A GENETIC LOCAL SEARCH ALGORITHM FOR SEQUENCING PROBLEM ON A SYNCHRONOUS FLOW LINE

Banu Soylu
Orta Doğu Teknik Üniversitesi, Endüstri Mühendisliği Bölümü, 06531, Ankara

Abstract: Genetic Algorithms (GAs) have been applied on a variety of combinatorial optimization problems with high success. GAs have also become increasingly popular as a means of solving flowshop sequencing problems. The synchronous line sequencing (SLS) problem is one of sequencing jobs in a synchronous flow line with the objective of minimizing makespan. In this study, I apply two genetic local search (GLS) algorithms, which is a hybrid algorithm of a local search (LS) and a genetic algorithm (GA), to the problem environment. I also use a simple construction heuristic to obtain one of the members of the initial population, the others are generated randomly. I compare the performance of GLS algorithms with the optimum results which was obtained from a branch and bound algorithm with tight upper and lower bounding procedures.

Keywords: *Genetic Algorithms, Local Search, Flowshop*

1. Introduction

GLS is a hybrid algorithm where a local search (LS) procedure is incorporated into a GA. In this study, I apply two modified GLS algorithms to the SLS problem. I also modify the LS procedure where all the neighborhood solutions of a solution are not searched. I determine the number of neighbors to be searched according to job size. When the job size increases, the number of searched neighbors increases. Therefore, it is dynamically updated.

Exact algorithms can hardly be designed to solve large sequencing problems which is generally NP-hard. The solution times of this algorithms is very high for the large-size problems. Therefore, the right way to proceed is with heuristic techniques. The synchronous line sequencing problem is NP-hard. The problem is equivalent to classical flowshop sequencing problem with makespan minimization and synchronous transfers. We described a branch & bound algorithm together with several bounding procedures in our previous study (Soylu et. al., 2002). However, the performance of algorithm was low for large size problems in a reasonable time.

A large number of papers on various flowshop models have appeared in the literature. Many of these studies consider makespan minimization. Johnson (1954) presented an optimizing rule for the two machines makespan problem. Garey , Johnson and Sethi (1976) gave the NP-hardness proof for the three-machine cases.

GA and GLS algorithms have been applied flowshop sequencing problems with high success. A GA for flowshop scheduling is proposed by Reeves (1995). Murata et al. (1994) presents a genetic local search algorithm for flowshop scheduling problems. Murata et al. (1996) compare various crossover and mutation operators and show that the two point order crossover and shift change mutation operators are effective for this problem. They consider two hybrid variants of the GA, namely the genetic local search and genetic simulated annealing algorithms and show high performance for the hybrid models.

In this study, I focus on the implementation of the GLS into the SLS problem by using the structure of the problem. The main objective is to investigate the performance of algorithms and the effect of various genetic operators under the framework of SLS. To the best of our knowledge, this study reports the first results that examines the performance of different GLS algorithms over the optimal solution of benchmark SLS problems which is obtained from a B&B algorithm with tight upper and lower bounds.

2. Problem Definition

In a synchronous line, speed of the transfer mechanism is adjusted by the end of each unit completion; i.e. cycle. The aim of the SLS problem is to sequence the jobs on a synchronous line and its objective is to minimize the total cycle time, i.e. makespan. Each station is given exactly the same amount of time to operate on each job. This time is called cycle time, i.e. the maximum processing time of the jobs that are on the line simultaneously. We let C_t denote the time spent in the t^{th} cycle. The total cycle time, i.e. makespan is then $\sum_t C_t$. Figure 1 illustrates the concept of cycle times in a synchronous line having 3 stations and 3 jobs.

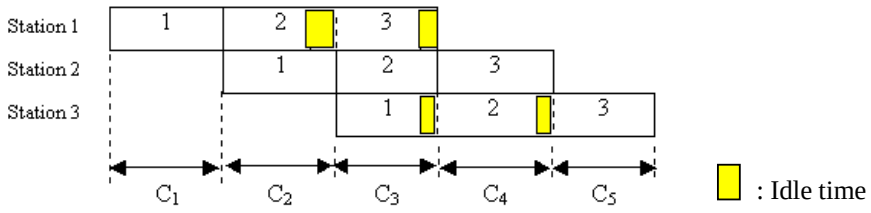

Figure 1. Illustration of cycle times in a mixed model synchronous line

During the first cycle, only the first station is busy. By the end of the first cycle, the job processed on the first station is transferred to the second station and a new job is loaded to the first one. By the end of the second cycle, jobs 1 pass to the third station and job 2 pass to the second station. The cycle times are computed as follows:

$C_1 = p_{11}$, $C_2 = \max\{p_{21}, p_{12}\}$, $C_3 = \max\{p_{31}, p_{22}, p_{13}\}$, $C_4 = \max\{p_{32}, p_{23}\}$, $C_5 = p_{33}$.

Pij is the processing time of job i on station j.

Note that makespan is $\sum_{t=1}^{5} C_t$

We described a branch & bound algorithm together with several bounding procedures in our previous study (Soylu et al., 2002). But the performance of algorithm was low for large size problems in a reasonable time. Therefore, heuristics are suitable for large problem instances.

3. SLS Heuristic for Initial Population

The algorithm tries to assign the jobs having largest processing times of different stations to the same cycle. By this, we aim to reduce the makespan by minimizing idle time of the stations at a cycle. We obtain different sequences for each station and calculate the makespan value corresponding to each schedule. At the end, I use the schedule with the minimum makespan value.

4. Steps of the GLS Algorithms

Murata et al. (1994) presents a genetic local search algorithm for flowshop scheduling problems. Their GLS is similar to the following GLS algorithms but the some operators and the manipulation of some steps are different from it because of the structure of the problem. For instance, to ensure the tradeoff between extrapolaration and exploitation, I generate one of the elements in the initial population by using a heuristic algorithm which is proven to work well for the SLS problem (Soylu et al., 2002). I also use different selection strategies and different neighborhood size in the local search procedure. They also obtain just one offspring from each pair of parents in crossover but I obtain two offsprings from each pair of parents. According to these modifications, steps of the GLS algorithms and Local Search procedure are proposed as follows.

4.1. GLS Algorithm 1

Step 0 (Initialization): Let t be the generation index. Randomly generate an initial population including POPSIZE-1 solutions. Generate the other one element of the initial population by using SLS heuristic.
Step 1 (Evaluation): Calculate the fitness of each solution x in the current population.
Step 2 (Selection): Select POPSIZE-1 solutions from the current population according to the roulette-wheel selection operator.
Step 3 (Crossover): Randomly select two mates and apply two-point order crossover with the crossover probability P_c to generate two offsprings. When the crossover is not applied, parents are handled as offsprings. Repeat this until obtaining POPSIZE-1 members.
Step 4 (Mutation): Apply shift change mutation operator to each of the generated POPSIZE-1 solutions in Step 3 with the mutation probability P_m.
Step 5 (Local Search): Apply a local search procedure to each of the POPSIZE-1 solutions generated by the genetic operators.
Step 6 (Elitist Strategy): Add the best solution in the previous population to the current population.
Step 7 (Termination): If a prespecified stopping condition is satisfied, stop the algorithm. Otherwise update t as t = t + 1 and return to Step 1.

4.2. Local Search Procedure

Step 5 (Local Search): Apply a local search procedure to each of the POPSIZE solutions generated by the genetic operators.

Step 5-0: Specify an initial solution x. Each solution of the current population is used as an initial solution x in the GLS algorithms.

Step 5-1: Examine a neighborhood solution y of the current solution x.

Step 5-2 : If y is a better solution than the current solution x, replace the current solution x, with y and return to Step 5-1.

Step 5-3: If a certain number of neighborhood solutions (say, k solutions) of the current solution x have been already examined then end the local search procedure for the current solution x. Otherwise return to Step 5-1.

5. Experimental Results

I have experimented with the GLS algorithms in order to find the best parameter settings and to compare solution quality of the algorithms with the optimal solution. Experiments are performed for SLS problem of different sizes. Optimal solution of the SLS problem is found by using the branch & bound algorithm with tight upper and lower bounds.

I also compare the performance of two GLS algorithms according to % deviation from optimal value. Column 5 of the table 1 shows the average % deviation from optimal for four problem types. When we examine this table, we can easily observe that GLS algorithm 1 outperforms GLS algorithm 2. As the problem size increases, solution quality of the algorithms decreases, but this decrement is not so drastic. Average percent deviation of the GLS algorithm 1 from the optimal solution varies between 0.2% - 0.3% for small problems, 0.5% - 0.6% for medium ones. When we look at the CPU times of the algorithms, we see that it changes between 10 and 20 seconds which is very low.

Table 1. Comparison of GLS Algorithm 1 and GLS Algorithm 2.

Problem Size K x m	# of Gen.	Pc	Pm	Average % Deviation From Optimal {(GLS1 - OPT.)/OPT.}*100	Frequency for Getting Best One	CPU Time (seconds) GLS1		CPU Time (seconds) B&B	
						Average	Max.	Average	Max.
15x3	2000	0,8	0,1	0,6	5 of 10 instances	10,5	11	330	1217
20x3	2000	0,8	0,1	0,5	2 of 5 instances	17,6	18	4096	10147
25x2	2000	0,8	0,1	0,3	5 of 8 instances	12,7	15	98	774
30x2	2000	0,8	0,1	0,2	6 of 8 instances	19,2	20	63	343

Problem Size K x m	# of Gen.	Pc	Pm	Average % Deviation From Optimal {(GLS2- OPT.)/OPT.}*100	Frequency for Getting Best One	CPU Time (seconds) GLS2		CPU Time (seconds) B&B	
						Average	Max.	Average	Max.
15x3	500	0,8	0,1	1,4	1 of 10 instances	2,8	3	330	1217
20x3	500	0,8	0,1	2,6	0 of 5 instances	4,5	5	4096	10147
25x2	500	0,8	0,1	0,4	4 of 8 instances	3,5	4	98	774
30x2	500	0,8	0,1	0,6	2 of 8 instances *	4,8	5	63	343

* Note that B&B algorithm can solve 8 of 10 instances in a reasonable time (maximum cpu time is set to 24000 sec. for each instance)

6. Conclusion

In this study, I consider a flowshop sequencing problem on a synchronous line (SLS) with the objective of minimizing makespan. I develop an evolutionary approach for solving the SLS problem. I construct two GLS algorithms which uses genetic operators that performs well on flowshop sequencing problems. Basic structure of GLS algorithms is similar to the proposed GLS algorithm by Murata et al. (1994), but some modifications are done to improve their algorithm.

The major contribution of this study is development of a hybrid metaheuristic algorithm which performs well for the large size SLS problems. To the best of our knowledge, this is the first attempt to obtain a good solution to the SLS problem by a hybrid algorithm GLS.

References

H. Ishibuchi, T. Murata, S. Tomioka, (1997), "Effectiveness of genetic local search algorithms", Proc. of the seventh ICGA, Morgan Kaufmann Publishers, USA.

T. Murata and H. Ishibuchi, (1994), "Performance evaluation of genetic algorithms for flowshop scheduling problems", Proc. first ICEC (Orlando, USA, June 27-29, 1994), pp. 812-817.

T. Murata et al., (1996), "Genetic algorithm for flowshop scheduling problems", Computers and Industrial Engineering. Vol. 30, No. 4, pp. 1061-1071.

C. R. Reeves, (1993), Modern Heuristic Techniques for Combinatorial Problems, John Wiley&Sons, INC., New York, NY, United States of America.

B. Soylu, Ö. Kırca, M. Azizoğlu, (2002), "A mixed model line sequencing problem with makespan minimization", METU-IE, Technical report, No.02-08.