision variable indicating whether music piece $i \in I$ is rehearsed before $j \in I$ on rehearsal day $d \in D$.
- z_i^d is a binary decision variable indicating whether music piece $i \in I$ is rehearsed on rehearsal day $d \in D$.
- y_{ik}^d is a binary decision variable indicating whether music piece $i \in I$ is rehearsed as the k^{th} performance of rehearsal day $d \in D$.
- pl_{pik}^d is a binary decision variable indicating whether player $p \in P$ plays music piece $i \in I$, which has been rehearsed as the k^{th} performance of rehearsal day $d \in D$.
- s_{pk}^d is a binary decision variable indicating whether player $p \in P$ is at the rehearsal place during the k^{th} performance of rehearsal day $d \in D$.
- p_{pk}^d is a binary decision variable indicating whether player $p \in P$ is required during the k^{th} performance of rehearsal day $d \in D$.
- a_{pk}^d is a binary decision variable indicating the presence of player $p \in P$ in the k^{th} performance of rehearsal day $d \in D$ as his first performance, where it takes the value of one since then until the end of the day.
- l_{pk}^d is a binary decision variable indicating the presence of player $p \in P$ in the k^{th} performance of rehearsal day $d \in D$ as his last performance, where it takes the value of one from the beginning of the day until such a performance.
- w_{pk}^d is a binary decision variable indicating whether player $p \in P$ is idling at the rehearsal place during the k^{th} performance of rehearsal day $d \in D$.
- $wait_{pik}^d$ is a binary decision variable indicating whether player $p \in P$ is idling at the rehearsal place while music piece $i \in I$ is performed as the k^{th} performance of rehearsal day $d \in D$.
- u_p^d is a binary decision variable indicating whether player $p \in P$ is required at the rehearsal place on day $d \in D$; but, $u_p^d = 0$, if player $p \in P$ is required on day $d \in D$, and $u_p^d = 1$, otherwise.
- $come_p^d$ is a binary decision variable indicating the presence of player $p \in P$ on rehearsal day $d \in D$.

2.1.3. *Mathematical formulation of the MMRP-0*

The IP formulations of both MMRP-0 and MMRP-1 are developed based on two different models, that is, the Capacitated Vehicle Routing Problem (CVRP) and the Music Rehearsal Problem (MRP), where the rehearsal days and music pieces may be regarded as vehicle routes and customers in the CVRP setting. Every time a vehicle moves from one location to another — or music pieces in the context of MRP — the capacity of a vehicle is gradually consumed by music piece duration. The daily rehearsal limit is, thus, equivalent to the capacity of a vehicle. For clarity, Fig. 4 shows an example of a two-day MRP in the CVRP setting. In this example, nine music pieces (A - I) are arranged in two rehearsal days — five on day 1 and four on day 2 — with only three involved players. Observe that the order of music pieces rehearsed on day d is preserved by the CVRP decision variable x_{ij}^d, where $x_{\cdot j}^d$ could take the value of one only if the decision variable z_j^d equals one. These decision variables will be later linked with the MRP decision variable y_{ik}^d via a counter decision variable r_i^d that defines the order of music pieces to be rehearsed on each rehearsal day.

Objective Function

$$MinZ \quad = \quad c_a \sum_{p\in P}\sum_{d\in D} come_p^d + c_b \sum_{p\in P}\sum_{i\in I}\sum_{k\in K}\sum_{d\in D} wait_{pik}^d \cdot d_i \tag{1}$$

Similar to Sakulsom and Tharmmaphornphilas (2014), as we do assume that players are paid based on show-up days, improper music piece scheduling would only create unnecessary waiting that leads to an increase in days of attendance — and so the total rehearsal cost. The

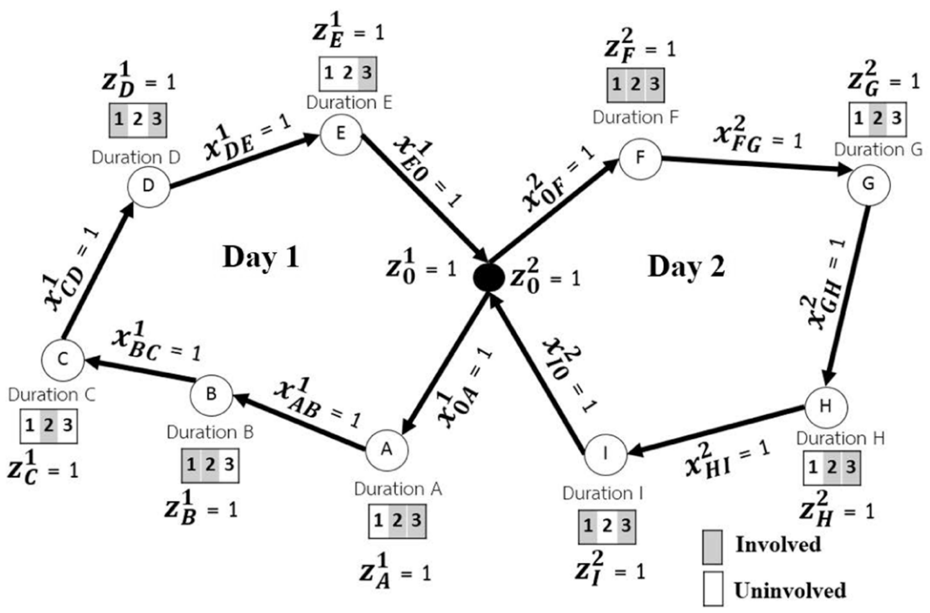

Fig. 4. An example of a two-day MRP without setup times (MMRP-0), with nine music pieces and three players, illustrated in a CVRP setting — where players in grey cells are required for the music pieces.

objective function of MMRP-0, as shown in Eq. (1), is therefore defined so that the total cost of show-up and waiting experienced by all players is minimized.

Constraints

Considering that the MMRP-0 is a combination between the CVRP and the MRP, its corresponding sets of constraints are therefore comprised of those from both problems as follows.

$$\sum_{i\in I_\phi} x_{ij}^d = z_j^d, \forall j \in I, d \in D \tag{2}$$

$$\sum_{j\in I_\phi} x_{ij}^d = z_i^d, \forall i \in I, d \in D \tag{3}$$

$$\sum_{i\in I} x_{\phi i}^d = z_\phi^d, \forall d \in D \tag{4}$$

$$\sum_{d\in D} z_i^d = 1, \forall i \in I \tag{5}$$

$$\sum_{i\in I_\phi} d_i \cdot z_i^d \le Q \cdot z_\phi^d, \forall d \in D \tag{6}$$

Similar to the CVRP, flow conservation is preserved by Eqs. (2) and (3), while Eq. (4) helps define the initiation of rehearsal day $d \in D$. Each music piece is assigned to exactly one rehearsal day by Eq. (5); and, in each rehearsal day, the total time spent must not exceed the daily available rehearsal limit (Q) as imposed by Inequality (6).

$$z_\phi^d \ge z_\phi^{d+1}, \forall d \in D \backslash |D| \tag{7}$$

$$r_j^d \ge x_{\phi j}^d - M\left(1 - x_{\phi j}^d\right), \forall j \in I, d \in D \tag{8}$$

$$r_j^d \ge r_i^d + x_{ij}^d - M\left(1 - x_{ij}^d\right), \forall i,j \in I, d \in D \tag{9}$$

Constraint (7) controls the initiation of new rehearsal days, where a new rehearsal day could be initiated only when the previous days exist. The sequence of music pieces on rehearsal day $d \in D$ — or equivalently r_i^d — is defined by Inequalities (8) and (9), which are equivalent to subtour elimination constraints in the context of CVRP.

Music piece scheduling is defined by Constraints (10) – (12), where Inequality (10) states that at most one music piece can be rehearsed in one particular performance order of a day, while Eqs. (11) and (12) ensure that each piece will definitely be rehearsed. The CVRP counter decision variable r_i^d is then linked with the binary decision variable y_{ik}^d as to define the performance sequence on each rehearsal day by Constraint (13). Lastly, Eqs. (14) and (15) prescribe the presence of required players during the day.

$$\sum_{i\in I} y_{ik}^d \le 1, \forall k \in K, d \in D \tag{10}$$

$$\sum_{k\in K}\sum_{d\in D} y_{ik}^d = 1, \forall i \in I \tag{11}$$

$$\sum_{k\in K} y_{ik}^d = z_i^d, \forall i \in I, d \in D \tag{12}$$

$$\sum_{k\in K} k \cdot y_{ik}^d = r_i^d, \forall i \in I, d \in D \tag{13}$$

$$pl_{pik}^d = y_{ik}^d \cdot play_{pi}\ , \forall p \in P, i \in I, k \in K, d \in D \tag{14}$$

$$p_{pk}^d = \sum_{i\in I} pl_{pik}^d\ , \forall p \in P, k \in K, d \in D \tag{15}$$

To better explain Eqs. (13) – (15), let us consider music piece E in Fig. 4, which is the last performance of day 1. As music piece E is the fifth performance of day 1, it follows from the previous CVRP constraints that $r_E^1 = 5$ and $z_E^1 = 1$. Moreover, since y_{ik}^d is a binary decision variable and k is a positive integer number, Eq. (13) would lead to $\sum_{k\in K} k \cdot y_{Ek}^1 = r_E^1 = 5$, or equivalently $y_{E5}^1 = 1$ — implying that music piece E is rehearsed as the fifth performance of day 1. Once, the performance order of music piece E is defined, we can then entail a set of players required for music piece E by Eqs. (14) and (15), *i.e.* $pl_{1E5}^1 = pl_{2E5}^1 = 1$ and $p_{15}^1 = p_{25}^1 = 1$, respectively.

$$a_{pk}^d \ge p_{pk}^d\ , \forall p \in P, k \in K, d \in D \tag{16}$$

$$l_{pk}^d \ge p_{pk}^d\ , \forall p \in P, k \in K, d \in D \tag{17}$$

$$a_{pk}^d \le a_{p(k+1)}^d\ , \forall p \in P, k \in K \backslash |K|, d \in D \tag{18}$$

$$l_{pk}^{d} \geq l_{p(k+1)}^{d}\ , \forall p \in P, k \in K \backslash |K|, d \in D \tag{19}$$

$$s_{pk}^{d} \geq a_{pk}^{d} + l_{pk}^{d} - 1, \forall p \in P, k \in K, d \in D \tag{20}$$

Inequalities (16) – (20) are adapted from Sakulsom and Tharmmaphornphilas (2014), where Constraints (16) and (17) help force players to arrive and leave at proper time periods, while Constraints (18) – (20) help define the time periods at which the players must stay in the rehearsal place. In order to visualize this set of constraints, let us consider a sequence of music pieces performed on day 1 of the example shown in Fig. 4; but, we will emphasize only on the first player — namely, Player 1 (see Fig. 5 for more details). Based on this sample sequence $A - B - C - D - E$, Player 1 must arrive at the beginning of music piece B and he may leave once music piece D ends. In this case, by the definitions of a_{pk}^{d} and l_{pk}^{d}, a_{1k}^{1} will take the value of one from the second performance ($k = 2$) until the end of the day ($k = 5$), while l_{1k}^{1} will take the value of one from the first performance ($k = 1$) until the one that he last performs, *i.e.* music piece D with $k = 4$. Based on the values of a_{1k}^{1} and l_{1k}^{1}, the time period for which Player 1 must stay in the rehearsal place (s_{1k}^{1}) — from the second to the fourth performance — can be determined by Constraint (20), as shown in Table 1.

Since we know exactly when a player arrives and leaves the rehearsal place, from s_{ik}^{d}, the waiting time that a player experiences can then be defined by Constraints (21) and (22), where Inequality (22) links the order of scheduled music pieces defined by Eq. (21) with actual music durations in the objective function.

$$w_{pk}^{d} = s_{pk}^{d} - p_{pk}^{d}\ , \forall p \in P, k \in K, d \in D \tag{21}$$

$$wait_{pik}^{d} \geq w_{pk}^{d} + y_{ik}^{d} - 1, \forall p \in P, i \in I, k \in K, d \in D \tag{22}$$

For instance, based on Table 1, if we know the values of p_{1k}^{1} and s_{1k}^{1}, the time period at which Player 1 must wait can be determined by Eq. (21), as shown in Table 2. And, since we know that $w_{13}^{1} = 1$ and $y_{C3}^{1} = 1$, thus $wait_{1C3}^{1} = 1$ by Inequality (22) and the fact that $wait_{1C3}^{1}$ is a binary decision variable. The time period at which Player 1 waits on day 1, as denoted in the objective function, is therefore $\sum_{i \in P} \sum_{k \in K} wait_{1ik}^{1} \cdot d_i = wait_{1C3}^{1} \cdot d_3 = d_3$.

In addition to the boundary constraints for all decision variables, Inequalities (23) and (24) makeup the last set of MMRP-0 constraints that helps define the presence of player $p \in P$ on any rehearsal day $d \in D$. More specifically, if player $p_1 \in P$ is present on day $d_1 \in D$, or equivalently $\sum_{k \in K} a_{p_1 k}^{d_1} \geq 1$, $u_{p_1}^{d_1}$ and $come_{p}^{d}$ must equal to 0 and 1, respectively.

Table 1
Values of a_{1k}^{1}, l_{1k}^{1}, and s_{1k}^{1} for Player 1 based on Fig. 5.

Sequence (k)	1	2	3	4	5
a_{1k}^{1}	0	1	1	1	1
l_{1k}^{1}	1	1	1	1	0
s_{1k}^{1}	0	1	1	1	0

Table 2
Values of p_{1k}^{1}, s_{1k}^{1}, and w_{1k}^{1} for the example shown in Fig. 5.

Sequence (k)	1	2	3	4	5
p_{1k}^{1}	0	1	0	1	0
s_{1k}^{1}	0	1	1	1	0
w_{1k}^{1}	0	0	1	0	0

But, on the contrary, if $\sum_{k \in K} a_{p_1 k}^{d_1} = 0$, this player p_1 is not required on day d_1, and, hence, $u_{p_1}^{d_1}$ will be 1 forcing $come_{p}^{d}$ to be zero.

$$-come_{p}^{d} + 1 \leq M \cdot u_{p}^{d}\ , \forall p \in P, d \in D \tag{23}$$

$$\sum_{k \in K} a_{pk}^{d} \leq M \cdot \left(1 - u_{p}^{d}\right)\ , \forall p \in P, d \in D \tag{24}$$

2.2. The Multiple-Day Music Rehearsal Problem with Setup Times (MMRP-1)

2.2.1. Sets and parameters

Besides the index sets and parameters defined in the previous section, we do need a parameter $setup_{ij}$ to denote the setup time required for rehearsing music piece $j \in I$ right after music piece $i \in I$. In this setting, we assume that the setup time between music pieces $i \in I$ and $j \in I$ is proportional to the difference between required player sets. Mathematically, $setup_{ij} = \alpha \cdot \triangle_{ij}$, where α is a setup parameter and $\triangle_{ij}$ is the difference between player sets of music pieces $i \in I$ and $j \in I$.

2.2.2. Decision variables

Likewise, an additional decision variable set_{pij}^{d} is introduced to the MMRP-1 so that waiting time experienced by player $p \in P$ during setup between music pieces $i \in I$ and $j \in I$ on rehearsal day $d \in D$ is properly captured.

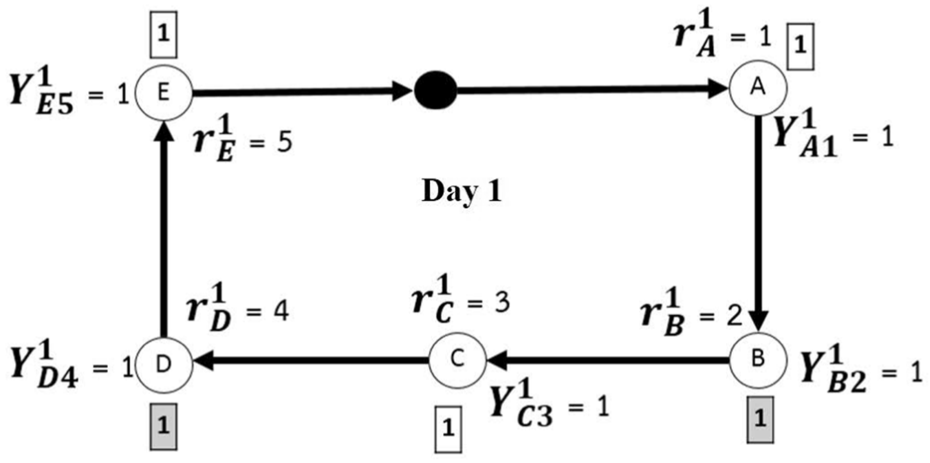

Fig. 5. The rehearsal schedule on the first day of Fig. 4 with emphasis on the first player.

2.2.3. Mathematical formulation of the MMRP-1

Objective Function

$$MinZ = c_a\sum_{p\in P}\sum_{d\in D}come_p^d + c_b\sum_{p\in P}\sum_{i\in I}\sum_{k\in K}\sum_{d\in D}wait_{pik}^d\cdot d_i + c_b\sum_{p\in P}\sum_{i\in I_\phi}\sum_{j\in I}\sum_{d\in D}setup_{ij}\cdot set_{pij}^d \quad (25)$$

The objective function of MMRP-1, as shown in Eq. (25), slightly differs from that of MMRP-0 as we include unproductive time from waiting during setups into consideration, *i.e.* the third term. The concept of setups in the MMRP-1 is quite similar to that of manufacturing systems as we need to spend additional time to clear, prepare, and set the stage for the next pieces to be rehearsed. It is worth noting that, while we may avoid unnecessarily long setup periods by arranging similar music pieces to be rehearsed next to each other, this may, however, be inefficient in terms of idling as different music pieces may require different sets of players. Accordingly, both idling and waiting during setups should be concurrently optimized, along with the explicit cost of player attendance, as stated in the above objective function.

Constraints

To define the MMRP-1, we do need all constraints previously described in the MMRP-0; but, with slight modification on Constraint (6) so that setup times are included in the daily available rehearsal limit — as shown in Inequality (26) below. Furthermore, Inequality (27) is needed to help define setups between music pieces. For instance, if $x_{AB}^1 = 1$ and $wait_{1B2}^1 = 1$ — or equivalently, music piece B is rehearsed right after music piece A on the first rehearsal day, but Player 1 is not involved with music piece B, which is the second performance of a day — the duration that Player 1 needs to wait must be computed based on both the setup from A to B and the time spent for B, which is separately captured by set_{1AB}^1 and $wait_{1B2}^1$, respectively.

$$\sum_{j\in I}\left[\sum_{i\in I_\phi}\left(setup_{ij}\cdot x_{ij}^d\right) + d_j\cdot z_j^d\right] \leq Q\cdot z_\phi^d \text{ ,}\forall d\in D \quad (26)$$

$$set_{pij}^d \geq wait_{pjk}^d + x_{ij}^d - 1, \forall p\in P, i\in I_\phi, j\in I, k\in K, d\in D \quad (27)$$

3. Methodology

While we are able to successfully formulate the IP formulations for both MMRP-0 and MMRP-1 by a combination of two different $\mathcal{NP}$-hard problems, it is less likely that we can solve practical instances of MMRP-0 and MMRP-1 to optimality by any exact methods in a limited amount of time, due to their complexity. As such, in this paper, an Adaptive Large Neighborhood Search (ALNS) heuristic is devised to help solve large instances of both MMRP-0 and MMRP-1.

3.1. Adaptive Large Neighborhood Search (ALNS)

The ALNS is an extension of the Large Neighborhood Search (LNS), introduced by Shaw (1997), where a large collection of variables are modified by several fast heuristics under an adaptive ruin-and-repair paradigm (Schrimpf, Schneider, Stamm-Wilbrandt, & Dueck, 2000), as illustrated in Algorithm 1 (Ropke & Pisinger, 2006a; Pisinger & Ropke, 2007).

Algorithm 1. The general framework of ALNS.

```
1:  Input: Problem Instances and the ALNS parameter setting.
2:  Create: Create an initial solution ($x$) and set $x_{best} \leftarrow x$.
3:  While stopping criteria have not been met do
4:      Select: Select $q$ requests to be destroyed and repaired.
5:      Destroy: Choose a destroy operator ($N^-$) based on its probability $\pi_{N^-}$.
6:      Repair: Choose a repair operator ($N^+$) based on its probability $\pi_{N^+}$.
7:      Evaluate: Evaluate new solution ($x'$).
8:      If $x'$ is accepted then
```

(continued on next column)

(continued)

```
9:          Update: $x \leftarrow x'$
10:         Update: Update probability $\pi_{N^-}$ and $\pi_{N^+}$
11:     End If
12:     If $f(x) < f(x_{best})$ then
13:         Update: $x_{best} \leftarrow x$
14:     End If
15: End While
16: Return: $x_{best}$.
```

In each iteration of the ALNS, q requests are removed and reinserted back to the incumbent solution by different destroy and repair heuristics, whose selection probabilities adaptively change based on their previous performances. These selection probabilities are initially set equally for all heuristics at the beginning of the search (segment); and, at the end of each iteration, they will be reassessed based on the quality of generated solution. While pairs of heuristics that provide new best-known or incremental improvement solutions should definitely be rewarded with higher selection probabilities, pairs of heuristics that provide inferior solutions are also eligible for rewards in the typical ALNS setting. This is due to the fact that they could help diversify search space and avoid being trapped at local extrema.

When compared to other well-known searches, such as the Variable Neighborhood Search (VNS), the ALNS is advantageous due to its flexibility with much less restrictions on neighborhood structure and parameter settings in the destroy-repair phase. In particular, the ALNS works only on a set of predefined removal and repair operators, whereas the VNS heavily relies on a highly structured neighborhood with variable depth (Pisinger & Ropke, 2007). The ALNS is found to be one of the promising metaheuristics in logistical domains, as it could improve best-known solutions of several standard benchmark Vehicle Routing Problem (VRP) instances as reported by Ropke and Pisinger (2006a, 2006b) and Pisinger and Ropke (2007). In addition, the ALNS has been successfully applied to a variety of VRP variants, including the Vehicle Routing Problem with Multiple Routes (Azi, Gendreau, & Potvin, 2014; Francois, Arda, & Crama, 2019), the Share-a-Ride Problem (Li, Krushinsky, Woensel, & Reijers, 2016), the Pickup and Delivery Problem with Time Windows and Scheduled Lines (Ghilas, Demir, & Woensel, 2016), the Mechanical Harvester Assignment and Routing Problem with Time Windows (Pitakaso & Sethanan, 2019), and the Two-Echelon Inventory Routing Problem with Perishable Products (Rohmer, Claassen, & Laporte, 2019), with different sets of destroy and repair operators that well suit the problems.

Hybridized algorithms between the ALNS and other well-known heuristic frameworks were also explored by Muller, Spoorendonk, and Pisinger (2012), Qu and Bard (2012) and Koc, Bektas, Jabali, and Laporte (2015). In Muller et al. (2012), the ALNS was combined with a Mixed Integer Programming (MIP) solver in the repairing phase of the Multi-Item Capacitated Lot Sizing Problem with Setup Times. Qu and Bard (2012), on the other hand, applied the ALNS along with the Greedy Randomized Adaptive Search Procedure (GRASP) to the Pickup and Delivery Problems with Transshipment, where the ALNS was called upon to improve initial solutions constructed by the GRASP in a two-phase fashion. Instead of using a Simulated Annealing (SA) framework for the search, Koc et al. (2015) have recently introduced a very interesting Hybrid Evolutionary Algorithm (HEA) that combined the concept of well-known Genetic Algorithms (GA) (Coello, Lamont, & Van Veldhuizen, 2007) with the ALNS framework to four different types of the Heterogenous Fleet Vehicle Routing Problems with Time Windows. In their proposed HEA, the initial population was first constructed based on a modified Clarke-Wright Savings algorithm with selected ALNS features. Once the number of initial solutions reached a predefined number (n_p), the evolutionary search was then executed through the GA framework, where the ALNS was applied as to intensify the set of elite solutions during the search.

3.2. Overall structure of the proposed ALNS

Our proposed ALNS differs from others found in the existing literature due to the MMRP's unique characteristics, where similarity values have played an important role in both construction and improvement phases of the algorithmic framework. Conceptually, the similarity value is a measure of music piece closeness in terms of both present and absent player sets — it could be regarded as an extension of the bit-flip concept proposed by Sakulsom and Tharmmaphornphilas (2014). More specifically, if A and $A^{'}$ denote a player-piece array and its bit-flip counterpart, the similarity array of music pieces in terms of player sets (S) could be defined by $A^T{\cdot}A + (A^{'})^T{\cdot}A^{'}$, where $A^T{\cdot}A$ presents the similarity array of music pieces in terms of present players, and, likewise, $(A^{'})^T{\cdot}A^{'}$ presents the similarity array of music pieces in terms of absent players.

In our ALNS setting, the similarity values between pairs of music pieces will be first calculated and used for the construction of an initial solution by a simple greedy heuristic. Once the initial solution is created, it will then undergo a series of adaptive destroy-repair procedure until one of the stopping criteria has been met. The initial weight and probability of each operator are set equally at the beginning of the search and adaptively updated based on its performance. More formally, δ_1 is awarded for pairs of operators providing new global best solutions (F_{best}), δ_2 is awarded for the pairs with incremental improvement, and, lastly, δ_3 is awarded for those with new solutions. It is worth noting that inferior solutions may be accepted in this framework as to avoid being stuck at local solutions by a Simulated Annealing (SA) approach. Algorithm 2 below illustrates the overall structure of the proposed ALNS, where two stopping criteria have been set: (*i*) the number of predefined iterations (It_{max}) has been reached and (*ii*) the best-known solution is not improved for a fixed number of iterations (It_w).

Algorithm 2. The overall structure of the proposed ALNS for MMRP-0 and MMRP-1.

```
1: Input: Problem Instances and the ALNS parameter setting.
2: Compute: Compute the Similarity array ($S$).
3: Create: Create an initial solution ($F_0$) by a simple greedy heuristic (IntGen) and set
   $F_0$ as the global best solution ($F_{best}$).
4: While stopping criteria have not been met do
5:     Select: Randomly select number of music pieces ($q$) to be removed from the
       incumbent solution.
6:     Destroy: Randomly select a destroy operator based on its weight and
       probability (Destroy).
7:     Repair: Randomly select a repair operator based on its weight and probability
       (Repair).
8:     Evaluate: Evaluate new solution ($F^{'}$).
9:     If $F^{'}$ is better than the current $F_{best}$ then
10:        Update: $F_{best} \leftarrow F^{'}$
11:    Else
12:        Select: Apply Simulated Annealing (SA) for the acceptance of $F^{'}$.
13:    End If
14:    Adjust: Adjust the weights, and so the probabilities, of all operators ($W - adj$).
15: End While
16: Return: The rehearsal schedule.
```

Based on Algorithm 2, there are several sub-computational modules required for the generation of rehearsal schedule as follows.

- *IntGen* is a sub-computational module for the creation of initial solution (F_0).
- *Destroy* and *Repair* are sub-computational modules that will be repeatedly called for solution improvement.
- *SA* is an escape mechanism that accepts inferior solutions based on the concept of Simulated Annealing.
- $W - adj$ is a weight adjustment module that updates the weights and so probabilities of all destroy and repair operators.

3.2.1. IntGen

IntGen generates an initial solution to the MMRP based on the similarity array (S), where an initial music piece is randomly selected and placed as the first performance of a day. Unassigned music pieces are then appended to the current music piece, one at a time, based on the similarity values with respect to such a piece. Nevertheless, the selected music piece is eligible only if its duration is less than or equal to the remaining rehearsal period on such a date. If this condition does not hold true, the next highest similarity music piece will be checked, or a new rehearsal day will be initiated. And, if it is the latter case, the whole process will be repeated over and over again until all pieces are assigned, as shown in Algorithm 3.

Algorithm 3. The detailed structure of *IntGen*.

```
1: Input: Problem Instances, Similarity Array ($S$), Unassigned Music Pieces ($P$), Empty
   Rehearsal Plan ($P_h$).
2: Initialization: Set the rehearsal period $d$ to 0.
3: While $P$ is not empty do
4:   Select: Randomly select $p_c \in P$ and set it as the first to be rehearsed.
5:   Update: $P \leftarrow P \backslash p_c, P_h \leftarrow p_c$, and $d \leftarrow d + d_{p_c}$.
6:   Sort: Sort $P$ based on the similarity value $S$ with respect to the last music piece of
     $P_h$, namely $p_c$, and let $\overline{P}$ be such a sorted list.
7:   While $d \neq 0$ and $d < Q$ do
8:     Select: Select the first element in $\overline{P}$, denoted by $\overline{p}_1$.
9:     If $d + d_{\overline{p}_1} \leq Q$ then
10:        Append: Append $\overline{p}_1$ to $P_h$.
11:        Update: $P \leftarrow P \backslash \overline{p}_1, P_h \leftarrow \overline{p}_1, p_c \leftarrow \overline{p}_1$, and $d \leftarrow d + d_{\overline{p}_1}$.
12:        Update: Update $\overline{P}$ with respect to $p_c$.
13:    Else
14:        If $\overline{p}_1$ is not the last in the list $\overline{P}$ then
15:            Update: Update $\overline{P}$ with $\overline{P} \backslash \overline{p}_1$.
16:        Else
17:            If $\overline{p}_1$ is the last in the list $\overline{P}$ then
18:                Open: Open a new rehearsal day and reset $d$ to 0.
19:            End If
20:        End If
21:    End If
22:  End While
23: End While
24: Return: The initial rehearsal schedule ($P_h$).
```

3.2.2. Destroy

There are three destroy operators in this proposed ALNS, each of which removes between q_l and q_u music pieces from the incumbent solution, as follows.

1. Random Removal (D_1): Random removal randomly removes music pieces until the predefined number of removals is reached.
2. Worst Removal (D_2): Worst Removal sequentially removes music pieces that give the maximum cost reduction when compared to that of the base solution, one at a time, until the predefined number of removals is reached.
3. Shaw Removal (D_3): Shaw removal focuses on the removals of related music pieces so that removals and repairs are easily executed (Shaw, 1998). In the literature, Shaw removal is usually defined by relatedness between elements i and j (γ_{ij}), where we define γ_{ij} by the similarity of player sets between music pieces i and j as shown in Eq. (28).

$$\gamma_{ij} = \varphi_1\left(A_{i.}^T{\cdot}A_{.j}\right) + \varphi_2\left(\left(A_{i.}^{'}\right)^T{\cdot}A_{.j}^{'}\right), \tag{28}$$

where A and $A^{'}$ are player-piece array and its bit-flip counterpart, and φ_1 and φ_2 are normalization weights, with $\varphi_1 = \varphi_2 = 1$.

Shaw Removal may be further defined based on the values of φ_1 and φ_2. For instance, if we are interested only in the change of present players, we may set $\varphi_1 = 1$ and $\varphi_2 = 0$, *i.e.* Present-Based Removal. On the contrary, we may set $\varphi_1 = 0$ and $\varphi_2 = 1$ in a case where emphasis has been put on the change of absent players (Absent-Based Removal) like that of Sakulsom and

Tharmmaphornphilas (2014).

Besides those three destroy operators, we also applied two local search heuristics, namely 2-exchange (*2EX*) and 3-exchange (*3EX*), during the destroy-repair phase.

1. 2-exchange (*2EX*): In each iteration of *2EX*, all music pieces from the i^{th} to the j^{th} positions are removed and reconnected in a reverse order. For example, given a rehearsal plan $(1,2,3,4,5,6)$, if music pieces 2 and 5 are selected under *2EX* operator, *2EX* will return a modified rehearsal plan $(1,5,4,3,2,6)$ as a new solution.
2. 3-exchange (*3EX*): In each iteration of *3EX*, three different music pieces will be selected and all or parts of their music strings will be swapped with the adjacent music strings. For instance, given a rehearsal plan $(1,2,3,4,5,6,7,8)$, if music pieces 1, 3, and 6 are selected under *3EX* operator, *3EX* will return a modified rehearsal plan $(2,1,4,3,5,7,6,8)$ as one plausible solution — there are about seven *3EX* solutions due to partial exchange.

3.2.3. *Repair*

There are four types of repair operators deployed in this ALNS, whose detailed information is described below.

1. Random Insertion (R_1): Random insertion randomly and sequentially places unassigned music pieces back to the incumbent solution until all pieces are scheduled.
2. Greedy Insertion (R_2): Greedy insertion places unassigned music pieces back to the incumbent solution, one at a time, in such a way that the incremental cost of rehearsal is minimized.

3 – 4. Regret-2 and Regret-3 Insertions ($R_3 - R_4$): The concept of Regret insertion is quite simple, as we first place an unassigned music piece that we will regret the most at its cheapest position; and, we continue in this fashion until all music pieces are assigned. Regret insertion may be regarded as an advanced greedy insertion with look-ahead information, known as a regret value (c_i^*) defined by Eq. (29).

$$c_i^* = \triangle f_{i,x_{(i,2)}} - \triangle f_{i,x_{(i,1)}}, \tag{29}$$

where $\triangle f_{i,x_{(i,1)}}$ and $\triangle f_{i,x_{(i,2)}}$ denote the rehearsal costs when music piece i is inserted at the best ($x_{(i,1)}$) and the second best ($x_{(i,2)}$) positions, respectively. In each iteration of Regret-2 insertion, an unassigned music piece with $\max_i c_i^*$ is first inserted at its best location — it should be placed now or else we will later regret not doing so — and the procedure continues until all pieces are assigned

For Regret-3 Insertion, the regret value is modified to Eq. (30), with the same insertion rule as Regret-2; and, based on Eq. (30), Regret-*n* insertion could be constructed by lifting the maximal value of j to n.

$$c_i^* = \sum_{j=1}^{3}\left(\triangle f_{i,x_{(i,j)}} - \triangle f_{i,x_{(i,1)}}\right) \tag{30}$$

It is worth noting that, when *2EX* and *3EX* are selected and executed, no repair operator is required. And, if the resulting solution is infeasible, no mark will be awarded to the selected destroy and repair operators.

3.2.4. *SA*

A standard SA procedure is applied for the acceptance of inferior solutions, where an inferior solution (F') is probably accepted with an acceptance rate of p_{accept} defined by Eq. (31).

$$p_{accept} = e^{-\frac{c(F')-c(F)}{T}}, \tag{31}$$

where F and $c(F)$ are the incumbent solution and its associated cost, and T is the temperature — initially set at T_{start} and gradually reduced by β % each iteration, *i.e.* the cooling rate (r_c) equals $1-\beta$.

3.2.5. *W − adj*

Initially, all destroy and repair operators, as well as *2EX* and *3EX*, are awarded with equal weight — and so probability. At the end of each iteration, the weights of the selected operators will be increased by δ_1 if the resulting solution leads to a new global best one and δ_2 if the resulting solution is better than the incumbent solution. We also award a mark of δ_3 to those generating a new solution, where $\delta_3 < \delta_2 < \delta_1$. However, no mark will be awarded if the resulting solution is infeasible. The awarded marks will be accumulated and used for the recomputation of selection probability from one iteration to the next until the algorithm terminates.

4. Computational results

4.1. *Instance and ALNS settings*

Six different problem settings — with 10 instances each — are generated for both MMRP-0 and MMRP-1. Each setting comprises of 10 players, $P \in \{10,12,14\}$ music pieces, and $D \in \{2,3\}$ rehearsal days. As such, a total of 120 instances will be generated as testbeds for the proposed ALNS. Tables 3 and 4 summarize all parameter values of the MMRP and the ALNS used in this research — some of which are set based on preliminary experimental runs, such as δ values, T, and r_c, while the rest are set based on previous research. When compared to Sakulsom and Tharmmaphornphilas (2014), our generated instances are comparatively larger with twice the number of involved players in most cases.

4.2. *Overall results on 120 generated instances*

The results from ALNS are compared with those of the IP as solved by CPLEX in terms of both solution quality and computational time; but, due to the MMRP complexity, the computation time of CPLEX is limited at four hours on a computer with 2.20 GHz Core2Duo processors and 4 GB of RAM. Based on this limit, CPLEX can find 29 optimal solutions out of 120 instances — mostly the MMRP-0 — while the maximum and average optimality gaps for the rest are 38.30% and 13.03%, respectively. Among these 29 instances, the ALNS can match 28 optimal solutions with significantly less computational time — about 8.82% of the time spent by CPLEX — while the only non-optimal solution is about 1.11% worse than the optimal objective value found by CPLEX. The comparison of computational times required by the ALNS and CPLEX on all 120 instances are also reported in Fig. 6.

In terms of solution quality, as measured by $\frac{C(ALNS)-C(CLEX)}{C(CPLEX)} \cdot 100\%$ — where $C(\Omega)$ denotes the objective value found by Ω — the ALNS can find the solutions that are as good as or better than those found by the time-restricted CPLEX in most cases (108 out of 120 instances), while the rest are within the maximal deviation of 5%, or about 1.36% on average, as

Table 3
Parameter settings for the MMRP.

Parameter	Definition	Value
I	Number of music pieces	$\{10,12,14\}$
P	Number of players	10
D	Number of rehearsal days	$\{2,3\}$
K	Maximum number of performance order in each rehearsal day	$\{10,12,14\}$
Q_{MMRP-0}	Daily available rehearsal limit for the MMRP-0	[18,45]
Q_{MMRP-1}	Daily available rehearsal limit for the MMRP-1	[20,48]
d_i	Duration of music piece $i \in I$	[3,9]
c_a	Player's daily wage	100
c_b	Penalty for waiting (player's hourly wage)	10
α	Setup parameter for the MMRP-1	0.1

Table 4
Parameter settings for the proposed ALNS.

Parameter	Definition	Value
δ_0	Initial weight awarded to all operators	10
δ_1	Weight awarded to operators that result in a new global best	3
δ_2	Weight awarded to operators that result in a better solution	2
δ_3	Weight awarded to operators that result in a new solution	1
q	Number of music pieces to be removed in each ALNS iteration	[2,4]
T_{start}	Initial temperature of the SA	100,000
r_c	Cooling rate of the SA, *i.e.* $\beta = 0.01$	0.99
It_{max}	Total number of ALSN iterations	100,000
It_w	Number of iterations with no improvement	10,000

illustrated in Fig. 7.

The detailed comparison of results from both ALNS and CPLEX on MMRP-0 and MMRP-1 instances is also reported in Tables 5 and 6. With regard to the MMRP-0 (Table 5), the ALNS can find 10 better solutions with an average incremental improvement of 1.68%, while the average deviation of the seven inferior solutions is just about 1.21%, with significantly less computational times in all cases. We also find that the ALNS tends to perform better in the complicated MMRP-1 settings (Table 6), as it provides 24 better solutions, with an average deviation of 3.17% better than solutions from the time-restricted CPLEX, while the average deviation of the five inferior ALNS solutions is just about 1.57%.

Regarding the destroy-repair operators, *3EX* is found to be the operator that contributes most significantly, as it possesses the highest selection probability at the end of the search for most instances (85 out of 120 instances). Additionally, *3EX* is also found to be the most efficient destroy-repair operator as it can find the global best solutions for about 46 instances, followed by *2EX* in 43 instances, leaving D_1 and R_1 as the worst destroy and repair operators with the least selection probabilities, respectively.

4.3. Results on benchmark MMRP-0 instances

To further assess the performance of our proposed ALNS, 78 additional experiments are conducted based on instances from Sakulsom and Tharmmaphornphilas (2014) (MMRP-0), whose settings are summarized in Table 7.

Among these 78 benchmark instances, Instances 1 to 70 are regarded as small instances with only five players, while Instances 71 through 77 are larger instances with twice the number of players; lastly, Instance 78 is a very large instance involving with 20 players, 40 music pieces, and a comparatively long rehearsal period of five days. The comparison of

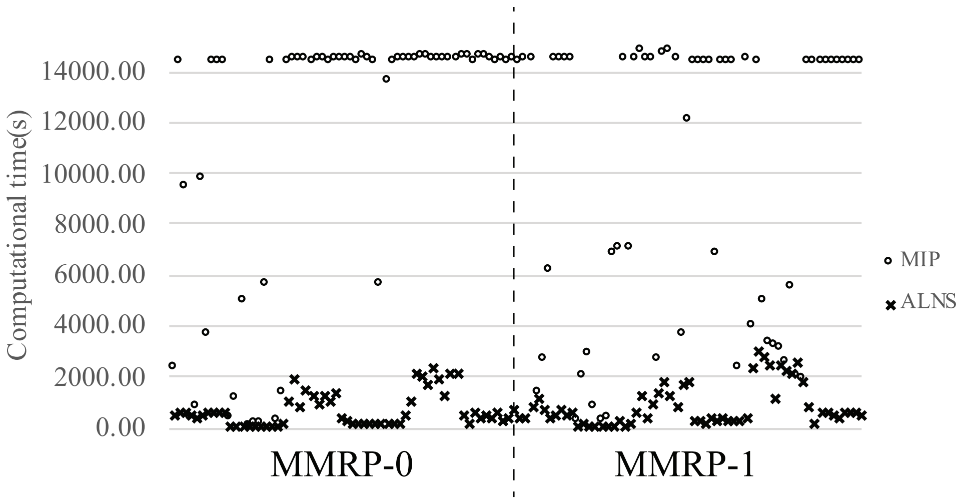

Fig. 6. Comparison of computational times required by both ALNS and CPLEX on all 120 instances.

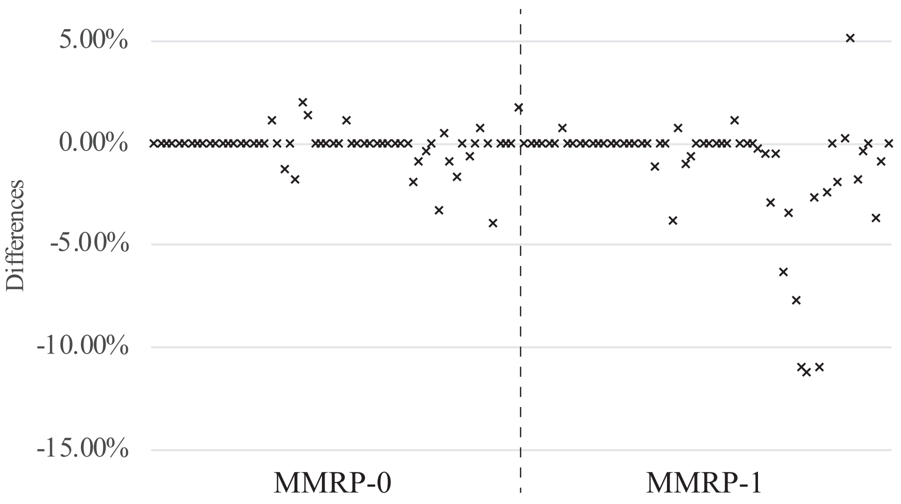

Fig. 7. Percentages of solution deviations between the ALNS and CPLEX on all 120 instances.

Table 5
The computational results for MMRP-0.

Number of			Average elapsed time (s)			Number of solutions when compared to CPLEX		
Players	Pieces	Days	CPLEX	ALNS	Diff (%)	Equal	Better[1]	Worse[2]
10	10	2	8426.57	554.36	93.42	10	0	0
10	12	2	2877.58	111.89	96.11	9	0	1
10	14	2	14490.34	1165.27	91.96	6	2	2
10	10	3	13515.90	207.06	98.47	9	0	1
10	12	3	14535.98	1736.95	88.05	3	6	1
10	14	3	14516.78	470.28	96.76	6	2	2

[1]The percentage of solution deviation is about 1.68% better than CPLEX on average.
[2]The percentage of solution deviation is about 1.21% worse than CPLEX on average.

Table 6
The computational results for MMRP-1.

Number of			Average elapsed time (s)			Number of solutions when compared to CPLEX		
Players	Pieces	Days	CPLEX	ALNS	Diff (%)	Equal	Better[1]	Worse[2]
10	10	2	11152.45	646.91	94.20	9	0	1
10	12	2	4231.03	131.12	96.90	10	0	0
10	14	2	12093.33	1217.47	89.93	3	6	1
10	10	3	12447.29	309.25	97.52	7	2	1
10	12	3	4513.66	2323.49	48.52	0	10	0
10	14	3	14412.63	563.32	96.09	2	6	2

[1]The percentage of solution deviation is about 3.17% better than CPLEX on average.
[2]The percentage of solution deviation is about 1.57% worse than CPLEX on average.

Table 7
The settings of MMRP-0 benchmark instances from Sakulsom and Tharmmaphornphilas (2014).

Instance number	Number of			Number of instances
	Players	Pieces	Days	for each setting
1–30	5	$\{10, 12, 14\}$	2	10
31–70	5	$\{10, 12, 14, 16\}$	3	10
71–73	10	$\{10, 12, 14\}$	2	1
74–77	10	$\{10, 12, 14, 16\}$	3	1
78	20	40	5	1

results between the ALNS and the benchmark approach by Sakulsom and Tharmmaphornphilas (2014), *i.e.* 2-phase method, on these 78 benchmark instances is summarized in Tables 8 and 9.

From Table 8, it can be seen that the proposed ALNS is comparatively efficient for small to moderate instances as it provides the solutions that are as good as those found by the 2-phase heuristic — with smaller computational times for larger instances. Although there are seven instances that the ALNS performs a bit worse, the percentage of solution deviation among these instances is just about 1.04% on average.

The ALNS tends to perform much better in large and complicated benchmark instances (Instances 71–78), with no inferior solution reported, as shown in Table 9. In particular, the ALNS can find three new best known solutions to Instances 71, 75, and 76. And, for a very large problem (Instance 78), the 2-phase heuristic spent almost 6.30 h just to determine the minimum number of show-up days, while the ALNS is able to find a complete solution within one hour, although the number of show-up days is slightly worse — about 1.08% when compared to that of the 2-phase heuristic.

5. Conclusions

We have proposed an ALNS framework to solve two variants of the MMRP, namely the MMRP without setup times (MMRP-0) and the MMRP with setup times (MMRP-1), where a setup time occurs whenever there is a change on player sets between two consecutively scheduled music pieces. The proposed ALNS is found to be greatly efficient as it requires comparatively less computational time to produce equivalent or better solutions than the exact method and a benchmark heuristic from the literature. In particular, out of 120 generated instances, the ALNS can find 34 better solutions and match 74 others from the CPLEX solver with an average computational time of 16.63% — while the average solution deviation of 12 inferior solutions is just about 1.36%. Regarding those 78 benchmark instances, the ALNS tends to outperform the benchmark heuristic, especially on large benchmark instances. More specifically, out of seven large benchmark instances, the ALNS can find three new best known solutions. And, for the largest benchmark instance, a complete rehearsal schedule can be successfully determined within an hour of computation time, while the benchmark heuristic

Table 8
The computational results of ALNS when compared to those of Sakulsom and Tharmmaphornphilas (2014) on Instances 1–70.

Instance	Number of			Average elapsed time (s)		Number of solutions when compared to 2-phase method		
	Players	Pieces	Days	2-Phase	ALNS	Equal	Better	Worse[1]
1–10	5	10	2	28.81	98.06	10	0	0
11–20	5	12	2	141.38	163.70	10	0	0
21–30	5	14	2	425.16	319.64	8	0	2
31–40	5	10	3	114.87	45.23	10	0	0
41–50	5	12	3	53.00	78.75	9	0	1
51–60	5	14	3	220.68	118.94	9	0	1
61–70	5	16	3	546.44	151.76	7	0	3

[1]The percentage of solution deviation is about 1.04% worse than the 2-phase method on average.

Table 9
The computational results of ALNS when compared to those of Sakulsom and Tharmmaphornphilas (2014) on Instances 71–78.

Instance	Number of			Elapsed Time (s)		Show-up Days			Waiting Time Slots		
	Players	Pieces	Days	2-Phase	ALNS	2-Phase	ALNS	Diff (%)	2-Phase	ALNS	Diff (%)
71	10	10	2	119.56	386.14	19	19	0	7	5	28.57
72	10	12	2	171.06	427.03	18	18	0	27	27	0
73	10	14	2	2008.06	875.74	19	19	0	23	23	0
74	10	10	3	121.19	133.86	25	25	0	3	3	0
75	10	12	3	371.12	197.18	27	27	0	5	0	100
76	10	14	3	397.54	222.31	25	25	0	17	9	47.06
77	10	16	3	671.75	691.34	25	25	0	19	19	0
78	20	40	5	22398.02 [1]	3799.69	93	94	−1.08	–[2]	239	–

[1]This reported computational time was only from the first phase of the 2-phase heuristic.
[2]No result on waiting time slot was reported by Sakulsom and Tharmmaphornphilas (2014) due to prohibitive computational time during the scheduling phase.

spent a lot more time just to determine the minimum day of attendance with no reported schedule.

While our focus lies on a typical rehearsal problem, with no specific restrictions on both music pieces and players, there are several practical constraints that could be incorporated into the MMRP, and its FPSP counterpart, for more realistic planning. Examples include music piece (shooting scene) precedence constraints, compatibility constraints between music pieces and rehearsed periods (shooting periods), player preference or player availability constraints, and rehearsal place (shooting place) related constraints, such as unequal daily rent and availability. We may also introduce dynamism into the problem, or even extend this present work to related problems in other domains, which will further enrich the resulting problems — and so the development of algorithmic framework in subsequent studies. These plausible extensions include the operational decisions in a flexible manufacturing system (FMS) — *e.g.* the Process Plan Selection Problem (PPSP) or the Machine Loading Problem (MLP) investigated by Solimanpur, Sattari, and Abazari (2012), Abazari et al. (2012) and Singh, Singh, and Khan (2016) — as they share some common characteristics with the MMRP. For instance, jobs, machines, and production schedules in an FMS may be viewed as music pieces, rehearsal days, and rehearsal schedules in the MMRP setting, respectively; although further detailed mapping is needed between the stated problems. In light of this observation, we expect that this present work could be adapted and applied to those related problems, which, in turn, opens new ideas for both researchers and practitioners in other relevant fields.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by the Research Grant for New Scholar Ratchadaphiseksomphot Endowment Fund Chulalongkorn University, as well as the EU Horizon 2020 Marie Sklodowska-Curie Research and Innovation Staff Exchange project GOLF (reference number GOLF-777742).

References

Abazari, A., Solimanpur, M., & Sattari, H. (2012). Optimum loading of machines in a flexible manufacturing system using a mixed-integer linear mathematical programming model and genetic algorithm. *Computers & Industrial Engineering, 62*, 469–478.
Adelson, R., Norman, J., & Laporte, G. (1976). A dynamic programming formulation with diverse applications. *Operational Research Quarterly, 27*(1), 119–121.
Azi, N., Gendreau, M., & Potvin, J. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research, 41*, 167173.
Bellman, R. (1957). *Dynamic programming*. New York: Princeton University Press.
Bomsdorf, F., & Derigs, U. (2008). A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. *OR Spectrum, 30*, 751–772.
Cheng, T., Diamond, J., & Lin, B. (1993). Optimal scheduling in film product to minimize talent hold cost. *Optimization Theory and Application, 79*(3), 479–492.
Coello, C., Lamont, G., & Van Veldhuizen, D. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.
de la Banda, M., Stuckey, P., & Chu, G. (2011). Solving talent scheduling with dynamic programming. *INFORMS Journal on Computing, 23*(1), 120–137.
Francois, V., Arda, Y., & Crama, Y. (2019). Adaptive large neighborhood search for multitrip vehicle routing with time windows. *Transportation Science, 63*(6), 17061730.
Ghilas, V., Demir, E., & Woensel, T. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research, 72*, 12–30.
Koc, C., Bektas, T., Jabali, O., & Laporte, G. (2015). A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. *Computers & Operations Research, 64*, 11–27.
Lenstra, J., & Rinnooy Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11*(2), 211–227.
Li, B., Krushinsky, D., Woensel, T., & Reijers, H. (2016). An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers & Operations Research, 66*, 170–180.
Muller, L., Spoorendonk, S., & Pisinger, D. (2012). A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research, 218*, 614–623.
Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*, 24032435.
Pitakaso, R., & Sethanan, K. (2019). Adaptive large neighborhood search for scheduling sugarcane inbound logistics equipment and machinery under a sharing infield resource system. *Computers and Electronics in Agriculture, 158*, 313–325.
Qin, H., Zhang, Z., Lim, A., & Liang, X. (2016). An enhanced branch-and-bound algorithm for the talent scheduling problem. *European Journal of Operational Research, 250*, 412–426.
Qu, Y., & Bard, J. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research, 39*, 2439–2456.
Rohmer, S., Claassen, G., & Laporte, G. (2019). A two-echelon inventory routing problem for perishable products. *Computers & Operations Research, 107*, 156–172.
Ropke, S., & Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455–472.
Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research, 171*, 750–775.
Sakulsom, N. & Tharmmaphornphilas, W. (2011). A multi-objective music rehearsal scheduling problem. In *The 12th Asia Pacific industrial engineering and management systems conference* (pp. 25–29).
Sakulsom, N., & Tharmmaphornphilas, W. (2014). Scheduling a music rehearsal problem with unequal music piece length. *Computers & Industrial Engineering, 70*, 20–30.
Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results – using the ruin & recreate principle. *Journal of Computational Physics, 159*, 139–171.
Shaw, P. (1997). *A new local search algorithm providing high quality solutions to vehicle routing problems*. Technical report, Glasgow: University of Strathclyde.
Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *The 4th International conference on principles and practice of constraint programming*.
Singh, R., Singh, R., & Khan, B. (2016). Meta-hierarchical-heuristic-mathematical-model of loading problems in flexible manufacturing system for development of an intelligent approach. *International Journal of Industrial Engineering Computations, 7*, 177–190.
Smith, B. (2003). *Constraint programming in practice: Scheduling a rehearsal*. Technical report, Report APES-67-2003. http://www.dcs.st-and.ac.uk/apes.
Solimanpur, M., Sattari, H., & Abazari, A. (2012). Optimum process plan selection via branch-and-bound algorithm in an automated manufacturing environment. *International Journal of Operational Research, 13*(3), 281–294.
Wang, S., Chuang, Y., & Lin, B. (2016). Minimizing talent cost and operating cost in film production. *Journal of Industrial and Production Engineering, 33*(1), 17–31.