Iterative local search methods for the talent scheduling problem

Yury Kochetov
Sobolev Institute of Mathematics, Novosibirsk State University
Novosibirsk, Russia
jkochet@math.nsc.ru

Abstract

We study the talent scheduling problem which can be reformulated as the column permutation problem for a binary matrix. The problem is NP-hard in the strong sense. Modified greedy algorithm and iterative local search metaheuristics are developed for this problem. Computational experiments show that we can find optimal or near optimal solutions easily by these methods if number of columns and number of rows of the matrix are at most 100.

Keywords Local search, greedy algorithms, metaheuristics.

1 Introduction

In the talent scheduling problem we assume that a film production project is divided into n independent shooting days (scenes). There are m talents (actors) involved at different scenes of the project. An $(m \times n)$ binary matrix A indicates the scenes for each talent. The element a_{ij} of the matrix is set to 1 if talent i is needed for scene j and it is 0 otherwise. The per-diem payment for talent i is a positive value w_i. Each talent starts to job in his first shooting day and finishes in his last shooting day. We wish to find a permutation of scenes to minimize the total payment for the talents.

If there is a permutation without delays of the talents (free days) then the total payment is $\sum_{i=n}^{n} w_i \sum_{j=1}^{m} a_{ij}$ and the problem is polynomially solvable [1, 2]. If it is not the case, the problem is NP-hard. The branch and bound method for the problem is presented in [3]. It allows us to solve the problem for small dimensions only, $n = m \leq 15$. Three integer linear programming formulations of the problem are studied in [4]. They can be used for solving the problem by commercial software. Unfortunately, the integrality gap is large for these formulations and solver CPLEX 11.0 can not find global optimum for $n = m = 30$ during 24 hours on Pentium 2.8 GHz. The dynamic programming method for the problem is presented in [5]. We can solve the problem for small dimensions by the method again, $n = m \leq 30$. The first metaheuristic approach, a genetic algorithm, in presented in [6]. It is not strong and tested for $m \leq 40, n \leq 70$ only.

In this paper we develop three metaheuristic algorithms: a hybrid simulated annealing, stochastic tabu search and genetic local search. We show that these algorithms can discover the optimal solutions for random generated instances with $n = m \leq 100$ quickly. They use a new constructive heuristic to create good starting solutions and three neighborhoods *Insert, Rotate*, and *k-Opt* for improving a current solution or population of solutions. These metaheuristics incorporate idea of Variable Neighborhood Search method [7] to change neighborhoods during the iterative process.

The paper is organized as follows. Section 2 introduces the notations and the mathematical model. Section 3 describes the constructive greedy heuristics. Three neighborhoods for local search are presented in Section 4. The stochastic tabu search, hybrid simulated annealing, and genetic local search are described in Sections 5, 6 and 7 correspondingly. Computational results are discussed in Section 8. In final Section 9, we give some conclusions and directions for future research.

2 Mathematical model

Let us introduce the following variables:
e_i is the earliest shooting day for talent i;
l_i is the latest shooting day for talent i;
x_{jt} is the scheduling for the project, i.e.

$$x_{jt} = \begin{cases} 1 & \text{if scene } j \text{ is scheduled in day } t \text{ of shooting} \\ 0, & \text{otherwise.} \end{cases}$$

We wish to minimize the total payment $F(e, l, x)$ defined by

$$F(e, l, x) = \sum_{i=1}^{m} w_i(l_i - e_i + 1)$$

subject to

$$\sum_{j=1}^{n} x_{jt} = \sum_{t=1}^{n} x_{jt} = 1, \qquad 1 \le j, t \le n;$$

$$a_{ij}e_i \le \sum_{t=1}^{n} a_{ij}tx_{jt} \le l_i, \qquad 1 \le i \le m, \quad 1 \le j \le n;$$

$$x_{jt} \in \{0, 1\}, \qquad 1 \le j, t \le n;$$

$$l_i, e_i \ge 1, \text{ integer}, \qquad 1 \le i \le m.$$

The objective function is the total payment for the talents. The first restriction requires one day for each scene and one scene for each day. The second restriction defines the earliest and latest days for each talent. Other restrictions are integrality constraints. It is easy to see that the last restriction can be omitted. The optimal solution is not changed in this case. Thus we have got the mixed integer linear program with n^2 binary variables x_{jt} and nonnegative $2m$ continuous variables e_i and l_i. Unfortunately, this formulation has large integrality gap as the previous pure integer formulations [4] and we can solve the problem by CPLEX software for small dimensions only. For large scale instances we develop metaheuristics which can discover optimal or near optimal solutions.

3 Fast constructive algorithms

As we have mentioned above, the problem is polynomially solvable if there is a scheduling without free days of the talents. For testing this property, a PQ–tree data structure is applied [2]. It is polynomial and sophisticated algorithm. It carries out at most m iterations. In each iteration we consider a subset of talents, and the current permutation of scenes is modified in order to remove free days for an additional talent. If it is not possible, the algorithm terminates with answer "No". If we do that for all talents, we obtain the desired permutation and the algorithm terminates with answer "Yes". Below we present two greedy algorithms for the general case. The first naive algorithm carries out n iterations. It starts by selecting a scene for the first day. In each iteration a subset of scenes is considered and we select an additional scene for this subset to minimized an additional payment for the talents.

There are at least two weak points in this framework. It is not clear how to select the first scene. But we can test all scenes and return the best found solution. In each iteration, we

consider a subset of scene and calculate the objective function for this subset only. It may be not correct. For some talents we must pay for free days because they will be needed later. Now we present a modified greedy algorithm where in each iteration we check whether a talent will be needed for unscheduled scenes or not. An auxiliary set J is used for the unscheduled scenes and permutation σ is a result of the algorithm.

Modified Greedy Algorithm

1. Put $J \leftarrow \{1, \dots, n\}$, $I \leftarrow \emptyset$.
2. Select a scene j for the first day, put $\sigma(1) \leftarrow j$, $J \leftarrow J \setminus \{j\}$.
3. For i:=1 to m do if $a_{ij} = 1$ then $I \leftarrow I \cup \{i\}$. { Identify the involved talents }
4. For each day $t := 2$ to n do
   4.1. For each $i \in I$, if $\sum_{j \in J} a_{ij} = 0$ then $I \leftarrow I \setminus \{i\}$. { Remove unused talents }
   4.2. Find a scene j_0 with minimal additional payment
   $$\Delta_{j_0} = \min_{j \in J} \Big\{ \sum_{i \in I} w_i(1 - a_{ij}) + \sum_{i \notin I} w_i a_{ij} \Big\}$$
   4.3. Put $\sigma(t) \leftarrow j_0$; $J \leftarrow J \setminus \{j_0\}$.
   4.4. For each $i \in I$, if $a_{ij_0} = 1$ then $I \leftarrow I \cup \{i\}$. { Involve new talents}
5. Return the permutation σ.

Figure 1 shows computational results for the greedy algorithms. Each point at the curves corresponds to the average deviation from the optimum for 30 randomly generated instances, $n = m = 100$. For each instance we apply the algorithms 100 times with each scene for the first day and return the best found solution. As we can see, the algorithms produce near optimal solutions for instances with high density, $d = \sum_{ij} a_{ij}/mn$. For low density $d = 0.05$ the average deviation is large, it is about 50% for the naive greedy algorithm and about 10% for the modified greedy algorithm. The instances with low density are the most difficult. Thus, the remain part of the paper is devoted to metaheuristics for this difficult case.

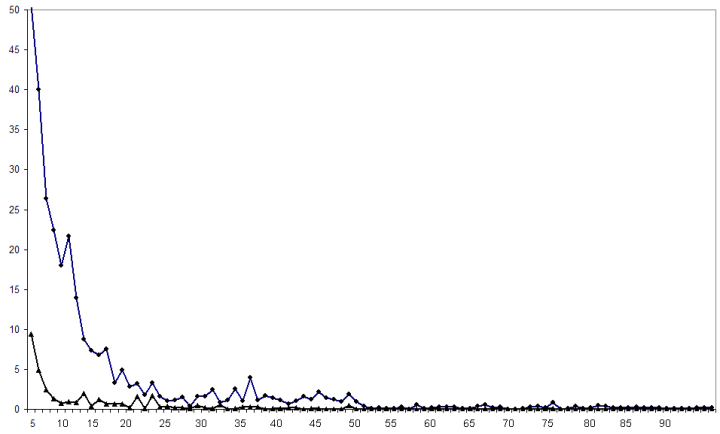

Figure 1: Average deviations of the greedy algorithms

4 Neighborhoods

For a permutation σ we define the following neighborhoods.

The *Insert* neighborhood is the set of all permutations which can be obtained from σ by insertion an element (scene) to a new position (day).

The *Rotate* neighborhood is the set of all permutations which can be obtained from σ by choosing two elements and using inverts order for these elements and all elements between them.

The *k-Opt* neighborhood is the set of all permutations which differ from σ in at most k positions.

It is easy to verify that the neighborhoods have the following important property. For two arbitrary permutations σ' and σ'' there is a finite number q and permutations $\sigma^l, l = 0, \ldots, q$ such that $\sigma^0 = \sigma'$, $\sigma^q = \sigma''$ and each permutation σ^l is neighboring for σ^{l-1}. In other words, using each neighborhood we can reach arbitrary solution (for example, optimal one) from arbitrary starting solution. This property is important for asymptotic properties of the iterative processes based on the Markov Chains [8, 9].

5 Stochastic tabu search

In this section we present a variant of the well-known tabu search algorithm [10] which uses three randomized neighborhoods. For a neighborhood N, the randomized neighborhood N_p with parameter p is a subset of N. Each element of N is included in N_p with probability p independently from other elements. The neighborhood N_p may be found empty or coincide with N. By definition of the *Insert*, *Rotate* and *k-Opt* neighborhoods, each neighboring solution is defined by at most k positions in current permutation. We store these positions and use three tabu lists, one for each neighborhood. The length of each list is a constant. Following [9], we use small lists for small p. Below we show the pseudocode for the stochastic tabu search heuristic. Parameters Δ_{in} and Δ_{div} define number of iterations for intensification and diversification stages. The earliest and latest days for each talent are easily calculated by the scheduling x_{jt}. Thus we use notation $F(x)$ for the objective function instead of $F(e, l, x)$.

Stochastic Tabu Search

Initialization: Create a starting solution (x_{jt}) and empty tabu lists;
put $(x^*_{jt}) \leftarrow (x_{jt})$, $N \leftarrow$ *Rotate*, $t \leftarrow 1, t_{in} \leftarrow -\Delta_{in}, t_{div} \leftarrow -\Delta_{div}, p \leftarrow p^0$;

Iterative process: **While** not termination condition **do**

1. If $t > t_{in} + \Delta_{in}$ then $N \leftarrow$ *Rotate*; If $t = t_{div} + \Delta_{div}$ then $p \leftarrow 2p$;
2. Find the best solution (x'_{jt}) in $N_p(x_{jt})$ without forbidden elements;
3. Put $(x_{jt}) \leftarrow (x'_{jt})$; $t \leftarrow t + 1$; update the corresponding tabu list;
4. If $F(x^*_{jt}) > F(x_{jt})$ then $(x^*_{jt}) \leftarrow (x_{jt})$;
5. If intensification condition met then $t_{in} \leftarrow t$; $N \leftarrow Insert \cup$ 2-*Opt*; $(x_{jt}) \leftarrow (x^*_{jt})$;
6. If diversification condition met then $t_{div} \leftarrow t$; $p \leftarrow p/2$;

end while

Out computational results show that the *Rotate* neighborhood produces better local optima than the *Insert* and 2-*Opt* neighborhoods. We use the *Rotate* neighborhood for local search and other neighborhoods for intensification. In order to diversify the search process we decrease the size of the randomized neighborhood at Step 6. As a result, the local search becomes more chaotic, and after Δ_{div} iterations we continue the search in a new area of the feasible domain. The starting value p^0 is selected as 0.10.

6 Hybrid simulated annealing

This algorithm explicitly combines ideas of two well-known metaheuristics: simulated annealing [8] and variable neighborhood search [7]. We use framework of the annealing and systematically change neighborhoods in this iterative process. The local improvement algorithm is applied to intensify the search after decreasing the temperature. For diversification we apply a random walk by the 3-*Opt* neighborhood. Below we show the pseudocode for our hybrid algorithm.

Hybrid Simulated Annealing

Initialization: Create a starting solution (x_{jt}); put $t \leftarrow 1, (x^*_{jt}) \leftarrow (x_{jt})$,
define starting temperature c and cooling factor γ;
Iterative process: **While** not termination condition **do**

1. For $t := 1$ to T_{max} do
   1.1. Select a neighborhood;
   1.2. Pick out a neighboring solution (x'_{jt}) and calculate $\Delta = F(x'_{jt}) - F(x_{jt})$;
   1.3. If $\Delta \leq 0$ then $(x_{jt}) \leftarrow (x'_{jt})$ else $(x_{jt}) \leftarrow (x'_{jt})$ with probability $e^{-\Delta/c}$;
   1.4. If $F(x^*_{jt}) > F(x_{jt})$ then $(x^*_{jt}) \leftarrow (x_{jt})$;
   1.5. If diversification condition met then apply a random walk under 3-*Opt* neighborhood.
2. Decrease the temperature $c \leftarrow c\gamma$;
3. Apply the local improvement algorithm under *Insert* and 2-*Opt* neighborhoods;

end while

The good starting solution we obtain by modified greedy algorithm. It allows us to start with low temperature. The maximal number of iterations T_{max} for the given temperature is defines as the sire of the *Rotate* neighborhood. The cooling factor γ we put 0.99, the starting temperature is 18.

7 Genetic local search

As we have mentioned in Introduction, the first genetic algorithm for this problem is presented in [6]. In this algorithm the reproduction operator selects a number of solutions from the current population for crossing depending on their fitness. This is done by constructing a Monte Carlo wheel. A linear scaling of the fitness function is used for the wheel. Crossover is the well-known randomized two points operator. *Mutation* is a random step under the 2-*Opt* neighborhood. The starting population is generated at random. Below we show another genetic algorithm which

dominates the previous one. The main operators are modified in order to use the scheduling structure of the problem. Moreover, we apply the local improvement algorithm for each element of population to intensify the search process.

Genetic Local Search

Initialization: Create a starting population, put (x^*_{jt}) as the best solution in the population;
Iterative process: **While** not termination condition **do**

1. For $l := 1$ to L do
   1.1. Select two parents from the population;
   1.2. Produce an offspring solution;
   1.3. Apply mutation operator with a small probability;
   1.4. Include the obtained solution into the population;
2. Remove L the worst solutions from the population;
3. Update the best found solution (x^*_{jt});
4. If intensification condition met then apply local improvement algorithm under the randomized *Insert* and 2-*Opt* neighborhoods for each element of the population;

end while

The starting population is generated by modified greedy algorithm. We create n solutions using each scene for the first day and select an appropriate number of the best solutions. The parents are taking at random but one of them is the best in the current population. The offspring solution is produced as follows. We consider two permutations and move $\lceil n/2 \rceil$ random positions from one of them into the new permutation. The last positions are fulfilled according to the second permutation. The mutation operator is a moving to neighboring solution under one of *Insert*, *Rotate* or 2-*Opt* neighborhoods. The local improvement is applied if the best found solution is not changed during some iterations. For randomized neighborhoods we put $p = 0.05$. Size of the population is 10, and $L = 10$.

8 Computational results

The presented heuristic algorithms are coded in C++ and tested on random generated test instances with known optimal solutions and instances from the previous publications [3, 5, 6]. For random instances we put $n = m = 100, w_i \in [1, 100], d = 0.05$. For each talent we generate two scenes j_1, j_2 at random and put $a_{ij} = 1$ for $j = j_1, j_1 + 1, \ldots, j_2$ and $a_{ij} = 0$ otherwise. Thus we have optimal solution. Then we apply a random permutation for the columns of the matrix and use the result as the benchmark. Table 1 show the average deviation from the optimum for such 10 test instances. In brackets it is shown the number of trials from 10 when the optimal value is discovered. As we see these heuristics show small average deviation and can find global optima. The termination condition is the same for all methods: $15 \cdot 10^5$ objective function calculations. The running time is a few minutes on Pentium 2.8 GHz under the Windows XP Professional operating system.

Table 1. Average deviations from the optimum (%)

N	Naive Greedy Algorithm	Hybrid Simulated Annealing	Genetic Local Search	Stochastic Tabu Search
1	1	0 (10)	0.001 (9)	0 (10)
2	22.1	0 (10)	0.006 (7)	1.82 (6)
3	54.35	0 (10)	0 (10)	4.73 (6)
4	24.73	0 (10)	0 (10)	1.10 (4)
5	53.8	0 (10)	0 (10)	2.92 (7)
6	63.46	0.23 (8)	0.006 (9)	3.87 (4)
7	4.81	0 (10)	0 (10)	0.01 (9)
8	16.2	0.15 (9)	0.006 (8)	1.20 (5)
9	5.74	0 (10)	0 (10)	0 (10)
10	88	0 (10)	0 (10)	0 (10)

It is interesting to note that when we change the test instances and include one or two zeros between j_1 and j_2 for some talents and apply our heuristics, we discover solutions with better value than $\sum_{i=1}^{m} w_i \sum(j_2 - j_1 + 1)$. We do not know optimal values for this case but we have good near optimal values. Nevertheless, our iterative methods can improve these values.

To compare our methods with previous ones we test them on the instances available from internet [5, 6, 11]. For all cases we have $n \leq 70, m \leq 100$. Our methods have found the best known solutions quickly. Thus, we conclude that our local search methods can find the optimal or best known solutions with high frequency.

9 Conclusion and future research

In this paper we consider the talent scheduling problem and present iterative local search methods for solving this problem. We show that these methods can find the optimal solutions or the best known solutions if the number of talents and the number of scenes are at most 100.

For future research it is interesting to study a more general case of the problem when each scene has a time duration and we can shoot some scenes every day. Again, each talent starts to job in his first shooting day and finishes in his last shooting day. This new problem includes some aspects of the classical bin packing problem and may be more interesting from practical point of view. For theoretical research some questions are still open. For example, it is not clear how difficult the local search problem for each *Insert*, *Rotate*, and *k-Opt* neighborhoods. If daily payments are the same for all talents, in particular, $w_i = 1$ for all $i = 1, \ldots, m$, we can find local minimum in polynomial time by the simple local improvement algorithm. But for general case this local search problem may be PLS-complete [12].

10 Acknowledgments

- Many thanks to students of the Novosibirsk State University Mikhail Sivyh, Aleksey Hmelev, and Andrey Yakovlev for thier efforts for coding and testing of the optimization methods presented in the paper.

- This research was partially supported by RFBR grants 09-01-00059, 11-07-00474 and ADTP grant 2.1.1./3235.

References

[1] Veldhorst M., 1985. Approximation of the consecutive ones matrix augmentation problem. *SIAM Journal on Computing*, Vol. 14, pp. 709–729.

[2] Booth K. S., Lueker G. S., 1976. Testing for the consecutive ones property, intervals graphs, and graph planarity using PQ-tree algorithm. *Journal of Computer and System Science*, Vol. 13, pp. 335–379.

[3] 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*, Vol. 79, No. 3, pp. 479–492.

[4] Kononova P. A., Kochetov Yu. A., 2008. On the column permutation problem for a 0–1 matrix. *Proceedings of the 14-th Baikal international school–seminar Optimization Methods and Their Applications*, Irkutsk, Vol. 1, pp. 444–451.

[5] de la Banda M. G., Stuckey P. J., Chu G., 2011. Solving talent scheduling with dynamic programming. *INFORMS Journal on Computing*, Vol. 23, No. 1, pp. 120–137.

[6] Nordström A. L., Tufekci S., 1994. A genetic algorithm for the talent scheduling problem. *Computers & Operations Research*, Vol. 21, pp. 927–940.

[7] Hansen P., Mladenović N., 2002. Developments of Variable Neighborhood Search. In: Ribeiro C. C., Hansen P. (Eds.) *Essays and Surveys in Metaheuristics*. Kluwer Academic Publishers, pp. 415–440.

[8] Aarts E. H. L., Korst J. H. M., Laarhoven van P. J. M., 1997. Simulated annealing. In: Aarts E. H. L. Lenstra Y. K. (Eds.) *Local Search in Combinatorial Optimization*. Wiley, Chichester, pp. 91–120.

[9] Goncharov E. N., Kochetov Yu. A., 2002. The probabilistic tabu search algoritm for the unconstrained discrete optimization. *Discrete Analisys and Operations Research, Seria 2*, Vol. 9, No. 2, pp. 13–30 (in Russian).

[10] Glover F., Laguna M., 1997. *Tabu Search*. Kluwer Academic Publishers, Boston.

[11] http://ww2.cs.mu.oz.au/ pjs/talent/

[12] Kochetov Yu. A., 2008. Computational bounds of local search in combinatorial optimization. *Computational Mathematics and Mathematical Physics*, Vol. 48, No. 5, pp. 788–807.