Solving Scheduling Problems via Evolutionary Methods for Rule Sequence Optimization

Igor P. Norenkov[1], Erik D. Goodman[2]

[1]Moscow State Bauman Technical University, Russia, norenkov@aicad.isrir.msk.su
[2]GA Res. & Applic. Gp., 230 Engineering, Michigan State Univ., East Lansing, MI 48824, goodman@egr.msu.edu

Keywords: JSSP, Flow Shop Scheduling, Genetic Algorithms, Heuristic Combination Method, Hybrid

Abstract

The paper is devoted to solution of multistage scheduling problems by genetic algorithms. The Heuristics Combination Method (HCM) is described. The idea of HCM is to optimize the choice and sequence of application of a set of heuristic rules for schedule synthesis, from among a given initial set. Several approaches to improvement of the quality of the schedules synthesized are discussed. One of the approaches is a hybrid evolutionary-genetic method. Some experimental numerical results are given.

1. Introduction

Scheduling problems are found in many areas. As a rule, these problems are NP-hard tasks. Hence, practical problems which are sufficiently large are typically solved only approximately, using various heuristic rules. Unfortunately the degree of optimality of the solutions thus obtained is often not very good, and this quality can not be predicted *a priori*. Evolutionary methods (including genetic algorithms) comprise an important group of approximate methods. There are many publications devoted to genetic algorithm (GA) applications to scheduling problems -- see, for example, review work [1]. Some papers include information about GA applications to scheduling in aircraft design [2], resource distribution [3], manufacturing processes [4], frequency assignment in radio channels of networks [5], *etc.*

One of the main tasks in scheduling by GA is the formulation of the chromosome structure, i.e., the representation of the data about work distribution in time and between servers to be used by the GA. It is possible to utilize representations and operators which require “repair” of the genome after crossover operations, for example, to guarantee that the offspring represents a feasible schedule. However, such repair operations may break some useful building blocks and decrease scheduling efficiency. An interesting alternative is to use an implicit form of schedule representation on the chromosome, in which the gene at the i-th locus does not represent the number of a job or server, but instead the number of a heuristic to be used as the i-th step of generating the schedule. This is the defining feature for the Heuristics Combination Method - HCM [7] described here. The idea embodied in HCM was published simultaneously and independently in [7] and by Fang [8], both in 1994.

This paper describes evolutionary and genetic algorithms created on the basis of the HCM and examples of their applications to solving multistage Flow-Shop Scheduling Problems (FSSP). Section 1 presents the typical multistage JSSP. The HCM is described in Section 2. Section 3 is devoted to algorithmic questions about HCM realization. The experimental numerical results and efficiency evaluation of HCM are given in Section 4.

2. Multistage Scheduling Problems

The HCM may be applied to a wide class of discrete optimization and structure synthesis problems. The features of this class are the following:

1. The synthesis is a multistep process.
2. A fitness value may be computed only when the total schedule is compiled.
3. There are some heuristics that can be used at every step of synthesis.

A typical problem for solving by HCM is the following multistage Flow Shop Scheduling Problem [9]:

- there are N jobs; every job A_i is treated in q stages of service successively from the first stage ($k=1$) to the last stage ($k=q$), $i = 1..N$;
- there are M_k machines (servers) at the k-th stage, k = 1..q, the total number of servers is

$$M = \sum_{k=1}^{q} M_k$$

- only one job at a time may be served on a server; once started, the service cannot be interrupted;
- every job belongs to one family, and the j-th server has to undergo a setup change (with associated family-specific setup time) if two consecutive jobs assigned to the j-th server belong to different families;
- matrix P of service times is given; element P_{ij} is the time required to serve the i-th job on the j-th server;
- the matrices E_j of setup times for each server are given; element E_{jir} is the setup time when the i-th job precedes the r-th job on the j-th server;
- costs per unit service and setup time, C_i and R_j, respectively, are known;
- each job has both soft and hard due dates (completion times), D_i and T_i, respectively, and $T_i > D_i$;
- penalties $G1$ and $G2$ are assigned for violation of soft and hard due dates, respectively.

The problem is to find the minimum cost schedule (cost is the anti-fitness function to be minimized), including costs of processing times, setup times, and penalties for violating soft and hard due dates. The most complex problem used in our experiments has been described in [9]. Its initial data are N=105, q=4, M=15. This problem is called N105 in the discussion below.

3. Heuristics Combination Method

A schedule synthesis consists of two parts. The first is choice of order of consideration of jobs. The second part is assignment of jobs to servers. Each part may be determined by a variety of heuristic and/or genetic methods.

Here are some rules for job ordering sometimes used in the first part of scheduling. Rule A1 requires the ordering of jobs in accordance with increasing parameter $GAP = X_i$, where X_i is the time when i-th job service ends on the previous stage. Parameter $GAP = D_i$ is rule A2, and $GAP = (a_1*D_i+a_2)*(a_3*D_i+a_4)$ is rule A3, where $a_1..a_4$ are real coefficients. The set of rules is extended if we take into account only jobs in the interval $[t, t+T_{tr}]$, where t is current time, i.e., minimal X_i, for i *in* I, and I is the set of numbers of free (not yet assigned) jobs; T_{tr} is an additional parameter changed from one rule to another.

The rules for job assignment to servers are as follows. Rule B1 chooses the server providing the cheapest service. Rule B2 chooses the server providing the quickest service of the job. The set of assignment rules may be extended if we take into account preferences for successive service of same-family jobs.

The composition of rules from the first and second parts generates the set of heuristics. The solution of practical scheduling problem has shown that using single concrete heuristics does not lead to acceptable results. On the other hand, application of different heuristics at different steps of schedule synthesis generates a set of schedules containing significantly better variants. Hence, the task is search for the optimal sequence of application of heuristics.

The HCM is a method for addressing this optimization problem. The chromosome in the HCM includes V = N*q genes. The allele at the i-th locus is the name (i.e., number) of the heuristic to be applied at the i-th step of schedule synthesis. Therefore, the value at every locus need not be unique, and “repair” of offspring chromosomes is not required. The other advantage is that the simple substitution of a set of heuristics leads to adaptation of the GA to a new class of synthesis problems.

A chromosome in the multistage JSSP may be represented as a matrix CR of size $q*N$, where element CR_{ik} refers to the k-th step of synthesis at the i-th stage. Such a representation shows that two directions (horizontal and vertical) of crossover are suitable. The two directions may be explained by epistasis. Epistasis means that the fitness function depends on not only independently on the alleles at the i-th and k-th loci, but also on their joint presence on the chromosome. Alleles at loci with nearby values of index i and index k form building blocks. Disruption of a building block worsens convergence to optimal solution. The natural one-point crossover is good for schemata in one row of CR (they correspond to one stage of the schedule), but it is not effective with respect to schemata formed by alleles in loci with different values of index i under the same value of index k (they correspond to one job at different stages). Such crossover might be termed *horizontal*, and schemata including the alleles in loci CR_{ik} and $CR_{i,k+1}$ are horizontal building blocks.

Schemata including alleles in loci CR_{ik} and $CR_{i+1,k}$ are the vertical building blocks. The better reproduction of vertical building blocks requires a vertical crossover, which is an f-point one, where $f=q-1$, and the distance between adjacent crossover points is N. It appears desirable to use both horizontal and vertical crossovers. Coefficient $A = Q_v/Q_h$, where Q_v and Q_h are probabilities of vertical and horizontal crossover, respectively, is one of the parameters for control of the solution process.

4. Hybrid Scheduling Algorithms

The trajectory of a GA toward optimal solution can be characterized by the speed S_f of improvement of the fitness function (FF) during the initial generations and by the level L of stagnation or convergence when FF is almost unchanging. There are some applications in dynamic control when the number of FF evaluations cannot be large and parameter S_f is very important. On the other hand, L is the main index of GA effectiveness in most cases. Therefore it is important to find ways for improvement of L.

It is evident that the quality of the heuristics influences L. The degree of this influence is defined by experiment (see the next section). The proper choice of control parameters such as A is the other natural way. A further way of improvement of solution accuracy is application of a multipopulation (coarse-grain parallel) GA [9] and/or hybrid algorithms. Our multipopulation algorithms were realized using the GALOPPS [9] software.

The hybrid algorithms [6] are a combination of GA and evolutionary algorithms (EA). Although both the GA and EA are evolutionary, EA as used here will mean that a single chromosome or a population of chromosomes is used without crossover. The principal procedures used for our hybrid algorithms (HA) are described as follows:

Procedure SH1 is evolutionary stochastic hillclimbing:

```
FF = fitfun(CC);  /*fitfun is procedure of FF evaluation, CC is  chromosome*/
i  =  0;
while (i<K)  /*K  is equal to (0.2..0.5)*V,  where V is the number of genes*/
 {i++;
     Choose at random t loci and new random alleles at them; /* t = 3..8 */
     CC1 = current chromosome;
     z = fitfun(CC1);
     if (z < FF) {CC = CC1; FF = z; i=0;}
 }
```

Procedure SH2 is similar, but with choice of t adjacent loci, with random choice of first position.

Procedure SH3 is similar to SH2, but distance between chosen loci is equal to N positions and $t = (1...2)*q$. In other words, the chosen loci correspond to one or two columns of the matrix CR.

Procedure SH4 is like SH1, but simple hillclimbing is used in SH4, with mutations of all genes in turn.

Two alternative procedures may be used for generation of the initial population. In the first procedure, IP1, the population is formed from random chromosomes under the condition

$$FF < TS, \qquad (1)$$

where TS is a threshold value. The second procedure IP2 includes SH1 additionally, where SH1 is applied to chromosomes chosen under condition (1).

The crossover may be horizontal, vertical or mixed with ratio A in the interval [0, 1]. The members of the new generation are selected under the condition

$$FF < FF_{min}+\delta,$$

where FF_{min} is the minimal value of FF at the current time, and δ is a threshold value which depends on the solution method. Every offspring chosen by condition (2) is treated in one of procedures SH1..SH4.

5. Experimental Results

The aim of the experiments was to explore the influence of various factors on the efficiency of solving of several flow shop scheduling problems. We used some test tasks, including the complex task N105 and tests N21A, N21B and N21C, which have different values of matrices P for an example with N=21. P is set to P_a for N21A, P_b=1.3*P_a for N21B and P_c=1.4*P_a for N21C. Increasing P (service times) means that the tasks are harder to complete before their soft or hard due dates, which means that the role of the tardiness penalties in the fitness function is increased.

The influence of the set of heuristics chosen from is very large. Table 1 includes the values of the FF in test tasks in which only one heuristic from the set S1...S8 is used. The values of FF_a (the fitness function without taking into account the penalties) are shown in parentheses. For example, no single heuristic is able to satisfy the hard due dates in task N21C.

Problem/ Heuristic	N21A	N21A (no penalties)	N21B	N21B (no penalties)	N21C	N21C (no penalties)	N105	N105 (no penalties)
S1	5629	5529	14376	7196	13841	7691	24583	22417
S2	6702	5602	13376	7226	13961	7801	31391	22822
S3	6635	5565	11406	7216	14927	7737	23851	22470
S4	7694	5604	13391	7271	14941	7781	39423	22759
S5	6699	5579	9484	7304	13008	7818	23084	22879
S6	5696	5636	7413	7253	8928	7758	23283	23113
S7	8720	5590	13318	7148	17935	7765	69813	21979
S8	6521	5381	16093	6913	18583	7413	36602	21863
All	5474	5405	7205	7055	8793	7613	22520	
Pick 3	5451	5351	7165	6995	10728	7568	22059	

Table 1: Performance on four tasks using each heuristic alone and using either 3 best or all of them at random.

The row "All" shows the results of the HCM when all heuristics S1...S8 were used with equal probabilities.

The data from the Table 1 show that no heuristic applied separately leads to satisfactory results. Clearly, some method for selecting and sequencing the heuristics is required to achieve reasonable performance. We might try ordering the heuristics in accordance with the above values of FF or FF_a and by choice of some of the better of them. Row "Pick 3" shows the results when the 3 best heuristics for each problem are used, as determined according to FF_a. They were S8,S1,S3 for N21A and N21C, and S8,S7,S1 for N21B and N105. The hybrid algorithm (HA) with procedure SH2 was used with A = 0.5, t = 6, K= 20, the size of population N_p = 25 (for N21) or N_p = 13 (for N105). Computations were terminated after 10 generations. The last row of Table 1 shows that ordering by FF_a may give good results if due date constraints are not very difficult to satisfy (service times are short relative to due dates specified). In the other case, the better heuristics from the list ordered by FF must be added to the subset of heuristics to be applied. This recommendation was used for N21C. Adding rule S6 led to FF=7703 instead 10728.

The second significant factor that influences solution efficiency is the type of algorithm used. The comparison of a single-population algorithm (SPA) with a hybrid evolutionary-genetic algorithm (HA) was performed for task N21C. The values of FF received after 10,000 evaluations are shown in Table 2. The advantages of coarse-grain parallel genetic algorithms (cgPGA) in comparison with the SPA (single population GA) were described in [10]. Our experiments showed that HA performance was not inferior to cgPGA for this problem. Table 3 presents a sampling of the values of FF obtained in runs on problem N105 by the parallel GA program GALOPPS. Eight parallel subpopulations were used, each with population size N_p. Migration of some members between populations in a ring was performed every 5 generations. Columns show the results after n generations.

N_p	25	37	75	100	115
SPA	8666	7755	7756	7771	7725
HA with δ=4	7747	7707	7711	7730	----
HA with δ=10	7740	7712	7699	----	----

Table 2: Results with various algorithms and population sizes.

Run #	N_p	Gen 4	Gen 8	Gen 12	Gen 16	Gen 24	Gen 48
3	27	22182	22162	22134	22100	22064	-----
4	31	22161	22090	22070	22054	22031	22021
7	39	22181	22132	22055	22025	22013	-----
10	49	22140	22098	22089	22076	22076	-----
12	60	22179	22141	22102	22077	22059	-----

Table 3: Results of 8-subpopulation parallel GA runs on N105, various subpopulation sizes.

Problem	N105	N105
N_p	13	13
Heuristic Proportions	S1:S7:S8 = 4:3:1	S1:S7:S8 = 3:2:3
Population Initialization	IP2, K=40	IP2, K=60
Algorithm	SH1, K=120	SH2, K=60
Parameters	t=3; A=0	t=6; A=0.5
Range of Fitnesses	21956-22053	21887-22058
Average Final Fitness	22005	21980

Table 4: Fitnesses and fitness ranges, parameter settings for SH1, SH2 runs of hybrid algorithm (HA)

The results of the hybrid algorithm (HA) on task N105 after about 30,000 fitness evaluations are presented in Table 4. Here, the notation S1:S7:S8 = 4:3:1 means that a choice of heuristic S1 is made with probability 0.5, heuristic S7 with 0.375 and heuristic S8 with 0.125. Epistasis may be taken into account by selecting one procedure from SH1 (or SH4), SH2, SH3 and by the choice of coefficient A (ratio of vertical to horizontal crossover). Table 5 shows the fitnesses in problem N105 for various values of coefficient A (after 10 generations, N_p=13, K=20).

The table indicates a preference for algorithm SH2, although this conclusion is not strongly supported. The additional reason favoring SH2 is its quicker performance on test N21.

A	0.0	0.2	0.4	0.6	0.8	1.0	Average
SH1	22167	22222	22170	22103	22211	22166	22173
SH2	22150	22102	22129	22121	22123	22142	22128
SH3	22201	22124	22114	22176	22134	22148	22150
Average	22173	22149	22138	22133	22156	22152	-----

Table 5: Fitnesses versus algorithm, coefficient A, on N105.

6. Conclusion

An approach based on evolutionary principles of selection of optimal combinations of heuristics is a promising way to seek near-optimal solutions to complex scheduling problems. It may readily be realized using evolutionary-genetic algorithms. The quality of the solutions obtained depends on the subset of heuristics applied. It is advisable to select the set of eligible heuristics based on the peculiarities of the particular problems to be addressed.

Acknowledgements

The authors thank the members of the MSU Manufacturing Research Consortium, and particularly Dr. James Maley of Ford Motor Company, for suggesting this problem and partially supporting the initial phases of this work.

References

[1] Bruns, R., 1993, Direct chromosome representation and advanced genetic operators for production scheduling, *Proc. of fifth int. conf. on GA*, pp. 352-359.

[2] Bramlette, M., Bouchard, E., 1991, Genetic algorithms in parametric design of aircraft, in *Handbook of genetic algorithms*, L. Davis, ed., Van Nostrand Reinhold, New York, pp. 109-123.

[3] Syswerda, G., Palmucci J., 1991, The application of genetic algorithms to resource scheduling, *Proc. of fourth int. conf. on GA*, pp. 502-508.

[4] Yasinovsky, S.,1996, Genetic algorithms based hybrid system for job-shop scheduling, *Proc. EvCA'96*, Moscow, Russ. Acad. Sci., pp. 316-320.

[5] Kapsalis, A., Chardaire, P., Rayward-Smith, V., Smith, G., 1995, The radio-link frequency assignment problem: a case study using GA, *AISB Workshop on Evolutionary Computing*, Springer, pp. 117-131.

[6] Goldberg, D.,1989, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, New York.

[7] Norenkov, I. P., 1994, Scheduling and allocation for simulation and synthesis of CAD system hardware, *Proc. EWITD 94, East-West internat. conf.*, Moscow, ICSTI, pp. 20-24.

[8] Fang, H., Ross, P., Corne, D., 1994, A promising hybrid GA/heuristic approach for open-shop scheduling problems, *ECAI 94, 11th European Conf. on AI*, John Wiley & Sons, New York.

[9] Tcheprasov, V., Punch, W., Goodman, E., Ragatz G., Norenkov, I., 1996, A genetic algorithm to generate a pro-active scheduler for printed circuit board assembly, in *Proc. EvCA'96*, Russ. Acad. Sci., Moscow, pp. 232-244.

[10] Goodman, E., Punch, W., 1995, New techniques to improve coarse-grain parallel GA performance, *Proc. CAD'95*, Yalta, pp. 7-15.