A Memetic Algorithm for Operating Room Scheduling[1]

H. Fei[2, 3], A.Artiba[2], C.Chu[3]

Summary

Operating theatre is always the most important and expensive sector of hospitals. Considering the recent environment pressure, this sector is surely to be significant to health-care institutions' managers when they regard cost reduction programs as a primary tool for managing the system's profitability. Our study aims to find out an efficient tool to minimize the total overtime cost of the operating theatre planning, especially the operating rooms (ORs) scheduling.

Introduction

Since the last decade, the medical system is under the pressure of environment's change, such as the increasing population of old people and various diseases. Faced with such situation, hospitals have to take actions to develop the productive and quality of their service. What's more, the largest hospital cost category was the operating theatre according to some study results. Therefore, how to reduce the operating theatre's cost is always the hospitals' manager's objective. Meanwhile, the development of tools and methods for optimizing the function of this surgical unit is definitively to be many researchers' focus.

In that case, we will try to find out a good solution for daily operating room scheduling problem in order to minimize total overtime cost.

This paper is organized as follows: in section 1, the description is given for daily operating room scheduling problem; then, in section 2, we present the procedure of proposed memetic algorithm. At the end of this paper, computational experiment's result is list.

1 Problem description

The daily operating room scheduling problem can be described as follows:

From the beginning of a hospital's working day, there are N patients waiting to be operated in operating rooms (Supposed that there are M operating rooms available that day). For simplifying this problem, some hypothesis are given: 1) All the patients' pre-operations have already been done when the operating rooms open. 2) All the operating rooms are polyvalent so that each surgical operation can be operated in any of them. 3) In addition, the surgical operations can't be stopped while being operated. 4) All the other resources needed by the surgical operations, such as the surgical equips, the surgical equipments and recovery rooms' beds, are sufficient in this working day. 5) Moreover, for the sake of simplification, the urgent patients are supposed to be treated in other special operating rooms so that we can only consider the ordinary patients' schedule in this paper.

Informed by statistic materials, we can know in advance that each surgical operation's operating time. What's more, according to most hospitals' timetable, all the M operating rooms

[1] This research is included in project PAI which is financed by Belgian Science Policy.
[2] Facultés Universitaires Catholiques de Mons, Mons, Belgique.
[3] Université de Technologie de Troyes, Troyes, France

are opened and closed at a simultaneous time, respectively. Thus, a standard can be made that all the M operating rooms are ready at time 0 and will close at time *d*. In practice, if some surgical operations have to be finished after the supposed closing time *d*, the hospital should pay a certain fee for the additional time (overtime). Thus, the simplified operating room scheduling problem has a general objective to assign a number of patients into available operating rooms, in the simplest case they are identical, by minimizing the total overtime cost.

Considering the description above, the operating rooms scheduling problem can be regarded as a special case of industrial scheduling problems, thus it is possible for us to treat our described problem as a parallel machines problem aiming at minimizing total weighted tardiness $\sum w_j T_j$, where the weights w_j, $j= 1, . . ., N$ are related to the operating cost of surgical operations; $T_j=max\{0,C_j-d\}$ is surgical operation *j*'s possible additional time, C_j is surgical operation *j*'s completion time and *d* is the operating rooms' regular closing time.

2 A proposed memetic algorithm

Since minimizing-total-weighted-tardiness scheduling problems have already been proved to be strongly NP hard even for single machine case (Lenstra et al. 1977 [1]), the parallel machines total weighted tardiness problem is strongly NP hard in general. In conclusion, the currently stated problem is also NP hard. As we all know, even for the single machine case, instances with more than 50 jobs cannot be solved to optimality with state of the art of branch and bound algorithms (Matthijs den Besten et al., 2000 [2]). Therefore, many researchers have focused their study on heuristic methods, such as Genetic Algorithm (GA), Tabu Search, Annealing Simulation (AS) and so on, for solving NP-hard problems.

Considering the good performance of GAs for NP-hard problems, especially motivated by Paulo M. França et al., 2001 [3], where a local-search-based memetic algorithm was proposed for a total tardiness single machine scheduling problem, we will also develop a heuristic-based memetic algorithm for the parallel operating rooms scheduling problem.

2.1 Procedure of proposed heuristic-based memetic algorithm

Before describing our heuristic-based memetic algorithm procedure, some parameters are given: *Pop_size* (The individuals' number of the constructed population); P_m (The probability of mutation for a certain selected individual); P_c (The probability of crossover for a couple of selected parents); *Max_Itr_Reproduce* (The maximum reproduction iteration times).

Step1: (initialization) construct an initial population $POP=\{Indiv(i)| \ i=1,...,Pop\_size\}$, where individual' chromosome structure will be mention in section 2.2.

Step 2: (Heuristic For Local amelioration) A COVERT heuristic procedure, mentioned in section 2.2, is applied for each individual *indiv(i), i=1,2,..., Pop_size* , in order to ameliorate its performance.

Step 3: (evaluation) Calculate each individual *indiv(i)*'s fitness F_i of assembly *Pop* and form a fitness array $Fit=(F_1,F_2,...,F_{pop\_size})$.

Step 4: (crossover) Select two individuals randomly by making a roulette wheel of fitness array *Fit,* crossbreed them for a new individual, and perform the COVERT procedure for this new individual, then accept the newly obtained individual *indiv_C* as a progeniture. After doing this crossover step $PPc=Pop\_size*P_c$ time, we can get a progenitures' assembly $Offspring=\{Indiv\_C_1, Indiv\_C_2,\dots,Indiv\_C_{PPc}\}$.

Step 5: (Mutation) Select one individual randomly from assembly *Pop* or *Offspring* , mutate it according to a proposed mutation rule (see section 2.2) for a new progeniture *Indiv_M*, Do this mutation action $PPm=Pop\_size*P_m$ times and add all the new generated progenitures into the former progenitures' assembly *Offspring* obtained in step 4, then we can get an extended assembly: $Offspring=\{Indiv\_C_i|I=1,\dots,PPc\}U\{Indiv\_M_j|j=1,\dots,PPm\}$.

Step6: (Elitist selection) Select the individual, who performs best among all the individuals of assemblies *Pop* and *Offspring*, and let it be one member of the new population;

Step 7: (Reproduction) Calculate a fitness array for all individuals in assembly *Pop* and *Offspring*, then construct the new population *Pop* for the next iteration by making roulette wheel selection *Pop_size-1* times according to fitness array.

Step 8: (Stop test) If the reproduction times equal to *Max_Itr_reproduce* or the convergence degree of the present population is larger than 95%, stop this procedure and let the best performing one as our solution; otherwise go back to Step 3.

2.2 Detail of the memetic procedure's sub-procedures and rules

1) Coding and Initializing Population

We code each scheduling solution as an N+M-1 dimensions integer array and call it an individual or a chromosome. Its first part, consisting of N integers, denotes the possible operating order of surgical operations and each member represents a certain surgical operations's serial; The second part, composed of M-1 integers, represents the partition symbols. In the following, we will try to explain this coding strategy by considering a simple example assigning 9 surgical operations into 3 operationg rooms, whose chromosome structure is showed in figure 1.

P	2	5	8	1	3	4	6	7	9	4	6

Figure 1: coding of a certain surgical operations' assignment

According to this coding, the first member of the second part is 4, that means the first four surgical operations 2,5,8,1 are scheduled to the first operating room in order $\{2\rightarrow5\rightarrow8\rightarrow1\}$; the surgical operations between position 4 and 6 of the first part, i.e., surgical operations 3 and 4 are assigned in order $\{3\rightarrow4\}$to the second operating room; Finally, the rest three surgical operations are assigned to the last operating room in order $\{6\rightarrow7\rightarrow9\}$.

2) Heuristic For Local Amelioration.

This sub-procedure is to try to ameliorate the performance for each operating room when it has once been assigned a set of surgical operations. Considering the results given by Morton et al. 1993[4], a COVERT heuristic procedure is used to improve an individual S who means a schedule of assigning $N_{k,s}$ surgical operations into operating room OR_k , respectively.

Procedure COVERT:

Step 1: Set $G=\{1,\dots,N_{k,s}\}$, t_0=0;

Step 2: For each surgical operation *j* of un-scheduled surgical operations' set *G*, calculate its parameter $COVERT_j = (w_j/p_j)*Max\{0,1-Max(0,d_j-t_0-p_j)/(k*p_j)\}$, where k=2.

Step 3: chose the surgical operation j whose $COVERT_j$ is the largest, and schedule it for processing next starting at time t_0.

Step 4: Let $t_0=C_j$, $G=G\backslash\{j\}$. If there are still members in G, go to step 2.

3) Crossover

Here, a well-known crossover procedure named order crossover (OX), proposed by Goldberg, 1989[5], is applied to crossbreed two parents *P1* and *P2* for a progeniture *O*:

Step 0: (Propagate the partition structure) Copy the last M-1 positions' value(the second part of chromosome) from one selected parent, such as *P1*, into the offspring *O*;

Step 1: (get genes from one parents) Copy a randomly selected sub-part from the first N integers(the first part of chromosome) of the same parent *P1* in to offspring *O*;

Step 2: (complete the rest genes by the other parent) Copy the remaining genes from the first part of the other parent *P2* by making a left-to-right scan.

4) Mutation

Considering each individual (chromosome) is composed of two parts, we propose two mutation rules for these two parts respectively:

a) (Swapping mutation rule for the first part representing interventions' operating order):
   Firstly, select two random positions among them ; then, swap these selected two genes.
b) (Mutation rule for the second part—the part of partition symbols).:
   Motivated by Liu Min et al.,1999[6], our rule is as follows: Firstly, randomly generate an integer k from [$0,N$] and an integer j from [$N+1, N+M-1$], respectively; then replace the value of *jth* position in the selected individual(chromosome) with the generated k if k doesn't equal to that value, otherwise, re-generate this integer k until they are different.

5) Fitness for Evaluation and Reproduction

A $|SS|$-dimension fitness array is obtained for the current individuals' assembly SS, containing $|SS|$ individuals, by means of calculating its relative advantage of objective function. The detail is as follows:

Step 0:Let $F(s)=\sqrt{f_w-f_s}$ where $f_s=\sum_{j=1}^{N} w_j T_j, s\in SS$, and $f_w=\max\{f_s \mid s\in SS\}$.

Step 1: Calculate each individual's distribution probability:

$$P(s)=\frac{F(s)}{\sum_{k=1}^{|SS|} F(k)}, s\in SS$$

Step 2: (roulette wheel selection):

Step 2.1: Calculate $RW=\left\{rw_s \mid rw_0=0, rw_s=\sum_{k=1}^{s} p_k, s\in SS\right\}$

Step 2.2: Generate randomly *Sel_Num* real numbers λ_k, $k=1,\ldots,Sel\_Num$, that are uniformly distributed between 0 and 1. (Note: for the crossover selection, *Sel_num*=2; for the new population selection *Sel_num=Pop_size*).

Step 3.3: If $rw_{s-1} < \lambda_k < rw_s$, individual *s* is selected as the crossover parents or an individual of new population).

3 Numerical experiment

For not having found any benchmark problem in the literatures, we make a comparison between our proposed heuristic-based memetic algorithm (MA) and its corresponding general genetic algorithm (GA) on a set of randomly generated problems by using the same CPU time.

3.1 Data

The numbers of surgical operations are [N: 10,20,30, 40, 50, 60]. The numbers of operating rooms are [M: 2,4,6]. The surgical operation duration p_j are integers and are generated from uniform distribution [1,20]. Their weights w_j are also integers and are generated from uniform distribution [1,10]. As for the due date, i.e. the operating rooms' closing time, is assumed to be a same integer for all surgical operations according to a real parameter β: $d = \overline{P} + \beta \sum_{j=1}^{n} p_j$, where β=0.25 for N<=25 and β=0.05 for N>25, $\overline{P}$ is all the interventions' average processing time.

3.2 Lower Bound

A lower bound is developed according to Azizoglu et al. 1998[7] and Yalaoui et al.2002[8].Azizoglu et al., 1998[7] calculated a lower bound for a parallel machines tardiness minimization problem by allowing job splitting. Yalaoui et al. 2002[8] proposed another method called multiple shortest processing time lists (MSPTL) for the same kind of problem. Without loss of generality, we assume that jobs are indexed in the SPT order, namely, $p_1 \leq p_2 \leq \ldots \leq p_N$. Then, developed two lower bounds according to formulas(1) and (2), respectively and always accept the bigger one.

LB_M: This lower bound is equivalent to allowing job splitting and it is developed from the lower bound method of Azizoglu et al., 1998[7] according to theorems 1 and 2:

$$\sum_{i \notin S} w_{[i]} * Max\{0, \{\sum_{j=1}^{i} \{Min_k\{PT(k)\} + \frac{p_j}{m}\} - d\} \qquad (1)$$

Where $w_{[i]}$ is the *ith* smallest weight among all the un-sequenced jobs and *d* is their common due date.

LB_F: The following lower bound method is developed from the lower bound method of Yalaoui et al. 2002[8] thanks to theorem 3:

For the *kth* (*k=1, 2, ..., M*) single machine schedule, jobs for *k* to *N* are scheduled in increasing order of their indexes. Let $C_{k,j}$ be the jth smallest completion time in the *kth* single machine schedule, $w_{k,j}$ be its corresponding weight, $C\_W_{k,j}$ be a structured element, i.e., $C\_W_{k,j}=(C,w)_{k,j}$ where $C\_W_{k,j}.C=C_{k,j}$, $C\_W_{k,j}.w=w_{k,j}$. Let *G* be the set of these structured elements and rank all of them according to the completion times' increasing order. Let $C\_W_{MWSPT,j}$ be the jth element of the ranked set, then we can obtain a set

$G'=\{C\_W_{MWSPT,j}=(C,w)_{MWSPT,j} | j=1,\ldots,N\}$. Thus, a lower bound can be calculated as:

$$\sum_{i=1}^{N} C\_W_{MWSPTL,[j]}.w * \max(C\_W_{MWSPTL,j}.C-d,0) \quad (2)$$

Where $C\_W_{MWSPTL,[j]}.w$ is the *jth* smallest weight of set G' and d is the common due date.

Theorem 1: Let J_i and J_k be two common-due-date jobs such that $p_i \leq p_k$, $w_i \geq w_k$, and their due date is d. Then, there exists and optimal schedule in which J_k starts after the end of J_i.

Theorem 2: Let J_i and J_k be two jobs such that $p_i \leq p_k$, $w_i \leq w_k$ and they have a common due date d , Exchanging w_i and w_k does not increase the optimal total weighted tardiness.

Theorem 3: Let two series of numbers $(x_1,x_2,\ldots,x_N)$, $(z_1,z_2,\ldots,z_N)$ and an ordered series $(x_1',x_2',\ldots,x_N')$ be so that 1) $x_1' \leq x_2' \leq \ldots \leq x_N'$, 2) for $j=1,\ldots,N$, $x_j' \leq x_j$, 3) $z_j \geq 0, j=1,\ldots N$. If $(z_1',z_2',\ldots,z_N')$ is the series obtained by sorting the series $(z_1,z_2,\ldots z_N)$ in non decreasing order and $y>0$, the following relation holds: $\sum_{j=1}^{N} z_j' \max(x_j'-y,0) \leq \sum_{j=1}^{N} z_j \max(x_j-y,0)$

3.3 Results

We have tested 20 times for all the 18 cases. In 333 tested problems (92.5%), the memetic algorithm (MA) performs better than the general genetic algorithm (GA).

Reference

[1] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker (1977). Complexity of Machine Scheduling Problems, *Annals of Discrete Mathematics,* Vol. 1,pp 343--362.

[2] Matthijs den Besten, Thomas Stützle, Marco Dorigo,(2000). Ant Colony Optimization for the Total Weighted Tardiness Problem, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*.

[3] Paulo Franca, Alexandre Mendes, Pablo Moscato, (2001). Theory and Methodology: A Memetic Algorithm for the Total Tardiness Single Machine Scheduling Problem, *European Journal of Operational Research*, Vol. 132,pp 224-242;

[4] T.E.Morton, D.W. Pentico, (1993). Heuristic Scheduling Systems with Application to Production Systems and Project Management, *Wiley, New York*.

[5] D.E. Goldberg, (1989). In: Genetic Algorithms in Search, Optimization and Machine Learning, *Addison-Wesley, Reading, MA*.

[6] Liu Min, Wu Cheng,(1999). A Genetic Algorithm for Minimizing the Makespan in the Case of Scheduling Identical parallel machines, *Artificial Intelligence in Engineering*, Vol.13,pp 399-403;

[7] Meral Azizoglu, Omer Kirca,(1998), Tardiness Minimization on Parallel machines, *International Journal of Production Economics*, Vol. 55, 163-168.

[8] Farouk Yalaoui, Chengbin Chu,(2002). Parallel machine scheduling to minimize total tardiness, *International Journal of Production Econimics*, Vol. 76,pp 265-279.