Computers & Industrial Engineering 70 (2014) 20–30

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Scheduling a music rehearsal problem with unequal music piece length ☆

Noppadon Sakulsom, Wipawee Tharmmaphornphilas *

Department of Industrial Engineering, Chulalongkorn University, Thailand

ARTICLE INFO

Article history:
Received 7 September 2012
Received in revised form 3 December 2013
Accepted 31 December 2013
Available online 11 January 2014

Keywords:
Scheduling
Music rehearsal
Mixed integer programming
Cell formation
Unequal piece length

ABSTRACT

This paper studies a music rehearsal scheduling where music pieces require different sets of players and different rehearsal durations. The rehearsal is arranged in multiple days. The players need to show up only on the day that the pieces they play are scheduled. They must show up before the first piece they involve starts and leave after the last one ends. The objectives are to assign music pieces to the rehearsal days to minimize the total number of days that all players have to show up and sequence the music pieces within each day to minimize the total waiting time of the players. We propose a 2-stage methodology to schedule music pieces, which is a combination of a cell formation technique and an integer-programming model. From 77 test problems, the solutions from the proposed method are as good as the optimal solutions from MIP models and are better than the best founds in all the test problems. The computational time is also significantly less.

1. Introduction

To prepare for a concert, a music band needs to rehearse many music pieces several times. Generally, music pieces require different sets of players and different rehearsal durations. We consider the problem where the rehearsal period is longer than one day. To schedule the rehearsal, music pieces are assigned to rehearsal days to minimize the number of days that players need to show up. Music pieces are also sequenced in each day to minimize the total idle (non-playing) time when players are at the rehearsal. We assume that players show up only on the day that their pieces are rehearsed, and must arrive by the time the first piece they play starts and leave after the last piece they play finishes. Therefore, if a player plays two music pieces, it is better to assign them to the same day and schedule them as close as possible to meet the objectives.

A music band rehearsal scheduling problem has similar characteristics as a film production scheduling problem first introduced by Cheng, Diamond, and Lin (1993). A film producer needs to schedule scenes to be shot when they are on location. Each scene requires a different set of talents to be in. The talents are paid for each day they are in the location starting from the day that the first scene they are involved is shot to the day that the last scene they are involved is shot. Since there are some days that the talents are on the location without joining the scenes, called "hold days", the objective of this problem is to minimize a cost due to those hold days. It is assumed that the cost of each talent is different. The first approach to solve the film production problem was a branch-and-bound algorithm. However, since this problem is strongly NP-hard (Cheng et al., 1993), a heuristic algorithm was developed to reduce the computational time. Nordström and Tufekci (1994) proposed an algorithm combining a limited pairwise interchange heuristic procedure with the simple genetic algorithm framework to solve the problem proposed by Cheng et al. (1993). Their algorithm outperformed the heuristic proposed by Cheng et al. (1993) in terms of quality of solutions and computational time. Garcia de la Banda, Stuckey, and Chu (2011) applied a dynamic programming to the film production problem. The dynamic programming could find optimal solutions to large problems faster than the approach developed by Cheng et al. (1993). Kochetov (2011) developed three new heuristic algorithms including a hybrid simulated annealing algorithm, a stochastic tabu search algorithm and a genetic local search algorithm to solve the same problem. The proposed heuristics can find the optimal solutions within a few minutes. Bomsdorf and Derigs (2008) studied a movie shoot scheduling problem. They considered realistic constraints such as working time, scene precedence and resource availability and also various objectives such as minimizing makespan and minimizing resource costs. A greedy indirect search technique was applied to solve the problem. The proposed method provided schedules faster and better than manual scheduling method.

Smith (2003) adapted the film production scheduling problem to an orchestra band scheduling problem. Smith studied a music band rehearsal problem with nine music pieces having different

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng, ScD, PhD.

* Corresponding author. Tel.: +66 (0) 2218 6829; fax: +66 (0) 2218 6413.
E-mail addresses: nop_i7v2@hotmail.com (N. Sakulsom), wipawee.t@chula.ac.th (W. Tharmmaphornphilas).

0360-8352/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2013.12.017

lengths. Each piece requires a different set of players from five members of the orchestra. Same as scenes in the film production scheduling problem, the music pieces are sequenced to minimize the total time that players are waiting at the rehearsal while pieces that they do not play are rehearsed. This problem was solved as a constraint satisfaction problem. Gregory, Miller, and Prosser (2004) applied planning and model checking techniques to the music band rehearsal scheduling problem. They compared these two techniques with the constraint programming proposed in Smith (2003). The result showed that the planning technique outperformed both constraint programming and model checking technique in terms of computational time.

All previous music band rehearsal problems considered the case where music pieces are continuously sequenced from the first to the last pieces. Sakulsom and Tharmmaphornphilas (2011) considered arranging the rehearsal into multiple days, and the objective is to minimize the number of days that players need to show up. Also, within each day, they try to minimize the player waiting time. It is assumed that each music piece has the same rehearsal duration of a single slot.

Unlike the problem considered by Sakulsom and Tharmmaphornphilas (2011), we assume that music pieces have different rehearsal durations. There are limited time slots to rehearse in each rehearsal day and also a limited number of rehearsal days. Therefore, improper assigning music pieces to rehearsal days may lead to an infeasible solution. Players are paid only on the day that they show up at the rehearsal. Different from Cheng et al. (1993), we assume that the cost of each player is the same. Therefore, to minimize the cost, the objective is finding the way to minimize the number of days that players must show up. Moreover, to increase players' satisfaction, the second objective is to minimize player waiting time within a show-up day.

2. Problem statement

We consider a music rehearsal involving multiple rehearsal days with equal rehearsal time slots. Each music piece has a different length and requires a different set of players. Players have to show up at the rehearsal only on the days that the pieces they play are rehearsed. They have to arrive at the rehearsal immediately before the first piece they play starts, and can leave the rehearsal immediately after the last piece they play ends. An assignment of music pieces to each rehearsal day and a sequence of music pieces within a day must be determined to minimize the total number of days that the players must be in the rehearsal and also to minimize the total waiting time during the rehearsal days. We focus on minimizing the total number of show-up days as a primary objective and minimizing the total waiting time as a secondary objective. Therefore, two mixed integer programming models are developed to solve this problem. The first model is used to find the minimum number of show-up days. This optimal value is then input into the second model as one of its constraints. Note that a multi-objective model can replace these two models. Since the number of show-up days is the main concern, a significantly higher weight must be given to this objective. However, the optimal solutions from these 2 models and a multi-objective model are the same. The MIP models are described in the following sections.

2.1. A mixed integer programming model for assigning unequal music pieces to rehearsal days

This integer programming model is developed to assign unequal music pieces to rehearsal days, with the objective function to minimize the total number of days that players must be in the rehearsal.

Indices

I	= {1, 2, . . ., n} is a set of time slots in each rehearsal day
J	= {1, 2, . . ., m} is a set of music pieces
K	= {1, 2, . . ., p} is a set of players
D	= {1, 2, . . ., q} is a set rehearsal days

Parameters

$play_{kj}$	= 1 if player k plays piece j, 0 otherwise
$duration_j$	= the duration of piece j
M	= a large number that is greater than the number of slots in each rehearsal day

Decision variables

$playslot_{dki}$	= 1 if player k is required to play during slot i of day d, 0 otherwise
$timetable_{dij}$	= 1 if piece j is rehearsed during slot i of day d, 0 otherwise
$timestart_{dij}$	= 1 if piece j starts rehearsing in or before slot i of day d, 0 otherwise
$timeend_{dij}$	= 1 if piece j finishes rehearsing in or after slot i of day d, 0 otherwise
$stay_{dk}$	= 1 if player k is required to be in the rehearsal on day d, 0 otherwise
y_{dj}	= 1 if piece j is assigned to be rehearsed on day d, 0 otherwise
$z1$	= The total number of days that players are in the rehearsal

Objective function (1) is to minimize the number of days that players are in the rehearsal.

$$\text{Minimize} \quad z1 = \sum_{d=1}^{q}\sum_{k=1}^{p} stay_{dk} \tag{1}$$

Constraints (2) force to schedule only one music piece at a time. Constraints (3) assign each music piece to only one rehearsal day. Constraints (4) reserve rehearsal slots to complete that piece and constraints (5) assign players to the slots that they are required to play.

$$\sum_{j=1}^{m} timetable_{dij} \leqslant 1 \quad \forall\, d \in D,\ \forall\, i \in I \tag{2}$$

$$\sum_{d=1}^{q} y_{dj} = 1 \quad \forall\, j \in J \tag{3}$$

$$\sum_{i=1}^{n} timetable_{dij} = y_{dj} \times duration_j \quad \forall\, d \in D,\ \forall\, j \in J \tag{4}$$

$$playslot_{dki} = \sum_{j=1}^{m} timetable_{dij} \times play_{kj} \quad \forall\, d \in D,\ \forall\, k \in K,\ \forall\, i \in I \tag{5}$$

Constraints (6)–(10) force to rehearse each music piece without dividing it into parts. Constraints (6)–(8) are used to assign values to the decision variables $timestart_{dij}$, $timeend_{dij}$, $timetable_{dij}$. Constraints (9) and (10) force each piece to start rehearsing at the first

slot of the assigned timetable and finish rehearsing at the last slot of the assigned timetable.

$$timetable_{dij} \geqslant timestart_{dij} + timeend_{dij} - 1 \quad \forall d \in D, \forall i \in I, \forall j \in J \tag{6}$$

$$timestart_{dij} \leqslant timestart_{d(i+1)j} \quad \forall d \in D, \ \forall i \in \{1, \ldots, n-1\}, \ \forall j \in J \tag{7}$$

$$timeend_{dij} \geqslant timeend_{d(i+1)j} \quad \forall d \in D, \ \forall i \in \{1, \ldots, n-1\}, \ \forall j \in J \tag{8}$$

$$timestart_{dij} \geqslant timetable_{dij} \quad \forall d \in D, \ \forall i \in I, \ \forall j \in J \tag{9}$$

$$timeend_{dij} \geqslant timetable_{dij} \quad \forall d \in D, \ \forall i \in I, \ \forall j \in J \tag{10}$$

For example, If a piece start at the beginning of slot 2 and its duration is 4 slots, so it ends at the end of slot 5 the values of $timetable_{dij}$, $timestart_{dij}$ and $timeend_{dij}$ are as shown in Table 1.

Constraints (11) determine the presence of player k on the rehearsal day d.

$$\sum_{i=1}^{n} playslot_{dki} \leqslant stay_{dk} \times M \quad \forall d \in D, \ \forall k \in K \tag{11}$$

2.2. A mixed integer programming model for sequencing music pieces

This integer programming model is developed to sequence music pieces within each rehearsal day given the minimum number of days that players need to show up, which can be found from the integer programming model in Section 2.1. All constraints (Constraints (2)–(11)) from the first model still remain in this second model. However, to sequence the pieces, additional decision variables and constraints are included into the model. Also, the objective function is changed to minimize the total waiting time of the players.

Additional decision variables

r_{dki}	= 1 if player k is at the rehearsal during slot i of day d, 0 otherwise
a_{dki}	= 1 if player k arrives in or before slot i of day d, 0 otherwise
l_{dki}	= 1 if player k leaves in or after slot i of day d, 0 otherwise
w_{dki}	= 1 if player k is at the rehearsal during slot i of day d but does not play, 0 otherwise
$z2$	= The total waiting time.

Objective function (12) is to minimize the total time that players are in the rehearsal but not play.

$$\text{Minimize} \quad z2 = \sum_{d=1}^{q}\sum_{k=1}^{p}\sum_{i=1}^{n} w_{dki} \tag{12}$$

Constraints (13)–(15) determine the presence of each player in each rehearsal slot.

$$r_{dki} \geqslant a_{dki} + l_{dki} - 1 \quad \forall d \in D, \ \forall k \in K, \ \forall i \in I \tag{13}$$

$$a_{dki} \leqslant a_{dk(i+1)} \quad \forall d \in D, \ \forall k \in K, \ \forall i \in \{1, \ldots, n-1\} \tag{14}$$

$$l_{dki} \geqslant l_{dk(i+1)} \quad \forall d \in D, \ \forall k \in K, \ \forall i \in \{1, \ldots, n-1\} \tag{15}$$

For example, if a player arrives at the rehearsal at the beginning of slot 3 and leaves at the end of slot 7 the values of a_{dki}, l_{dki} and r_{dki} are as shown in Table 2.

Constraints (16) and (17) force each player to arrive at the beginning of the first slot that s/he plays and leave at the end of the last slot that s/he plays.

$$a_{dki} \geqslant playslot_{dki} \quad \forall d \in D, \ \forall k \in K, \ \forall i \in I \tag{16}$$

$$l_{dki} \geqslant playslot_{dki} \quad \forall d \in D, \ \forall k \in K, \ \forall i \in I \tag{17}$$

Constraints (18) determine if player k has to wait in slot i or not.

$$r_{dki} - w_{dki} \leqslant playslot_{dki} \quad \forall d \in D, \ \forall k \in K, \ \forall i \in I \tag{18}$$

Constraint (19) forces the total number of days that players are in the rehearsal to be the minimum number of days found from the previous MIP model in Section 2.1.

$$\sum_{d=1}^{q}\sum_{k=1}^{p} stay_{dk} \leqslant z1 \tag{19}$$

3. Methodology

Since sequencing music pieces with rehearsal days is an NP-hard problem (Cheng et al., 1993), the computational time that the MIP models use to find the optimal solution grows rapidly as the number of music pieces increases as shown in Fig. 1. Therefore, if we can manage to solve the problem with only one rehearsal day, it will reduce the computational time significantly. According to this concept, we developed a 2-stage decision methodology to solve the problem. The first stage is to assign music pieces to rehearsal days to reduce the number of days that players need to show up. With the solutions from this stage, we can switch from a multiple rehearsal days problem to a single rehearsal day but many problems in the second stage. Using this method, we expect to find the optimal solution faster. However, there may be multiple optimal solutions from the first stage, which lead to different solutions in the second stage. A heuristic algorithm based on a cell formation algorithm was developed to find the alternative optimal solutions. These solutions are sent to an integer programming model in the second stage to develop the sequence of music pieces in each rehearsal day separately. Note that the proposed methodology is different from consecutively solving 2 MIP models in Section 2, where only optimal z-value (not optimal solution) from the first MIP model becomes a constraint for the second MIP model. Therefore, the solution from the second model in Section 2 (if it can be obtained) is the global optimal solution.

Table 1
Values of $timetable_{dij}$, $timestart_{dij}$ and $timeend_{dij}$.

Slot	1	2	3	4	5	6	7	8	9
timestart	0	1	1	1	1	1	1	1	1
timeend	1	1	1	1	1	0	0	0	0
timetable	0	1	1	1	1	0	0	0	0

Table 2
Values of a_{dki}, l_{dki} and r_{dki}.

Slot	1	2	3	4	5	6	7	8	9
a	0	0	1	1	1	1	1	1	1
l	1	1	1	1	1	1	1	0	0
r	0	0	1	1	1	1	1	0	0

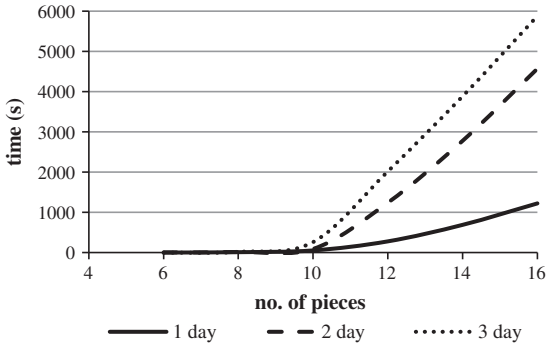

Fig. 1. The average computational time from MIP models versus the number of music pieces.

3.1. Heuristics to assign music pieces to rehearsal days

We found that a method to assign music pieces to rehearsal days is similar to a method to assign parts to cells in a well-known cell formation problem. In a cellular manufacturing, products are arranged into groups and assigned to cells to reduce setup time, in-process inventories, increase product quality and improve productivity. Each cell must contain machines required to produce all products that are assigned to that cell. Assigning music pieces to rehearsal days is like assigning parts to cells where a set of players required by those pieces is analogous to a set of machines required by parts. The number of parts assigned to each cell is limited by a cell capacity, which is similar to the number of pieces assigned to each rehearsal day is limited by the available slots in that day. The objective of assigning music pieces to rehearsal days that is minimizing the total number of days that players have to be at the rehearsal is like minimizing the number of machines required in the production system.

McCormick, Schweitzer, and White (1972) developed Bond Energy Algorithm (BEA) to solve machines and parts grouping problem. The concept of BEA is to calculate the Measure of Effectiveness (ME), which is the index that shows similarity of parts in terms of using machines. A sequential-selection suboptimal algorithm was then used to arrange parts to get the highest ME arrangement. The higher ME leads to the better part grouping. Boe and Cheng (1991) developed "Close Neighbor Algorithm" based upon the concept of BEA. Close Neighbor Algorithm calculated the "closeness" between each pair of machines from the number of parts, which involved in these machines. Then machines are arranged based upon their closeness. Moon and Gen (1999) applied a genetic algorithm to minimize cost of the design of independent manufacturing cells, which is the sum of the machining and duplicating costs. The duplicating cost is the cost due to duplication of bottleneck machines. Mak, Wong, and Wang (2000) proposed another genetic algorithm based heuristics to group parts and machines into clusters. The heuristics sequenced the rows and columns of a machine-part incidence and evaluated the solution with the bond energy measure of the matrix. Wu, Chu, Wang, and Yan (2007) developed a genetic algorithm to minimize the total cost due to movements within cells and between cells and also minimize exceptional elements. They also studied how GA parameters such as crossover rate, mutation rate or population size affected the algorithm performance. Tariq, Hussain, and Ghafoor (2009) proposed a combined local search to a genetic algorithm. This algorithm was used to minimize movements between cells and maximize the within-cell machine utilization. Paydar and Saidi-Mehrabad (2013) proposed a mathematical model and a hybrid genetic algorithm to solve the cell formation problem with an unknown number of cells. Grouping efficacy was used as an objective function to be maximized. Chung, Wu, and Chang (2011) studied a cell formation problem considering alternative process routings and machine reliability. They proposed a tabu search algorithm combined with the mutation operation, which is commonly used in a genetic algorithm, to solve the problem. There exist several methodologies that can be applied to formulate cells in a cell formation problem. However, to the best of our knowledge, none of them has applied those algorithms to a music scheduling problem except Sakulsom and Tharmmaphornphilas (2011).

Sakulsom and Tharmmaphornphilas (2011) proposed a methodology for the music band rehearsal-scheduling problem, where they assumed a single slot for all music piece lengths. Therefore, the number of music pieces that are assigned to each rehearsal day is less than or equal to the number of slots in that day. Due to unequal music piece length in this paper, a heuristic algorithm developed by Sakulsom and Tharmmaphornphilas (2011) cannot be applied directly. For example, 12 unequal length music pieces are assigned to rehearsal days. Assume that each rehearsal day has 18 available slots. The total duration of 12 music pieces is 36 slots; therefore, at least 2 rehearsal days are required. Applying an algorithm in Sakulsom and Tharmmaphornphilas (2011), the solution can be shown in Table 3, which the total duration in the first rehearsal day is 20 slots exceeding available rehearsal time. However, if we assign music pieces by considering their lengths properly, the solution can be shown in Table 4, where the total duration is 18 slots in both rehearsal days, which fit in the available rehearsal time.

3.1.1. Minimum rehearsal days

Since the rehearsal time slots in each day are limited, we must determine the minimum number of rehearsal days that is feasible to assign music pieces. First, the number of rehearsal days is found by dividing the total duration of all music pieces by the rehearsal time available in each day. Then, an algorithm for minimizing makespan proposed by Baker (1974) is performed to verify the feasibility of this number of rehearsal days. Music pieces are sequenced via the longest processing time (LPT) rule and assigned to the day with the least total time used. If the total time used in any day is longer than the available time, a piece in that day will be swapped with a piece in a day that total time used is shorter than the available time. After all possible pairs are swapped, if the schedule is infeasible, the number of rehearsal days will increase by 1 and the algorithm will be repeated again. The process is continued until a feasible solution is found.

3.1.2. Assignment of music pieces

Heuristics to assign unequal length music pieces to rehearsal days is improved from heuristics to assign equal length music pieces to rehearsal days in Sakulsom and Tharmmaphornphilas (2011). The heuristics is applied a concept of cell formation. How-

Table 3
A solution from a model developed for an equal music piece length problem.

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

Table 4
A solution from a model developed for an unequal music piece length problem.

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

ever, instead of grouping pieces that require the same set of players into the same day, we focus on grouping pieces that do not require the same set of players into the same day so that those players do not need to show up on that day.

Since the heuristics focuses on players who are not involved in music pieces, **bit-flip** is done in the piece-player array. Normally, 1 in this piece-player array means players play those pieces and 0 means players are not involved in those pieces. After the bit-flip, the meaning is opposite. The bit-flipped array of the previous example problem is shown in Table 5. Then, **player coefficient** is assigned. We search for players who are possible to have at least one free day and a coefficient of 1 is assigned to those players. For example in Table 5, since each rehearsal day has available rehearsal time of 18 slots, players who are possible to have a free day must not involve in at least 18 slots. Player 3 does not involve in 18 slots so he is assigned a coefficient of 1. On the contrary, player 1 does not involve in only 13 slots. This player must attend both rehearsal days so he is assigned a coefficient of 0. To minimize the number of days that players need to attend the rehearsal, we focus on only players who are possible to be free at least one day. Therefore, player coefficients and bit-flip piece-player assignments are multiplied. The modified array is shown in Table 6.

Then, the **ClosenessCalculation** between each pair of pieces is performed. Based on Close Neighbor Algorithm (Boe & Cheng, 1991), closeness is the sum of the product of each element belonging to a pair of pieces in the modified array. If a player is not involved in both piece i and j and his coefficient is 1, the closeness between piece i and j on this player will be 1. Closeness array is shown in Table 7. Closeness values are used to indicate pieces that should be assigned to the same day.

After the closeness is calculated, pieces are assigned to rehearsal days using the algorithm in Fig. 2. The first assigned piece is arbitrarily selected from the ones having the highest sum closeness values and assigned to the first rehearsal day. After assigning that piece, the total time used is compared to the available slots. If the total time used is longer than or equal to available slots, it will consider that the rehearsal day is full and the algorithm will assign the remaining music pieces to other rehearsal days. However, if the duration is shorter than available slots, the algorithm will continue assigning the remaining pieces to this day. Before assigning the next piece, player coefficients and closeness values need to be updated. All players who are involved in any already assigned pieces are no longer free on this rehearsal day, so those player coefficients turn to 0. Then the modified array is updated based on the new player coefficients and the closeness values of the remaining pieces are recalculated. The next assigned piece is arbitrarily selected from the ones having the highest closeness sum to all pieces already assigned to the current rehearsal day.

Once the first day is full, the algorithm moves to the next rehearsal day. All already assigned pieces are removed from the list. The whole process (starting from calculating player coefficients, updating the modified array, determining closeness values among

Table 6
Modified array.

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

Table 7
A closeness array.

	Piece												Sum
	1	2	3	4	5	6	7	8	9	10	11	12	
Piece													
1	0	1	1	2	1	1	2	3	0	2	3	2	18
2	1	0	0	2	1	1	1	2	0	2	2	0	12
3	1	0	0	0	0	0	0	1	0	1	1	1	5
4	2	2	0	0	1	1	2	3	0	2	3	1	17
5	1	1	0	1	0	1	1	1	0	1	1	0	8
6	1	1	0	1	1	0	1	1	0	1	1	0	8
7	2	1	0	2	1	1	0	2	0	1	2	1	13
8	3	2	1	3	1	1	2	0	0	3	4	2	22
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	2	2	1	2	1	1	1	3	0	0	3	1	17
11	3	2	1	3	1	1	2	4	0	3	0	2	22
12	2	0	1	1	0	0	1	2	0	1	2	0	10

Table 5
Bit-flipped array.

Piece	1	2	3	4	5	6	7	8	9	10	11	12	Player coefficient
Player 1	0	0	0	1	0	0	0	1	0	0	1	1	0
Player 2	1	0	0	0	0	1	1	0	0	1	1	0	0
Player 3	0	1	0	1	0	0	0	1	0	1	1	0	1
Player 4	1	0	1	0	0	0	0	1	0	1	1	1	1
Player 5	1	0	0	1	0	0	1	1	0	0	1	1	1
Player 6	1	0	0	1	0	0	1	1	1	0	0	0	0
Player 7	0	0	1	1	1	0	0	0	1	0	0	1	0
Player 8	0	0	0	0	1	0	0	1	0	1	0	1	0
Player 9	1	1	0	1	1	1	1	1	0	1	1	0	1
Player 10	0	0	0	0	0	1	1	1	0	0	0	1	0
Duration	4	4	2	4	2	1	4	3	2	4	3	3	

Inputs
$\mathcal{F}$: A set of feasible solutions
$\mathcal{S}$: A set of all music pieces
$\mathcal{S}_a$: A set of music pieces already scheduled
$\mathcal{S}_d$: A set of music pieces assigned to day d
$\mathcal{S}_n$: A set of music pieces not assigned to rehearsal day
$\boldsymbol{M}$: A set of modified element of all player k and piece j
$\boldsymbol{C}$: Closeness matrix
$\boldsymbol{P}$: Player matrix with bit flip
t_d: Total time used in day d
T_d: Available time in day d
N: Number of feasible solutions required

repeat
  Assign a randomly selected piece to a random day if not the first solution.
  for each day **do**
    while t_d ¡ T_d and $\mathcal{S}_n \neq \phi$ **do**
      if $\mathcal{S}_d \neq \phi$ **then**
        for each piece $i \in \mathcal{S}_d$ **do**
          Set coefficient of player k to 0 if the player plays piece i in day d
      $\boldsymbol{M} \leftarrow$ Player coefficients $\times \boldsymbol{P}$
      $\boldsymbol{C} \leftarrow$ ClosenessCalculation($\boldsymbol{M}$)
      if $\mathcal{S}_d \neq \phi$ **then**
        Assign a piece p in $\mathcal{S}_n$ with largest row sum in $\boldsymbol{C}$ to $\mathcal{S}_d$, with only columns $k \in \mathcal{S}_d$ selected
      else
        Assign a piece p in $\mathcal{S}_n$ with largest row sum in $\boldsymbol{C}$ to $\mathcal{S}_d$
      $\mathcal{S}_n \leftarrow \mathcal{S}_n - p$
    $\mathcal{S}_a \leftarrow \mathcal{S}_a \cup \{\mathcal{S}_d\}$
  $\mathcal{S}_a \leftarrow$ EliminateExcessive($\mathcal{S}_a$)
  $\mathcal{S}_a \leftarrow$ LocalSearch($\mathcal{S}_a$)
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathcal{S}_a\}$
until $|\mathcal{F}|$ is equal to N
Return $\mathcal{F}$

Fig. 2. The algorithm to assign pieces to rehearsal days.

procedure ELIMINATEEXCESSIVE($\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_D\}$)
  $\mathcal{D}_x \subset \{1, 2, \ldots, D\}$: Set of excess days ($t_d \geq T_d$)

  for each day $d \in \mathcal{D}_x$ **do**
    for each day $d' \in \mathcal{D}_x^c$ **do**
      for each piece $i \in \mathcal{S}_d$ **do**
        Set coefficient of player k to 1 if the player only plays piece i in day d
        $\boldsymbol{M} \leftarrow$ Player coefficients $\times \boldsymbol{P}$
        $\boldsymbol{C} \leftarrow$ ClosenessCalculation($\boldsymbol{M}$)
        for each piece $j \in \mathcal{S}_{d'}$ **do**
          Compute closeness sum matrix for leaving piece i and entering piece j
      Find highest value entry (p_l, p_e) in closeness sum matrix with length(p_e) < length(p_l)
      if such pair (p_l, p_e) exists **then**
        Swap piece p_l in day d and piece p_e in day d'
        if day d' becomes an excess day **then**
          Append day d' to $\mathcal{D}_x$
        if day d becomes a feasible day **then**
          Remove day d from $\mathcal{D}_x$
          Break
      else
        Break

Fig. 3. The algorithm to fix infeasible solutions.

the remaining pieces, and assigning pieces) is repeated until every piece is scheduled.

Note that since the algorithm assigns a music piece before checking available slots, an infeasible solution can be occurred. The algorithm allows an infeasible schedule in order to explore a larger search space, and this infeasibility is repaired later. Without allowing the search to enter the infeasible region, the algorithm would find the same solutions repeatedly and could not get to optimality.

When an infeasible solution occurs from the previous assignment, the algorithm in Fig. 3 must be performed to fix the solution. The methodology is done by swapping a piece in an infeasible day, called a leaving piece, to a piece in a feasible day whose total time used is shorter than the available slots, called an entering piece. The closeness values are also used to select a pair of pieces to be swapped. For example, as the solution shown in Table 8, pieces 2, 4, 7, 10 and 11 are assigned to day 1 and pieces 1, 3, 5, 6, 8, 9 and 12 are assigned to day 2. In this case, 18 slots are available in each

Table 8
A solution with an excessive rehearsal day.

Day 1							
Piece	2	4	7	10	11		
Duration	4	4	4	4	3		
Day 2							
Piece	1	3	5	6	8	9	12
Duration	4	2	2	1	3	2	3

Table 9
The closeness values between entering and leaving music pieces.

Leaving piece	Entering pieces							Duration
	1	3	5	6	8	9	12	
2	9	2	4	4	12	0	5	4
4	8	2	4	4	11	0	4	4
7	8	2	4	4	12	0	4	4
10	8	1	4	4	11	0	4	4
11	7	1	4	4	10	0	3	3
Duration	4	2	2	1	3	2	3	

day. The total time used in day 1 is 19 slots, which is longer than the available slots and the total time used in day 2 is 17, which is shorter than the available slots. If piece 2 is a leaving piece and piece 3 is an entering piece, the closeness of this pair is the sum of closeness between piece 3 and all pieces in day 1 except piece 2. A closeness array of all possible entering and leaving pieces are shown in Table 9. The selected pair is the one with the highest closeness where the duration of entering piece is shorter than the one of leaving piece. If two or more pairs have the same highest closeness, the pair with leaving and entering pieces which are assigned to the rehearsal day later will be selected. For example, from Table 9, two pairs (2–8 and 7–8), have the highest closeness. In this case, the algorithm selects piece 7 as a leaving piece because piece 7 is assigned to this rehearsal day after piece 2. The process is continued until a feasible solution is obtained.

Once we obtain a feasible solution, a local search is performed. The local search tries to improve the assignment by swapping of a pair of pieces between days without violating the feasibility. The local search algorithm is shown in Fig. 4. The current solution is replaced whenever the better solution is found. This local search continues until there is no improvement after performing all pairwise interchange.

Since the methodology needs alternative optimal solutions from stage 1 to avoid suboptimal solution in stage 2, the heuristics search for other optimal solutions by randomly pre-assigning some music pieces to a rehearsal day. Then, the algorithm calculates player coefficients, updating the modified array, and determining closeness values of the remaining pieces. For the rehearsal day having pre-assigned pieces, the next selected piece is the one that has the highest closeness value to those already assigned pieces. In case of no pre-assignment in a day, the first selected piece is the one with the highest closeness value. From the experiment, we found that the number of pre-assigned pieces affects the search results. If the number of pre-assigned pieces is too low, the algorithm tends to reach the same optimal solutions easily. On the other hand, if the number of pre-assigned pieces is too high, the algorithm will randomly search and hardly reach an optimal solution. We also found that the number of pre-assigned pieces should be about 20% of the total number of pieces.

3.2. A mixed integer programming model for daily sequencing music pieces

After assigning music pieces to rehearsal days, all alternative optimal solutions are solved using an integer programming model to minimize the total waiting time during each rehearsal day. The integer programming model in this section is similar to the one in Section 2.2. However, this model sequences music pieces within each day separately. Therefore, we can reduce dimension "day" of all decision variables from the previous model. This enables the model to be solved much faster. The integer programming model for daily sequencing music pieces is as follows.

Indices

I	= $\{1, 2, \ldots, n\}$ is a set of time slots in each rehearsal day
J	= $\{1, 2, \ldots, m\}$ is a set of music pieces
K	= $\{1, 2, \ldots, p\}$ is a set of players

Parameters

$play_{kj}$	= 1 if player k plays piece j, 0 otherwise
$duration_j$	= The duration of piece j

Variables

$playslot_{ki}$	= 1 if player k is required to play during slot i, 0 otherwise
$timetable_{ij}$	= 1 if piece j is rehearsed during slot i, 0 otherwise
$timestart_{ij}$	= 1 if piece j begins rehearsing in or before slot i, 0 otherwise
$timeend_{ij}$	= 1 if piece j finishes rehearsing in or after slot i, 0 otherwise
r_{ki}	= 1 if player k is at the rehearsal during slot i, 0 otherwise
a_{ki}	= 1 if player k arrives in or before slot i, 0 otherwise
l_{ki}	= 1 if player k leaves in or after slot i, 0 otherwise
w_{ki}	= 1 if player k is at the rehearsal during slot i but does not play, 0 otherwise
z	= The total waiting time.

Objective function (20) is to find the minimum time that players are in the rehearsal but not play.

$$\text{Minimize} \quad z = \sum_{k=1}^{p}\sum_{i=1}^{n} w_{ki} \tag{20}$$

Constraints (21) and (22) force to schedule only one piece at a time and also reserve rehearsal slots to complete that piece.

$$\sum_{j=1}^{m} timetable_{ij} \leqslant 1 \quad \forall\, i \in I \tag{21}$$

$$\sum_{i=1}^{n} timetable_{ij} = duration_j \quad \forall\, j \in J \tag{22}$$

Constraints (23) assign players to slots they are required to play.

$$playslot_{ki} = \sum_{j=1}^{m} timetable_{ij} \times play_{kj} \quad \forall\, k \in K,\ \forall\, i \in I \tag{23}$$

Constraints (24)–(28) force to rehearse each music piece without dividing it into parts. These are similar to Constraints (6)–(10). Constraints (24)–(26) are used to assign values to the decision variables $timestart_{ij}$, $timeend_{ij}$, $timetable_{ij}$. Constraints (27) and (28) force each piece to start rehearsing at the first slot of the assigned timetable and finish rehearsing at the last slot of the assigned timetable.

$$timetable_{ij} \geqslant timestart_{ij} + timeend_{ij} - 1 \quad \forall\, i \in I,\ \forall\, j \in J \tag{24}$$

```
procedure LocalSearch(S = {S_1, S_2, ..., S_D})
    S_d: A set of music pieces assigned to day d
    t_d: Total duration of all pieces assigned to day d
    T_d: Available time in day d

    repeat
        for each pair of pieces (p_j, p_k), p_j and p_k from two different days d_1 and d_2 do
            if t_{d_1} − length(p_j) + length(p_k) ≤ T_{d_1} and t_{d_2} − length(p_k) + length(p_j) ≤ T_{d_2} then
                Swap piece p_j in day d_1 and piece p_k in day d_2
                if less number of total days is obtained after swapping then
                    Update S_{d_1} and S_{d_2}
                    Break
    until No improvement or no pair to swap
```

Fig. 4. Local search algorithm.

$$timestart_{ij} \leqslant timestart_{(i+1)j} \quad \forall\, i \in \{1,\ldots,n-1\},\ \forall\, j \in J \tag{25}$$

$$timeend_{ij} \geqslant timeend_{(i+1)j} \quad \forall\, i \in \{1,\ldots,n-1\},\ \forall\, j \in J \tag{26}$$

$$timestart_{ij} \geqslant timetable_{ij} \quad \forall\, i \in I,\ \forall\, j \in J \tag{27}$$

$$timeend_{ij} \geqslant timetable_{ij} \quad \forall\, i \in I,\ \forall\, j \in J \tag{28}$$

Constraints (29)–(31) determine the presence of each player in each rehearsal slot.

$$r_{ki} \geqslant a_{ki} + l_{ki} - 1 \quad \forall\, k \in K,\ \forall\, i \in I \tag{29}$$

$$a_{ki} \leqslant a_{k(i+1)} \quad \forall\, k \in K,\ \forall\, i \in \{1,\ldots,n-1\} \tag{30}$$

$$l_{ki} \geqslant l_{k(i+1)} \quad \forall\, k \in K,\ \forall\, i \in \{1,\ldots,n-1\} \tag{31}$$

Constraints (32) and (33) force that each player must arrive at the beginning of the first slot that s/he plays and leave at the end of the last slot that s/he plays.

$$a_{ki} \geqslant playslot_{ki} \quad \forall\, k \in K,\ \forall\, i \in I \tag{32}$$

$$l_{ki} \geqslant playslot_{ki} \quad \forall\, k \in K,\ \forall\, i \in I \tag{33}$$

Constraints (34) determine if play k has to wait in slot i or not.

$$r_{ki} - w_{ki} \leqslant playslot_{ki} \quad \forall\, k \in K,\ \forall\, i \in I \tag{34}$$

4. Experimental results

4.1. Comparison of the proposed methodology and the MIP models

Recall that the proposed methodology divides the problem into 2 stages, and solves them independently. A heuristics is used as a solution methodology in the first stage and a MIP model is used in the second stage. To evaluate the proposed methodology, the heuristics solutions in stage 1 are compared to the optimal solutions from Section 2.1. Since we solve the problem as 2 stages separately, the solutions in stage 2 depend on the assignment from stage 1. Even though a MIP model is used, it does not guarantee the global optimum. Therefore, the second stage solutions are compared to the optimal solution from Section 2.2. Experiments were performed on 70 instances having 5 players and 7 instances having 10 players. Each of them has a different number of music pieces and rehearsal days.[1] The heuristics was implemented in MATLAB and the MIP models were solved by using CPLEX. All experiments ran on a computer with 2.20 GHz Core2Duo processor and 4 GB of RAM.

For the problems with 5 players, we applied the proposed methodology to 7 problem sizes determined by the number of music pieces and rehearsal days. Ten instances are randomly generated for each size. In the first stage problem, the proposed heuristics found the optimal solutions in all instances. On average, the heuristics can obtain the optimal solution within 17.02 s, where the computational time of the MIP is between 5.87 and 418.75 s depending on the number of pieces and days. Since we want to know all alternative solutions in stage 1 to get the global optima in the second stage, heuristics spends less than 36.31 s on average searching for those solutions. Table 10 shows the average computational time of each problem size, and the number of alternative solutions that can be obtained from the heuristics.

In the second stage, the MIP model in Section 2.2 found 66 optimal solutions out of 70 problems. It cannot obtain the optimal solution of 4 instances within 12 h so the best-found solutions are used to evaluate the proposed methodology. We found that the proposed methodology can provide the solutions as good as those obtained by the MIP model in all instances. Although the MIP in Section 3.2 has to deal with a number of solutions from the heuristic algorithm, as music pieces in each rehearsal day are sequenced separately, the model solves a few number of music pieces at a time. Consequently, the total computational time that the proposed methodology takes to sequence music pieces is less than the time taken by solving all music pieces together. From Table 10, the computational time of the proposed methodology is between 24.34 and 507.13 s, which is 85.57% less than the computational time of the MIP in Section 2.2.

We also explore larger problems with 10 players. The proposed methodology is applied to 7 problem sizes determined by the number of music pieces and rehearsal days. Due to the computational time, one instance is randomly generated for each size. In the first stage problem, the proposed heuristics found the optimal solutions in all instances as shown in Table 11. The heuristics can obtain the optimal solution within 466.38 s, where the computational time of the MIP is between 31.97 and 1140.89 s depending on the number of pieces and days. To get alternative solutions in stage 1, the heuristics spends less than 663.81 s searching for those solutions. Table 12 shows the computational time of each problem size, and the number of alternative solutions that can be obtained from the heuristics.

In the second stage, the MIP model in Section 2.2 found 2 optimal solutions out of 7 problems. The optimal solutions of 3 problems cannot be found within 12 h; therefore, the best-found solutions are used. For other 2 problems, feasible solution cannot be found within 12 h; therefore, we increase their computational time to 24 h. Feasible solutions can be obtained for one problem, and cannot be obtained for the others.

We found that the proposed methodology can provide the solutions as good as those obtained by the MIP model in 4 instances. It can provide better solutions in 2 instances. Table 11 shows the objective function values of each problem size. From Table 12,

[1] All test inputs and their best found solutions are provided at http://pioneer.netserv.chula.ac.th/~twipawee/testproblems.pdf.

Table 10
Comparisons of computational times for problems with 5 players.

Problem size			Average MIP elapsed time (s)			Average heuristics elapsed time (s)					Difference (%)
No. of pieces	No. of players	No. of days	Stage 1	Stage 2	Total	Stage 1 (single)	Stage 1 (multiple)	No. of solutions (min,max)	Stage 2	Total	
10	5	2	5.87	89.75	95.62	3.33	4.47	(1,7)	24.34	28.81	69.87
12	5	2	10.55	1239.04	1249.59	5.62	7.35	(1,82)	134.03	141.38	88.69
14	5	2	25.15	2776.90	2802.05	8.44	15.05	(1,109)	410.11	425.16	84.83
10	5	3	18.66	265.39	284.05	4.39	5.56	(1,73)	109.31	114.87	59.56
12	5	3	65.10	2118.49	2183.59	6.45	8.45	(1,17)	44.55	53.00	97.57
14	5	3	82.34	13277.33[a]	13359.67	11.87	26.01	(1,53)	194.67	220.68	98.35
16	5	3	418.75	14163.35[b]	14582.10	17.02	36.31	(1,297)	507.13	543.44	96.27

[a] Terminate CPLEX at 12 h for instances 6 and 8.
[b] Terminate CPLEX at 12 h for instances 5 and 6.

Table 11
Comparisons of objective function values for problems with 10 players.

Problem size			Show-up day (days)			Waiting time (slots)		
No. of pieces	No. of players	No. of days	MIP	Heuristics	Difference (%)	MIP	Heuristics	Difference (%)
10	10	2	19	19	0.00	7	7	0.00
12	10	2	18	18	0.00	27	27	0.00
14	10	2	19	19	0.00	38[a]	23	39.47
10	10	3	25	25	0.00	3[a]	3	0.00
12	10	3	27	27	0.00	5[a]	5	0.00
14	10	3	25	25	0.00	18[b]	17	5.56
16	10	3	25	25	0.00	–[c]	19	–

[a] Best-found solutions at 12 h.
[b] Best-found solution at 24 h.
[c] No feasible solution at 24 h.

Table 12
Comparisons of computational times for problems with 10 players.

Problem size			MIP elapsed time (s)			Heuristics elapsed time (s)					Difference (%)
No. of pieces	No. of players	No. of days	Stage 1	Stage 2	Total	Stage 1 (single)	Stage (multiple)	No. of solutions	Stage 2	Total	
10	10	2	31.97	4287.66	4319.63	26.16	77.64	7	41.92	119.56	97.23
12	10	2	21.19	16058.59	16079.78	93.11	150.01	1	21.05	171.06	98.94
14	10	2	76.44	43203.48[a]	43279.92	219.69	273.29	42	1734.77	2008.06	95.36
10	10	3	215.36	43203.25[a]	43418.61	35.31	108.35	3	12.84	121.19	99.72
12	10	3	865.30	43203.36[a]	44068.66	139.83	218.58	29	152.54	371.12	99.16
14	10	3	544.64	86403.95[b]	86948.59	282.14	392.28	1	5.25	397.53	99.54
16	10	3	1140.89	86403.95[b]	87544.84	466.38	663.81	1	7.94	671.75	99.23

[a] Terminate CPLEX at 12 h.
[b] Terminate CPLEX at 24 h.

the computational time of the proposed methodology is between 5.25 and 1734.77 s, which is 99.39% less than the computational time of the MIP in Section 2.2.

The computational times of the MIP models in Section 2 are very high compared to the ones of the proposed methodology. Hence, as the proposed methodology can find the optimal solutions with a small amount of time, it is reasonable to use this method to solve large problems in a reasonable amount of time.

We did experiment a large problem with 40 music pieces, 20 players and 5 rehearsal days.[2] The proposed methodology found the solution close to the best found of the MIP models. While the MIP model required 35 days of computational time, the methodology required only 6 h with the 13.98% difference in the objective function value. Note that the first feasible solution from the MIP modes was found after 11 days.

4.2. Benefits of multiple solutions

One of the important contributions of the proposed methodology is that it can provide many alternative solutions in stage 1, which lead to the global optimal solution in stage 2. Fig. 5 shows the second stage objective function values using a single solution from the MIP model in Section 2.1 against multiple solutions from heuristics.

[2] A test input and its best found solution are provided at http://pioneer.netserv.chula.ac.th/~twipawee/testproblems.pdf.

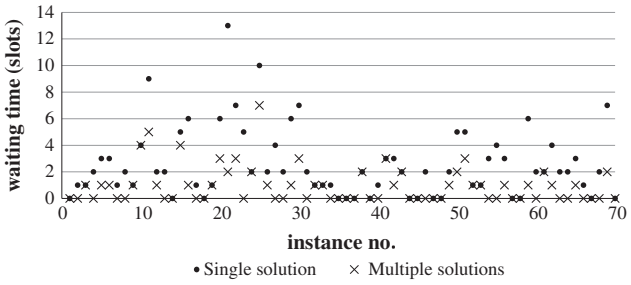

Fig. 5. Objective function values between single solution and multiple solutions.

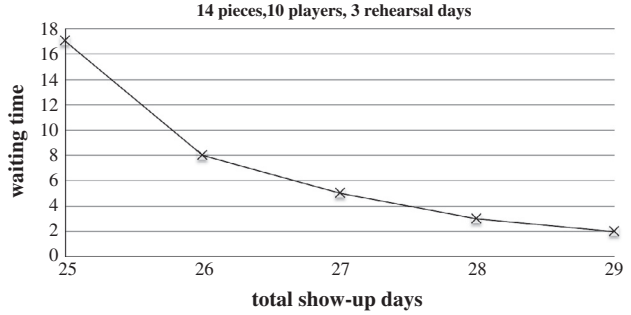

Fig. 6. Trade-off curve.

Table 13
Non-dominated solutions.

Solution no.	Show-up day	Waiting time	Solutions[a]
1	25	17	(8 2 13 1 14), (6 12 10 5 4), (3 11 7 9)
2	26	8	(8 2 14 9), (7 3 11 10), (4 5 12 6 1 13)
3	27	5	(13 2 8 12 1 14), (10 11 3 7), (4 5 6 9)
4	28	3	(7 3 11 10), (9 6 1 12 14), (4 5 8 2 13)
5	29	2	(7 3 11 10), (4 5 9 14), (13 1 12 6 2 8)

[a] Best found solutions due to running out of memory. They can be described as (pieces in day 1), (pieces in day 2), (pieces in day3).

From the 70 instances, using multiple solutions provide better objective function values in 43 instances and the average difference is up to 60%. Even though the computational time of using multiple solutions is higher than the one of using a single solution. It does not significantly increase with the problem sizes.

4.3. Trade-off of the two objectives

Since the primary objective of this paper is to minimize the number of show-up days and the secondary objective is to minimize the total waiting time. We propose a methodology to solve problems into 2 stages. With this methodology, the minimum number of show-up days is obtained while the total waiting time may not be minimized. However, it is interesting to see how these 2 objectives interact with each other.

To deal with multi-objective problems, weighing factors may be given to each objective and the sum of all objectives is considered. However, it is difficult to define the appropriate factor weights and these weights may vary from different points of view. To avoid this problem, the Pareto optimality is widely applied. From the search space, there is a set of "non-dominated" solutions, which is called the "Pareto optimal set". A feasible solution s_a dominates another feasible solution s_b if there is one or more objective(s) that s_a is better than s_b and s_a is as good as s_b in the remaining objectives. Therefore, the non-dominated solutions in the Pareto optimal set make a trade-off among themselves (Ehrgott (1999)).

Many multi-objective papers in cell-formation problems showed their Pareto optimal solutions. Dimopoulos (2006) proposed an algorithm based on a combination of genetic programming and genetic algorithm to solve cell-formation problem with two objectives: minimizing within-cell-load variation and total inter-cell move of parts. Dimopoulos (2007) considered minimizing intra-cell move of parts in addition to the previous two objectives and solved it using the same algorithm. In both papers, the sets of Pareto solutions are displayed in trade-off curves. Bajestani, Rabbani, Rahimi-Vahed, and Khoshkhou (2009) considered a multi-objective dynamic cell formation problem that the product demand changed over time period. A multi-objective scatter search was used to find locally Pareto-optimal frontier. Neto and Filho (2010) developed a simulation-based evolutionary approach to solve the cell formation problem with three objectives: minimizing level of work in process, intra-cell move of parts, and total investment on machines. Pareto frontier and non-dominated solutions were demonstrated.

To see a trade-off between 2 objectives that we considered, we used the proposed algorithm to find Pareto optimal solutions of an instance having 10 players, 14 pieces and 3 rehearsal days. Since the minimum number of show-up days, which is 25, is obtained and the maximum is 30 due to 10 players and 3 rehearsal days. We increased the right-hand side of constraint (19), which is used to force the number of show-up days not to be more than its right hand side value, from 25 to 30 to get better solutions in terms of waiting time. The Pareto frontier is shown in Fig. 6 and the non-dominated solutions are in Table 13. Note that the best solution from setting right-hand-side of Constraint (19) to 30 is 29 show-up days with 2-slot waiting time. If the show-up day is forced to be 30, the waiting time will be increased to 9 slots. Therefore, it does not belong to a non-dominated set.

5. Conclusion and future work

Since the proposed algorithm can find the optimal solution with a small amount of time, we believe that it can help music bands arranging their schedules that minimize the cost and also increase players' satisfaction in terms of waiting time. The algorithm can further be developed by considering other constraints such as music piece precedence constraint. For example, piece 1 should be rehearsed before piece 2 or pieces 4 and 5 should not be rehearsed on the same day. In addition, this paper assumes all players have equal costs. It is unclear how non-uniform player costs would affect the solution quality as well as the computational time. This issue will be explored in future work.

There are some limitations of the proposed heuristics algorithm. Since the algorithm allows solutions to be infeasible in the first place to avoid local optima, it would be much more difficult to repair the feasible solutions when the number of rehearsal days is large. Furthermore, when the number of rehearsal days increases, the algorithm may spend much more time searching for the same optimal solution. Therefore, the logic to avoid this event should be included into the algorithm to reduce the search time.

References

Bajestani, M. A., Rabbani, M., Rahimi-Vahed, A. R., & Khoshkhou, G. B. (2009). A multi-objective scatter search for a dynamic cell formation problem. *Computers and Operations Research, 36*, 777–794.
Baker, K. (1974). *Introduction to sequencing and scheduling*. New York: Wiley.

Boe, W. J., & Cheng, C. H. (1991). A close neighbour algorithm for designing cellular manufacturing systems. *Production Research, 29*, 2097–2116.

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. C. E., Diamond, J. E., & Lin, B. M. T. (1993). Optimal scheduling in film product to minimize talent hold cost. *Optimization Theory and Application*, 3.

Chung, S.-H., Wu, T.-H., & Chang, C.-C. (2011). An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. *Computers and Industrial Engineering, 60*, 7–15.

Dimopoulos, C. (2006). Multi-objective optimization of manufacturing cell design. *International Journal of Production Research, 44*(22), 4855–4875.

Dimopoulos, C. (2007). Explicit consideration of multiple objectives in cellular manufacturing. *Engineering Optimization, 39*(5), 551–565.

Ehrgott, M. (1999). *Multicriteria optimization* (2nd ed.). Berlin: Springer.

Garcia de la Banda, M., Stuckey, P. J., & Chu, G. (2011). Solving talent scheduling with dynamic programming. *Informs Journal on Computing, 23*, 120–137.

Gregory, P., Miller, A., & Prosser, P. (2004). Solving the rehearsal problem with planning and with model checking. *European Conference on Artificial Intelligence, 16*, 157–171.

Kochetov, Y. (2011). Iterative local search methods for the talent scheduling problem. In *Proceedings of 1st international symposium and 10th Balkan conference on operational research, September 22, Thessaloniki, Greece* (pp. 282–288).

Mak, K. L., Wong, Y. S., & Wang, X. X. (2000). An adaptive genetic algorithm for manufacturing cell formation. *Advanced Manufacturing Technology, 16*, 491–497.

McCormick, W. T., Schweitzer, P. J., & White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research, 20*, 993–1009.

Moon, C., & Gen, M. (1999). A genetic algorithm-based approach for design of independent manufacturing cells. *International Journal of Production Economics, 60–61*, 421–426.

Neto, A. R. P., & Filho, E. V. G. (2010). A simulation-based evolutionary multiobjective approach to manufacturing cell formation. *Computers and Industrial Engineering, 59*, 64–74.

Nordström, A. L., & Tufekci, S. (1994). A genetic algorithm for the talent scheduling problem. *Computers and Operations Research, 21*, 927–940.

Paydar, M. M., & Saidi-Mehrabad, M. (2013). A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. *Computers and Operations Research, 40*, 980–990.

Sakulsom, N., & Tharmmaphornphilas, W. (2011). A multi-objective music rehearsal scheduling problem. In *12th Asia Pacific industrial engineering and management systems conference* (pp. 25–29).

Smith, B. (2003). Constraint programming in practice: Scheduling a rehearsal. Research, report APES-67.

Tariq, A., Hussain, I., & Ghafoor, A. (2009). A hybrid genetic algorithm for machine-part grouping. *Computers and Industrial Engineering, 56*, 347–356.

Wu, X., Chu, C. X., Wang, Y., & Yan, W. A. (2007). A genetic algorithm for cellular manufacturing design and layout. *European Journal of Operational Research, 181*, 156–167.