Computers & Industrial Engineering 126 (2018) 673–680

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A generalized variable neighborhood search algorithm for the talent scheduling problem

Mohammad Ranjbar[a,*], Ahmad Kazemi[b]

[a] Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] School of Mathematical Sciences, Monash University, Melbourne, Australia

ARTICLE INFO

Keywords:
Talent scheduling
Metaheuristic
Generalized variable neighborhood search algorithm

ABSTRACT

An important part of the movie production process is to determine a shooting sequence of scenes to minimize the total cost of the actors, called the talent scheduling problem. We first modify a previously developed formulation of the problem as an integer linear programming model and develop a metaheuristic based on the generalized variable neighborhood search algorithm to solve it. The computational experiments based upon available test instances in the literature suggest that our algorithm outperforms previously developed algorithms especially for large-scale instances.

1. Introduction

A movie consists of several scenes where each of them includes a set of actors (talents). After finishing shooting process, the scenes are assembled to be presented, thus scenes are not shot in the sequence they are appeared. Determining the optimal sequence for shooting scenes in order to minimize the total costs of actors motivates the talent scheduling problem, which is described as follows. There are a set of scenes and a set of actors where a given subset of actors are involved in each scene. We assume all scenes are shot in a predetermined location. Also, each scene has a positive duration and each actor has a daily wage, paid for each day of a required interval. The required interval consists of the first day of the first scene an actor is needed to the last day of the last scene the actor is needed while he/she may not work in some days of this interval but his/her daily wage should be paid. The challenging optimization problem is that to determine the optimal permutation of scenes such that the total salary costs of actors is minimized.

A simplified version of the talent scheduling problem, called concert scheduling problem, was introduced by Adelson, Norman, and Laporte (1976) in which the costs of all actors are identical. Cheng, Diamond, and Lin (1993) considered the scene scheduling problem where each scene last a shooting day. Also, Cheng et al. (1993) proved that scene scheduling problem is NP-hard even if each actor is required in only two scenes, and all wages and durations are identical. Nordström and Tufekci (1994) developed several hybrid genetic algorithms for the scene scheduling problem. For the same problem, Fink and Voß (1999) designed and implemented a simulated annealing and several tabu search algorithms.

Smith (2003) developed a constraint programming approach to solve both problems introduced by Adelson et al. (1976) and Cheng et al. (1993). The talent scheduling problem in which scenes may have different durations and also actors may have non-identical daily wages was introduced by De la Banda, Stuckey, and Chu (2011). They developed a dynamic programming approach accelerated with preprocessing and restricting search procedures. Also, Qin, Zhang, Lim, and Liang (2016) developed an efficient branch-and-bound algorithm for the problem introduced by De la Banda et al. (2011), outperforming already developed algorithms. Bomsdorf and Derigs (2008) considered a generalized version of the problem, introduced by De la Banda et al. (2011), dealing with some of the practical constraints such as the precedence relations among the scenes, the resource limitations, and permitted time windows for the scenes and the actors.

Contribution of this paper is twofold: (1) we modify and improve a previously developed formulation of the talent scheduling problem; (2) we develop a generalized variable neighborhood search (GVNS) algorithm to solve the problem which has better performance than already developed algorithms in the literature especially for large-size instances.

The remainder of this paper is organized as follows. In Section 2, problem statement and formulation is presented. The GVNS algorithm is developed in Section 3. Section 4 is assigned to the computational experiments. Finally, conclusions and future research directions are presented in Section 5.

* Corresponding author.
E-mail addresses: m_ranjbar@um.ac.ir (M. Ranjbar), ahmad.kazemi@monash.edu (A. Kazemi).

https://doi.org/10.1016/j.cie.2018.10.028
Received 8 September 2017; Received in revised form 27 September 2018; Accepted 10 October 2018
Available online 12 October 2018
0360-8352/

2. Problem statement and formulation

The talent scheduling problem can be described mathematically as follows. Consider n scenes as the set $S = \{s_1, \cdots, s_n\}$ and m actors as the set $A = \{a_1, \cdots, a_m\}$. We define matrix $R_{m\times n}$ where $r_{ij} = 1$ if actor a_i should participate in scene s_j; otherwise r_{ij}=0. For each scene $s_j \in S$ with positive duration $d(s_j)$, a subset$a(s_j)$ of actors are required where$a(s_j) = \{a_i \in A | r_{ij} = 1\}$. Also, each actor a_i plays in a subset $s(a_i)$ of scenes where $s(a_i) = \{s_j \in S | r_{ij} = 1\}$. We show by $\boldsymbol{\sigma}$ the set of all $n!$ possible sequences of the n scenes where each sequence $\sigma = (\sigma_1, \cdots, \sigma_n) \in \boldsymbol{\sigma}$ indicates a solution. The daily wage for each actor $a_i \in A$, shown by $w(a_i)$, is paid based on the solution σ and from his/her starting day $sta_i(\sigma)$ to his/her finishing day $fta_i(\sigma)$. The goal of the talent scheduling problem is to find a shooting sequence $\sigma^* \in \boldsymbol{\sigma}$ that minimizes the total paid salaries.

In addition to aforementioned notations, we define the four following variables.

x_{kj}: a binary variable that gets 1 if scene s_j is shot immediately after scene s_k, and 0 otherwise.

sts_j: the starting day for shooting scene s_j.

sta_i: the starting day for working actor a_i.

fta_i: the finishing day for working actor a_i.

We define two dummy scenes 0 and $n + 1$ with $d(s_0) = d(s_{n+1}) = 0$ to indicate the first and the last scenes to be shot. In other words, we assume $sts_0 = 0$ and $sts_{n+1} = 1 + \sum_{j=1}^{n} d(s_j)$.

The integer linear programming model reads as follows.

$$Min \sum_{i=1}^{m} w(a_i)\left(fta_i - sta_i - \sum_{j=1|r_{ij}=1}^{n} d(s_j)\right) \tag{1}$$

s.t.

$$\sum_{j=1}^{n} x_{0j} = 1 \tag{2}$$

$$\sum_{k=1}^{n} x_{k,n+1} = 1 \tag{3}$$

$$\sum_{j=1,j\neq k}^{n+1} x_{kj} = 1; \quad k = 1, \cdots, n \tag{4}$$

$$\sum_{k=0,k\neq j}^{n} x_{kj} = 1; \quad j = 1, \cdots, n \tag{5}$$

$$sts_j + M(1 - x_{kj}) \geq sts_k + d(s_k); \quad k = 0, \cdots, n, \quad j = 1, \cdots, n+1 \tag{6}$$

$$sts_j - M(1 - x_{kj}) \leq sts_k + d(s_k); \quad k = 0, \cdots, n, \quad j = 1, \cdots, n+1 \tag{7}$$

$$sts_j \geq sta_i; \quad i = 1, \cdots, m, \quad j = 1, \cdots, n \ \ \& \ \ r_{ij} = 1 \tag{8}$$

$$sts_j \leq fta_i - d(s_j); \quad i = 1, \cdots, m, \quad j = 1, \cdots, n \ \ \& \ \ r_{ij} = 1 \tag{9}$$

$$sts_0 = 0 \tag{10}$$

$$sts_{n+1} = \sum_{j=1}^{n} d(s_j) + 1 \tag{11}$$

$$x_{kj} \in \{0, 1\}; \quad k = 0, \cdots, n, \quad j = 1, \cdots, n+1 \tag{12}$$

$$sts_j, sta_i, fta_i \in \ ^{+}; \quad i = 1, \cdots, m; \quad j = 0, \cdots, n+1 \tag{13}$$

The total costs is the total salaries paid to the actors for the duration they are employed. Clearly, the salary costs are paid to the actors for the times they are playing a role cannot be reduced by changing shooting sequence. Therefore, the objective function (1) only minimizes the total *holding costs*. Holding cost is the salary that should be paid to the actors for the time that they are not playing a role but are waiting until the next scenes, in which they are involved, are shot.

Constraints (2) and (3) guarantee that s_0 and s_{n+1} are shot as the first and the last scenes, respectively. Constraints (4) and (5) impose exactly one immediate predecessor and one immediate successor to each scene, respectively. Constraints (6) and (7) ensures that if $x_{kj} = 1$, then $sts_j = sts_k + d(s_j)$. Also, these constraints do not allow sub-tour creation in the solution. Constraints (8) and (9) ensure that each scene is shot when required actors are available. Constraints (10) and (11) fix the starting time of shooting scenes s_0 and s_{n+1}, respectively. Finally, Constraints (12) and (13) determine the type and feasible range of variables where $^{+}$ indicates the set of non-negative integers.

It should be noticed that in the recently published paper by Qin et al. (2016), a similar formulation has been developed but they have used a non-linear constraint instead of constraints (6) and (7) as constraint (14). Next, they have transformed this non-linear constraint to a set of linear constraint using four equations.

$$\sum_{j=1,k\neq j}^{n+1} sts_j x_{kj} = sts_k + d(s_k); \quad k = 0, \cdots, n \tag{14}$$

We claim that this equation does not work correctly. For example, consider an instance including two actors and three scenes. The equations related to constraints (10), (11) and (14) are listed as follows.

$$sts_1 X_{01} + sts_2 X_{02} + sts_3 X_{03} = sts_0 + d(s_0) \tag{I}$$

$$sts_2 X_{12} + sts_3 X_{13} = sts_1 + d(s_1) \tag{II}$$

$$sts_1 X_{21} + sts_3 X_{23} = sts_2 + d(s_2) \tag{III}$$

$$sts_1 X_{31} + sts_2 X_{32} = sts_3 + d(s_3) \tag{IV}$$

$$sts_0 = 0 \tag{V}$$

$$sts_4 = d(s_1) + d(s_2) + d(s_3) + 1 \tag{VI}$$

Also, assume duration of all scenes is equal to one day. For the arbitrary feasible sequence $\sigma = (1, 2, 3)$, we have $X_{01} = X_{12} = X_{23} = X_{34} = 1$. If we consider these values for Eqs. (I)-(VI), we obtain the following conflicting values resulting in an infeasible solution.

$sts_1 = sts_0$; $sts_2 = 1$; $sts_3 = sts_2 + 1$; $0 = sts_3 + 1$; $sts_0 = 0$ and $sts_4 = 4$

3. Generalized variable neighborhood search algorithm

In this section, a metaheuristic based on the general variable neighborhood search (GVNS) is developed for the talent scheduling problem. The GVNS performs local searches to reach local optimum in addition to a shaking procedure to avoid getting trap in local optimum. Moreover, GVNS employs variable neighborhood descent (VND) as a local search that systemically explores different neighborhood structures (See Mladenović and Hansen (1997)).

This algorithm begins with an initial solution. Since any permutation of scenes is a feasible solution, the initial solution is generated randomly. This algorithm is based on the GVNS developed by Mjirda, Todosijević, Hanafi, Hansen, and Mladenović (2016) for the well-known travelling salesman problem because it was very successful implementation and also our problem is very similar to the travelling salesman problem.

The sketch of our developed GVNS is presented in Algorithm 1 as pseudo-code. It includes a local search procedure, called VND and described in Section 3.2, and a shaking procedure, described in Section 3.3. It takes two parameters s_{max} (maximum number of moves in a shake) and t_{max} (time limit) as input. A random sequence X is given as the initial solution. Next a while-loop is reaped until termination criteria, consider as a time limit, is met. At each iteration of the GNVS, first a shaking procedure is applied on the current solution and then the VND is applied as a local search.

Input: s_{max} (Maximum number of moves in a shake), t_{max} (time limit)

$X \leftarrow$ generate a random sequence as an initial solution

while computation time < t_{max} **do**

  $s \leftarrow 1$;

  while $s \leq s_{max}$ **do**

    $X' \leftarrow$ Shake (X, s);

    $X'' \leftarrow$ VND (X');

    if $f(X'') < f(X)$ **do**

      $X \leftarrow X''$;

      $s \leftarrow 1$;

    else

      $s \leftarrow s + 1$;

    end

  end

End

Table 1
The example of the talent scheduling problem.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	$w(a_i)$
a_1	X	X			X	X	X						10
a_2	X			X	X	X			X			X	30
a_3				X	X					X	X	X	5
a_4								X	X	X	X		10
a_5			X	X	X								60
a_6				X		X							15
$d(s_j)$	1	3	2	1	5	1	2	1	6	3	2	1	

Table 2
A solution of the example problem before applying 2-opt move.

σ	s_8	s_{10}	s_{11}	s_9	s_{12}	s_7	s_6	s_4	s_5	s_3	s_1	s_2	Holding Costs
a_1						X	X		X		X	X	30
a_2				X	X		X	X	X		X		120
a_3		X	X		X			X	X				45
a_4	X	X	X	X									0
a_5								X	X	X			0
a_6							X	X					0

Table 3
A solution of the example problem after applying 2-opt move.

σ	s_8	s_{10}	s_{11}	s_9	s_{12}	s_1	s_6	s_4	s_5	s_3	s_7	s_2	Holding Costs
a_1						X	X		X		X	X	30
a_2				X	X	X	X	X	X				0
a_3		X	X		X			X	X				40
a_4	X	X	X	X									0
a_5								X	X	X			0
a_6							X	X					0

Table 4
A solution of the example problem before applying OR-opt-3 move.

σ	s_7	s_2	s_3	s_5	s_4	s_6	s_{12}	s_9	s_{10}	s_1	s_{11}	s_8	Holding Costs
a_1	X	X		X		X				X			130
a_2				X	X	X	X	X		X			30
a_3				X	X		X		X		X		40
a_4								X	X		X	X	10
a_5			X	X	X								0
a_6					X	X							0

Table 5
A solution of the example problem after applying OR-opt-3 move.

σ	s_7	s_2	s_3	s_5	s_4	s_6	s_1	s_{12}	s_9	s_{10}	s_{11}	s_8	Holding Costs
a_1	X	X		X		X	X						30
a_2				X	X	X	X	X	X				0
a_3				X	X			X		X	X		40
a_4									X	X	X	X	0
a_5			X	X	X								0
a_6					X	X							0

Table 6
A solution of the example problem before applying COMPRESS move.

σ	s_3	s_7	s_2	s_5	s_6	s_1	s_{12}	s_8	s_9	s_4	s_{11}	s_{10}	Holding Costs
a_1		X	X	X	X	X							0
a_2				X	X	X	X		X	X			30
a_3				X			X			X	X	X	45
a_4								X	X		X	X	10
a_5	X			X						X			900
a_6					X					X			135

3.1. Neighborhood structures

In this section, we discuss the 8 neighborhood structures that are used as local search operators. As the talent scheduling problem is very similar to the traveling salesman problem (TSP), we have used the moves that are most common in the literature of this problem, namely 2-opt and OR-opt. Since OR-opt is a special case of 3-opt, the *k*-opt

Table 7
A solution of the example problem after applying COMPRESS move.

σ	s_7	s_2	s_3	s_5	s_4	s_6	s_1	s_{12}	s_8	s_9	s_{11}	s_{10}	Holding Costs
a_1	X	X		X		X	X						30
a_2				X	X	X	X	X		X			30
a_3				X	X			X			X	X	45
a_4									X	X	X	X	0
a_5			X	X	X								0
a_6					X	X							0

Table 8
The APD for $t_{max} = 1$ s.

N	m							
	8	10	12	14	16	18	20	22
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
22	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26	0.000	0.000	0.000	0.000	0.079	0.077	0.000	0.006
28	0.000	0.000	0.000	0.000	0.312	0.000	0.000	0.019
30	0.000	0.000	0.000	0.000	0.222	1.020	0.097	0.000
32	0.000	0.000	0.054	0.000	0.003	0.945	1.398	0.385
34	0.000	0.189	0.000	0.314	2.352	0.428	4.493	−0.446
36	0.352	2.688	3.824	0.903	1.388	0.012	0.167	2.948
38	0.000	1.906	3.979	2.504	0.002	2.134	0.569	3.541
40	0.182	0.174	2.057	0.307	0.916	0.925	1.901	1.165
42	1.352	2.807	3.225	4.809	1.218	−0.552	1.621	−0.361
44	0.438	1.997	3.341	4.788	2.972	1.065	−0.015	1.321
46	0.548	0.945	3.600	5.360	1.829	5.990	0.428	1.905
48	4.497	4.670	2.417	2.473	0.852	4.160	2.729	2.443
50	4.017	4.575	4.302	3.404	−0.266	4.926	2.780	1.474
52	2.423	3.057	1.318	2.126	0.281	0.386	2.984	2.963
54	3.038	1.690	3.978	1.132	1.144	2.270	1.441	1.797
56	9.482	5.517	9.463	6.286	1.943	7.307	−0.714	0.813
58	11.031	12.143	3.060	3.154	6.909	−0.074	−2.372	−4.224
60	3.379	4.874	5.742	2.098	0.595	4.012	4.051	0.006
62	9.204	7.049	14.747	6.488	3.531	0.875	−1.199	0.113
64	4.907	10.632	3.835	3.408	9.345	−0.183	−0.923	−3.062

moves with $k > 2$ have not been used. In the problem at hand, the 2-opt move is swapping two scenes together which improve the solution. OR-opt-k move extracts k consecutive scene from the current sequence and inserts them in a new position between two scenes (Johnson and McGeoch (1995)). We have considered OR-opt-1, OR-opt-2, and OR-opt-3 moves. In addition, an OR-opt move is called backward OR-opt move if the extracted scene is inserted between the previous scenes and it is called forward OR-opt if it is inserted between the next scenes. Therefore, we have six different OR-opt moves, namely backward OR-opt-1, forward OR-opt-1, backward OR-opt-2, forward OR-opt-2, backward OR-opt-3, and forward OR-opt-3.

Consider an example including 6 actors and 12 scenes where information of parameters a_i and s_j are determined by the matrix $R_{m\times n}$ shown in Table 1, where cell r_{ij} is filled with the symbol "X" if a_i should participate in s_j. Also, the last row and the last column indicate duration of scenes and actors' wage, respectively.

Tables 2 and 3 illustrate a 2-opt move on a solution of the example problem in which position of scenes 1 and 7 are exchanged. This move reduce the holding costs from 195 to 70 cost unit. The obtained solution is one of the optimal solutions of the example problem.

Also, Tables 4 and 5 illustrate how forward OR-opt-3 move works. This move extract three consecutive scenes 12, 9 and 10 and insert them in the same order between scenes 1 and 11. This move reduces holding costs from 210 to 70 cost unit which is an optimal solution of the example.

The aforementioned neighborhood structures are designed for TSP in the literature. However, the cost structure of the talent scheduling problem is fundamentally different from TSP. Therefore, we propose a new neighborhood structure for the talent scheduling problem, called COMPRESS, which exploits the cost structure of the problem. For each actor, COMPRESS tries to reduce their holding cost by simultaneously delaying the shooting of the first presence of the actor and antedating the shooting of the last presence of the actor.

Consider a particular actor, a_i, and the sequence $Q_1 = \{S, s_1, S', s_2, \widehat{S}, s_{k-1}, S'', s_k, \bar{S}\}$ for shooting the scenes. In the sequence Q_1, capital letters represent a sequence of a subset of scenes and small letters represent a single scene. Moreover, assume that the actor a_i plays in the scenes s_1, s_2, s_{k-1}, and s_k, and does not participate in any

Table 9
The APD for $t_{max} = 10$ s.

n	m							
	8	10	12	14	16	18	20	22
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
22	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
28	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.019
30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
32	0.000	0.000	0.000	0.000	0.003	0.000	0.000	−0.394
34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.919
36	0.000	0.000	0.000	0.000	0.000	−1.619	0.000	−0.338
38	0.000	0.000	0.000	0.000	0.000	0.000	−1.050	−0.104
40	0.000	0.000	0.069	0.000	0.208	0.000	−0.081	−0.660
42	0.000	0.000	0.000	0.000	−0.628	−0.764	0.200	−3.751
44	0.000	0.000	0.610	0.000	0.346	−0.795	−1.200	−1.047
46	0.035	0.000	0.187	0.015	−0.983	−1.985	−3.342	−0.182
48	0.000	0.449	−0.600	−0.577	−0.973	−0.841	−1.139	−3.015
50	0.000	0.051	0.000	1.106	−2.010	−1.618	−0.384	−2.898
52	0.000	0.085	−2.603	−0.653	−2.193	−3.098	−1.913	0.113
54	0.006	−0.093	−1.399	−3.582	−3.799	−0.908	−2.344	−2.835
56	0.000	0.000	0.867	−1.454	2.236	0.937	−3.442	−3.231
58	0.017	0.000	−1.013	−2.737	1.564	0.346	−7.403	−3.977
60	0.000	0.804	−1.505	−1.925	−1.983	−1.710	−1.503	−4.398
62	1.011	0.440	−0.936	1.628	−1.547	−3.043	−4.967	−3.322
64	0.945	−0.476	−0.481	0.685	2.139	−4.891	−2.684	−3.070

Table 10
The APD for $t_{max} = 30$ s.

n	m							
	8	10	12	14	16	18	20	22
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
22	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
28	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
32	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.394
34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.919
36	0.000	0.000	0.000	0.000	0.000	−1.619	0.000	−0.338
38	0.000	0.000	0.000	0.000	0.000	0.000	−1.050	−0.174
40	0.000	0.000	0.000	0.000	0.000	0.000	−0.097	−0.660
42	0.000	0.000	0.000	0.000	−0.628	−0.688	0.000	−3.836
44	0.000	0.000	0.000	0.000	0.000	−0.980	−1.541	−1.684
46	0.000	0.000	0.187	0.000	−0.983	−2.402	−3.704	−0.182
48	0.000	0.000	−1.052	−0.602	−2.916	−0.870	−1.316	−3.015
50	0.000	0.000	0.000	0.000	−2.600	−1.771	−0.785	−2.998
52	0.000	0.000	−2.731	−0.646	−2.193	−3.396	−3.609	−0.840
54	0.000	−0.562	−1.569	−3.712	−3.273	−1.976	−3.225	−4.029
56	0.024	0.000	0.000	−2.069	−1.623	0.458	−3.878	−3.103
58	0.000	0.000	−1.133	−2.785	−0.578	−4.219	−8.251	−6.094
60	0.000	0.000	−2.273	−3.435	−2.003	−2.675	−4.132	−4.562
62	0.031	0.000	−1.314	0.000	−2.985	−3.750	−6.201	−4.906
64	0.012	−1.000	−5.473	−0.903	−0.976	−4.623	−4.130	−6.118

Table 11
The APD for $t_{max} = 60$ s.

n	m							
	8	10	12	14	16	18	20	22
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
22	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
28	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
32	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.394
34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.919
36	0.000	0.000	0.000	0.000	0.000	−1.619	0.000	−0.338
38	0.000	0.000	0.000	0.000	0.000	0.000	−1.050	−0.174
40	0.000	0.000	0.000	0.000	0.000	0.000	−0.097	−0.660
42	0.000	0.000	0.000	0.000	−0.628	−0.764	0.000	−3.836
44	0.000	0.000	0.000	0.000	0.000	−0.899	−1.666	−1.684
46	0.000	0.000	0.000	0.000	−0.983	−2.402	−3.984	−0.182
48	0.000	0.000	−0.938	−0.602	−2.916	−0.870	−1.316	−3.015
50	0.000	0.000	0.000	0.000	−2.600	−1.837	−0.785	−3.084
52	0.000	0.000	−2.731	−0.653	−2.193	−3.191	−3.652	−1.082
54	0.000	−0.349	−1.642	−3.712	−3.573	−2.027	−3.225	−4.101
56	0.000	0.000	0.000	−2.170	−1.653	0.064	−3.878	−3.411
58	0.000	0.000	−1.205	−2.828	−0.578	−4.219	−8.461	−6.311
60	0.000	0.000	−2.273	−3.437	−1.983	−2.924	−3.678	−4.616
62	0.277	0.000	−1.796	0.000	−3.928	−3.735	−7.284	−5.283
64	0.000	−1.004	−5.480	−0.496	−1.073	−6.200	−4.934	−7.628

scenes of the sequences S, $S^{'}$, $S^{''}$, and $\bar{S}$. It is not important if the actor plays in the scenes of the sequence $\widehat{S}$ or not. Therefore, based on the sequence Q_1,the actor a_i first presences when the scene s_1 is shot and their last presence is when the scene s_k is shot. The COMPRESS move tries to delay shooting scene s_1 and antedating shooting scene s_k as much as possible to obtain the sequence $Q_2 = \{S, S^{'}, s_1, s_2, \widehat{S}, s_{k-1}, s_k, S^{''}, \bar{S}\}$. COMPRESS is operated when the

Table 12
The APD for t_{max} = 180 s.

n	m							
	8	10	12	14	16	18	20	22
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
22	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
28	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
32	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.394
34	0.000	0.000	0.000	0.000	0.000	0.000	0.000	−0.919
36	0.000	0.000	0.000	0.000	0.000	−1.619	0.000	−0.338
38	0.000	0.000	0.000	0.000	0.000	0.000	−1.050	−0.174
40	0.000	0.000	0.000	0.000	0.000	0.000	−0.097	−0.660
42	0.000	0.000	0.000	0.000	−0.628	−0.764	0.000	−3.836
44	0.000	0.000	0.000	0.000	0.000	−0.980	−1.666	−1.684
46	0.000	0.000	0.000	0.000	−0.983	−2.402	−3.984	−0.182
48	0.000	0.000	−1.052	−0.602	−2.916	−0.870	−1.316	−3.015
50	0.000	0.000	0.000	0.000	−2.600	−1.837	−0.785	−3.084
52	0.000	0.000	−2.731	−0.840	−2.193	−3.396	−3.652	−1.082
54	0.000	−0.562	−1.642	−3.712	−3.873	−2.166	−3.225	−4.101
56	0.000	0.000	0.000	−2.170	−1.686	0.064	−3.901	−3.440
58	0.000	0.000	−1.197	−2.868	−0.578	−4.219	−8.461	−6.262
60	0.000	0.000	−2.196	−3.437	−2.003	−3.196	−4.132	−5.010
62	0.000	0.000	−1.796	0.000	−3.934	−3.959	−7.284	−5.288
64	0.000	−1.004	−5.923	−1.125	−1.084	−6.276	−4.940	−7.629

total holding costs decreases as the sequence changes from Q_1 to Q_2. Note that when the actor a_i participates in three scenes, s_2 and s_{k-1} are same, and the sequence $\widehat{S}$ is empty. Similarly, when the actor a_i plays in two scenes, s_2, s_{k-1}, s_k are same, and $\widehat{S}$ and S'' are empty. Note that this move is applicable only for the actors who participate in two or more scenes. In contrast to the previous neighborhood structures which iterate over the scenes, the COMPRESS move iterates over the actors. Tables 6 and 7 indicates an example of the COMPRESS move over the example problem. The holding costs of the solution presented in Table 6 is 1120 cost unit. If we apply the COMPRESS move to a_5, s_3 is inserted immediately before s_5 and s_4 is inserted immediately after s_5. This move reduces holding costs from 1120 to 105 cost unit.

3.2. Local search

In this section, the variable neighborhood descent (VND) procedure, which is used as local search in GVNS, is described. This VND systemically explores the neighborhoods that are mentioned in the previous section. The different arrangements of the neighborhood structures are examined and results are presented in the computational results section. The VND procedure uses a first improvement strategy and changes the neighborhood sequentially, implying it jumps to the next neighborhood structure if there is no improvement in the current neighborhood structure (Todosijević, Mjirda, Mladenović, Hanafi, & Gendron, 2017). On the other hand, it starts from the first neighborhood structure again if an improvement has been found. The algorithm is stopped whenever no improvement is found in the all neighborhood structures in an iteration. The details of the VND are presented in the Algorithm 2.

Input: X (A solution), N (Set of neighborhood structures), f (Objective function)
  $k \leftarrow 1$;
  while $k \leq 8$ **do**
    $X' \leftarrow$ Local Search (X, N_k);
    if $f(X') < f(X)$ **do**
      $X \leftarrow X'$;
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    End
  End

Table 13
Summary comparative results.

t_{max}(seconds)		1	10	30	60	180
(I)	APD	2.107	0.201	0.018	0.011	0.001
	MAX PD	41.242	10.443	2.407	1.058	0.465
	MIN PD	0.000	0.000	0.000	0.000	0.000
	#Positive PDs	347	90	24	9	3
	# Zero PDs	475	732	798	813	819
	# Negative PDs	0	0	0	0	0
(II)	APD	1.709	−0.519	−0.830	−0.872	−0.892
	MAX PD	41.242	10.903	2.407	1.058	0.465
	MIN PD	−15.086	−19.347	−19.460	−20.173	−20.173
	#Positive PDs	434	102	25	9	3
	# Zero PDs	476	734	800	815	821
	# Negative PDs	90	164	175	176	176
(III)	APD	−0.126	−3.848	−4.749	−4.952	−5.015
	MAX PD	15.429	10.903	1.503	0.000	0.000
	MIN PD	−15.086	−19.347	−19.460	−20.173	−20.173
	#Positive PDs	87	12	1	0	0
	# Zero PDs	1	2	2	2	2
	# Negative PDs	90	164	175	176	176

3.3. Shaking procedure

In order to escape from local optimum traps, a shaking procedure is employed within the solution algorithm. The shaking procedure performs s times a random 2-opt move. The details of the shaking procedure are presented in the Algorithm 3.

Input: X (A solution), s (The number of moves at a run of shaking procedure)
for $i = 1$ **to** s **do**
  Perform a random 2-opt move on X;
End

4. Computational experiments

We have coded our developed GVNS algorithm in C++ using the Microsoft Visual C++ 2010 compiler and performed all computational experiments on a PC Intel® Core ™ i7@ 3.1 GHz processor with 4 GB RAM.

4.1. Benchmark instances and parameter setting

In order to evaluate our developed GVNS algorithm, we performed computational results using two benchmark data sets, named Types 1 and 2. The Type 1 data set was introduced by Cheng et al. (1993) and Smith (2005) and includes seven instances, namely MobStory, film103, film105, film114, film117, film118 and film119. The size of these instances varies from 18 × 8 (18 scenes by 8 actors) to 28 × 8. Also, we used the Type 2 data set introduced by de la Banda et al. (2011). The size of instances of the Type 2 data set varies from 16 × 8 to 64 × 22, including 200 different combinations for the number of scenes and the number of actors and 20,000 instances (100 random instances for each combination is generated). Both data set and some of the solutions can be downloaded from http://www.tigerqin.com/publicatoins/talent-

Table 14
Impact of the neighborhood structures and moves.

		Base	COMPRESS	3-OR-opt	2-OR-opt	1-OR-opt	2-opt
(I)	APD	0.011	0.016	0.017	0.018	0.032	0.014
	MAX PD	1.058	1.493	1.931	3.187	3.097	4.867
	MIN PD	0.000	0.000	0.000	0.000	0.000	0.000
	#Positive PDs	9	12	18	18	45	11
	# Zero PDs	813	810	804	804	777	811
	# Negative PDs	0	0	0	0	0	0
(II)	APD	−0.872	−0.867	−0.860	−0.863	−0.828	−0.869
	MAX PD	1.058	1.493	1.931	3.187	3.097	4.867
	MIN PD	−20.173	−20.173	−20.173	−20.173	−20.173	−20.173
	#Positive PDs	9	12	19	18	46	11
	# Zero PDs	815	812	806	806	778	813
	# Negative PDs	176	176	175	176	176	176
(III)	APD	−4.952	−4.936	−4.910	−4.930	−4.800	−4.931
	MAX PD	0.000	0.000	1.355	0.000	0.175	0.000
	MIN PD	−20.173	−20.173	−20.173	−20.173	−20.173	−20.173
	#Positive PDs	0	0	1	0	1	0
	# Zero PDs	2	2	2	2	1	2
	# Negative PDs	176	176	175	176	176	176

scheduling-problem, provided by Qin et al. (2016). For each combination of n and m in Type 2 data set, the best known solutions of only 5 out 100 instances are reported, resulting in 1000 test instances. In order to compare our results with already developed solution approaches in the literature, we consider only 1000 instances with available solutions from the Type 2 data set.

There are two parameters in our GVNS, i.e. s_{max} and t_{max}. We consider $s_{max} = \frac{n}{2}$ using fine tuning and report our results for $t_{max} = 1, 10, 30, 60$ and 180 s.

4.2. Comparative results

Since test instances of the Type 1 data set are easy, our developed GVNS is able to find the optimal solutions of all these instances in less than 0.1 s. Thus, the challenging instances of Type 2 are more suitable for comparative results. Qin et al. (2016) have reported that solutions of 822 out of 1000 instances of Type 2 are optimal and solutions of 178 other instances are only upper bounds.

The comparative results, based on average percent deviation (APD), are presented for $t_{max} = 1, 10, 30, 60$ and 180 s in Tables 8–12, respectively. The percent deviation (PD) is calculated as $\frac{\text{obtained solution by GVNS} - \text{best known solution}}{\text{best known solution}} \times 100$ and APD is the average of percent deviations over 5 test instances for each combination of n and m of Type 2 data set. It should be noticed that the most of best known solutions in the literature are obtained by Qin et al. (2016). The cells indicate negative APD imply improvement in the best found solutions. As it was expected, by increasing the CPU run time, the APD is decreased for most combinations of n and m. Although Qin et al. (2016) develop a branch and bound algorithm and do not report their results within time limits, by comparing the following tables with Table 12 of them, we fairly conclude that our developed algorithm is more efficient especially for larger size instances.

Table 13 presents the summary comparative results based in three sets of instances. The results presented in the row (I) is based on 822 instances of Type 2 data set for which optimal solutions are available. In the row (II), the best found solutions of all 1000 test instances of Type 2 data set are considered while in the row (III) only 178 test instances, for which optimal solutions are not found in the literature, are considered. The results of Table 13 indicates that our GVNS is really efficient because it is able to find optimal solutions of 819 out of 822 instances (99.6%) while for the three remaining test instances the APD is around 0.4. Also, GVNS has been able to improve the best found solutions of 176 out of 178 instances (98.9%) with non-optimal solution where the average improvement is around 5 percent.

4.3. Impact of neighborhood structures

In this section, we analyze the impact of different moves based on the different neighborhood structure, described in Section 3.1. In order to investigate the impact of a move, we omit it from the GVNS and run the algorithm to study the change of APD. Since for most of the instances, noticeable improvement is not obtained after 60 s, we assume $t_{max} = 60$ s in this section. The new results are presented in Table 14 in which the column “Base” indicates the condition for which all moves are included in GVNS. Also, for example, the column “COMPRESS” presents the condition for which the COMPRESS move is excluded from GVNS. Furthermore, the columns x-OR-opt moves imply excluding both forward and backward types of that moves from the algorithm. The general structure of Table 14 is similar to Table 13. The results of Table 14 indicates that excluding each move from GVNS has a negative impact on the performance of the algorithm but the 1-OR-opt has the most significant impact.

5. Conclusions and future research directions

In this paper, we consider the talent scheduling problem and modify a previously developed formulation of the problem and develop an efficient metaheuristic algorithm based on the generalized variable neighborhood search algorithm for the problem. Our developed GVNS is able to find optimal solutions of 99.6% of test instances for which optimal solutions have been reported in the literature. Also, it improves the best known solutions of 98.9% of test instances for which only upper bounds have been informed in the previous researches. These computational results indicate that our developed GVNS is efficient and outperforms already developed algorithms in the literature for the talent scheduling problem.

For the future research, we suggest to extend our GVNS for the talent scheduling problem by considering more realistic assumptions such as setup times between scenes or stochastic durations for scenes. Incorporating the realistic assumptions make the talent scheduling problem even more complicated. Therefore, developing additional solution algorithms to the talent scheduling problem seems to be an interesting research topic to solve more broad classes of the problem.

Acknowledgement

This work is supported by Ferdowsi University of Mashhad as a research project with number 43212 and date 13-2-2017.

References

Adelson, R. M., Norman, J. M., & Laporte, G. (1976). A dynamic programming formulation with diverse applications. *Operational Research Quarterly (1970-1977), 27*(1), 119–121 Part1.

Bomsdorf, F., & Derigs, U. (2008). A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. *OR Spectrum, 30*(4), 751–772.

Cheng, T. C. E., Diamond, J. E., & Lin, B. M. T. (1993). Optimal scheduling in film production to minimize talent hold cost. *Journal of Optimization Theory and Applications, 79*(3), 479–492.

De la Banda, M. G., Stuckey, P. J., & Chu, G. (2011). Solving talent scheduling with dynamic programming. *INFORMS Journal on Computing, 23*(1), 120–137.

Fink, A., & Voß, S. (1999). Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research, 26*(1), 17–34.

Johnson, D. S., & McGeoch, L. A. (1995). The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts, & J. K. Lenstra (Eds.). *Local Search in Combinatorial Optimization* (pp. 215–310). New York: Wiley.

Mjirda, A., Todosijević, R., Hanafi, S., Hansen, P., & Mladenović, N. (2016). Sequential variable neighborhood descent variants: An empirical study on the traveling salesman problem. *International Transactions in Operational Research, 24*(3), 615–633.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research, 24*, 1097–1100.

Nordström, A. L., & Tufekci, S. (1994). A genetic algorithm for the talent scheduling problem. *Computers & Operations Research, 21*(8), 927–940.

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*(1), 412–426.

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

Smith, B. M. (2005). Caching search states in permutation problems. In P. van Beek (Vol. Ed.), *Lecture Notes in Computer Science: . Vol. 3709*. Berlin, Heidelberg: Springer.

Todosijević, R., Mjirda, A., Mladenović, M., Hanafi, S., & Gendron, B. (2017). A general variable neighborhood search variants for the travelling salesman problem with draft limits. *Optimization Letters, 11*(6), 1047–1056.