The Journal of
Mathematics and Computer Science

Available online at
http://www.TJMCS.com

The Journal of Mathematics and Computer Science Vol .5 No.2 (2012) 91-96

A Genetic Algorithm for Solving Scheduling Problem

Habibeh Nazif
Department of Mathematics, Payame Noor University, IRAN
h_nazif@pnu.ac.ir

Received: February 2012, Revised: November 2012
Online Publication: December 2012

Abstract

This paper considers a single machine family scheduling problem where jobs are partitioned into families and setup is required between these families. The objective is to find an optimal schedule that minimizes the total weighted completion time of the given jobs in the presence of the sequence independent family setup times. This problem has been proven to be strongly NP-hard. We introduce a genetic algorithm that employs an innovative crossover operator that utilizes an undirected bipartite graph to find the best offspring solution among an exponentially large number of potential offspring. Computational results are presented. The proposed algorithm is shown to be superior when compared with other local search methods namely the dynamic length tabu search and randomized steepest descent method.

Keywords: genetic algorithm; single machine scheduling; completion time.

2010 Mathematics Subject Classification: 90C27.

1. Introduction

The Single Machine Family Scheduling Problem (SMFSP) exists in many diverse areas from flexible manufacturing systems to airline industry. The main focus is on the efficient allocation of one or more resources to activities over time. For some recent surveys on the real world applications of the machine scheduling problems, see the collection of [16]. Due to the complexity studies conducted during the last three decades, it is now widely understood that the SMFSP for arbitrary family f is an NP-hard problem as shown by [8].

This paper focuses on SMFSP where N jobs are partitioned into F families. For each family f ($f = 1, 2, \dots, F$), jobs are split into batches, where a batch is defined as a maximal set of contiguously scheduled jobs from the same family which share the same setup time. A sequence independent family setup time s_f, is required at the start of the schedule and also whenever there is a switch in processing jobs from one family to jobs of another family. We assume that all the jobs are available at time zero, and that each job has a given processing time p_j and an associated positive weight w_j, for $j = 1,2, \dots, N$. The objective is to find a schedule which minimizes the total

weighted completion time of the jobs in the presence of the sequence independent family setup times. Based on the standard classification of [12], this problem can be represented as $1|s_f|\sum w_j C_j$. When all setup times are zero, the problem is solved in $O(N \log N)$ time by the Shortest Weighted Processing Time (SWPT) rule of [23] where jobs are sequenced in non-decreasing order of their processing time to weight ratios. This ordering is used as the priority rule for scheduling more complex system.
Set ups include adjusting tools, paint drying, cleanup, etc. The majority of scheduling research assumes setup times as either negligible and hence ignored or considered as part of the processing time. While this assumption simplifies the problem, it adversely affects the solution quality for many applications which require explicit treatment of setup.
Numerous optimization methods including exact methods, heuristics and local search algorithms have been proposed for the problem of $1|s_f|\sum w_j C_j$. Comprehensive reviews on scheduling with setup considerations are given by [4], [5], and [22]. Reference [20] also showed that jobs within each family are sequenced in SWPT order in an optimal schedule.
Reference [19] proposed a branch and bound algorithm by defining the changeover for a job in a new family as the set down operation from the previous family and a set up operation for the new family. Later, [10] developed a branch and bound algorithm by performing a Lagrangian relaxation to the problem. Reference [11] proposed two new lower bounds that are shown analytically to dominate the lower bound of [19] and can be computed more efficiently than the Lagrangian lower bound of [10].
References [2], [13], and [25] suggested numerous heuristics for the problem of $1|s_f|\sum w_j C_j$ Computational results shown that the heuristics of [2] and [25] generate better quality solutions compared to heuristic of [13]. Reference [18] proposed a genetic algorithm (GA) using a binary representation of solutions. A GA based on fundamental runs theory is proposed by [24]. Reference [9] developed four local search heuristics (descent method, simulated annealing, threshold accepting, and tabu search) for the problem. Reference [15] considered the problems with setup times that are proportionate to the length of the already scheduled jobs. Two heuristics are presented by [14] for the problem with one planned setup period, with the aim of minimizing the weighted sum of the completion times.
In this paper, we use the Optimized Crossover Genetic Algorithm (OCGA) proposed by [21] for the problem of $1|s_f|\sum w_j C_j$ to achieve better solution quality compared to other local search algorithms. The proposed OCGA employs an innovative crossover operator that utilizes an undirected bipartite graph to find the best offspring solution among an exponentially large number of potential offspring.

2. Optimized Crossover Genetic Algorithm

The concept of optimized crossover was first proposed by [1] for the independent set problem. They used the merge operation formulated by [6] as an optimized crossover and constructed a bipartite graph from the two parent independent sets to produce the two new children. The first child determined using a matching algorithm in the graph and the second child is obtained by taking all nodes in the union of parents which do not lie in the independent set corresponding to the optimum child. Computational results indicated that the proposed algorithm is one of the best heuristics for independent set problem in terms of robustness and time performance. Reference [7] incorporated this approach to produce a superior GA. They examined variations of each element of the GA and developed a steady state replacement that performed better than its competitors on most problems. A greedy GA is proposed by [3] for the quadratic assignment problem. They determined an optimized child using a complete undirected bipartite graph constructed by the two parents.
We first proposed an optimized crossover operator within a GA for the problem of $1|s_f|\sum w_j C_j$ in [21]. Using the optimized crossover scheme, the two parents produced two new children: the O-child (Optimum child) and the E-child (Exploratory child). The O-child is constructed in a way so as to have the best objective function value from the feasible set of children, while the E-child is constructed so as to maintain the diversity of the search space.
In order to generate O-child and E-child, the optimized crossover first constructs an undirected bipartite graph from a pair of selected parents. Then, it determines all the maximum matching of

the graph, where each matching representing a temporary offspring in the respective problems. There will be exactly 2^k temporary offspring where k denotes the number of jobs of selected family which represent a different bit situation in the selected parents. However, we consider maximum 2^5 temporary offspring in the graph in every case to prevent the method from being too time consuming. Finally, a temporary offspring with the best (i.e. least) objective function value is selected as the O-child. The E-child is obtained by taking all genes in the union of parent individuals which do not lie in the individual corresponding to the O-child, together with all genes in the intersection of the parent individuals. For more details see [21].

Briefly, the OCGA uses the binary representation to define the partition of family into batches where '1' means the first job in a batch and '0' means a contiguously sequenced job in a batch. The length of the individual corresponds to the number of jobs N. The probabilistic binary tournament selection scheme is used to select the parents. A pair of selected parents from the population will produced two offspring via the optimized crossover operator explained earlier with a crossover probability, p_c. A F-point swap operator is used to produce offspring when the optimized crossover is not applied. This operator starts by selecting a random point within a family from a parent then swaps the substrings separated by the point. This process is repeated for each family within the parent to form a new offspring. To maintain the population diversity, a binary mutation is randomly applied to each offspring, where each element in the offspring is visited and altered with a given probability p_m = $1/N$. Finally, an elitism replacement scheme with filtration strategy is used to preserve good solutions and to avoid premature convergence. The algorithm of the OCGA is given below:

Algorithm OCGA (source: [21])

```
begin
    Initialize Population (randomly generated);
    Fitness Evaluation;
    repeat
        Probabilistic Binary Tournament Selection;
        Optimized Crossover;
            F-point Swap (no optimized crossover);
        Binary Mutation;
        Fitness Evaluation;
        Elitism Replacement with Filtration;
    until (end condition is satisfied);
    return (fittest solution found);
end
```

3. Competitors

To assess the performance of the proposed OCGA, we compare the algorithm with two local search methods namely Dynamic Length Tabu Search (DLTS) and Randomized Steepest Decent Method (RSDM) in solving the problem of $1|s_f|\sum w_j C_j$. DLTS and RSDM are first proposed by [17] for the problem of $1|s_f|\, L_{max}$. We believe that the algorithms will be useful to compare with the proposed OCGA. Thus we modify the algorithms to solve the problem of minimizing the total weighted completion time.

The DLTS applies the shift job neighborhood approach and dynamically controls the length of the tabu list during implementation in order to achieve better solution quality. Detailed descriptions of the shift-job neighborhood approach can be found in [2] and [9]. The basic role of a tabu list is to prevent cycling. Therefore, the length of the tabu list should be as short as possible but long enough to allow the search to move away from the local optimum. Such processes can have an important influence on which moves are available to be selected at a given iteration.

After a move is executed, the job that is shifted is stored in the tabu list, or both jobs are stored if the move is effectively transpose of the adjacent jobs. Thus, a neighbor is tabu if it is generated by shifting one of the jobs in the tabu list. An aspiration criterion is used in DLTS, in which if the solution value of a tabu neighbor is better than that for all solutions generated thus far, then its tabu status is overridden.

In the case of RSDM, the same shift-job neighborhood search procedure is employed. The RSDM adopts an acceptance rule that allows neutral moves to be made for up to M consecutive iterations where M is a parameter, before terminating the algorithm. Hence, when multiple identical good solutions are found in a single iteration, the algorithm selects a move randomly from the list of the identical good solutions. This strategy is applied to escape the search from falling into the same local optimum. Also note that deteriorating moves are not considered in the algorithm.

4. Results and Discussion

We present the results of the computational experiments of the algorithms for the problem of $1|s_f|\sum w_j C_j$. All algorithms are coded in ANSI-C language and implemented on a Pentium 4, 2.0 GHz computer with 2.0 GB RAM. Problem instances are generated with 50 and 100 jobs, and with 4, 8 and 12 families. Jobs are distributed uniformly across families, so that each family contains $\lfloor N/F \rfloor$ or $\lceil N/F \rceil$ jobs. This classification of problem instances has been widely used by other literature. Based on the problem instances proposed by [11], the two setup times class medium (A) and small (B) are randomly generated integers from the two uniform distributions of [0,100], [0,50] respectively. The three sets of integer processing times are randomly generated from the uniform distributions of [1,100], [1,10] and [1,1000]. Also job weights are randomly sampled integers from the uniform distribution of [1,10]. For each combination of *N*, *F*, processing times and setup times class, five problem instances are created.
We used the lower bound proposed by [11] to assess the quality of solutions generated by the algorithms. Algorithms are compared by listing, for each combination of value *N*, *F*, processing times and setup times class, the average relative percentage deviation (*ARD*) and the maximum relative percentage deviation (*MRD*) of the heuristic solution value from the lower bound. The *ARD* and *MRD* formulas are shown in equations (1) and (2) respectively.

$$ARD = \frac{\sum_{i=1}^{I}\sum_{r=1}^{R}(\frac{UB_{ir}-LB_i}{LB_i}\times 100\%)}{IR} \quad (1)$$

$$MRD = \max_{\substack{i=1,2,\ldots,I \\ r=1,2,\ldots,R}} (\frac{UB_{ir}-LB_i}{LB_i} \times 100\%) \quad (2)$$

where,

I= number of problem instances with the relevant combination of parameters;
R= number of repeated runs for problem instance *i* ($i = 1, 2, \ldots, I$);
UB_{ir}= heuristic solution found in *r*th run of problem instance *i*;
LB_i= lower bound of the problem instance *i*.

The computational results for the problem $1|s_f|\sum w_j C_j$ are shown in Table 1. For each combination of problem instances, 30 runs were performed. In order to have a fair comparison between different algorithms, we employed a duration of 15 CPU seconds per run in this experiment. Table 1 shows the computational results in which for each algorithm, the entries report the average value of *ARD* and *MRD* computed over the five problem instances with three combinations of processing times (i.e. 450 runs). For each setup class, the final line gives the average over all values of *N* and *F*. The final line of Table 1 gives the overall average value over all setup classes.
From Table 1, we can clearly conclude that the OCGA outperforms the RSDM and DLTS algorithms. In fact, our proposed algorithm is better in generating high quality solutions compared to others. We have also found that computational difficulty as measured by relative deviation from the lower bound increases with problem size. When processing times are sampled from the restrictive [1,1000] range, the total weighted completion time is considerably increased. However, with the [1,10] small range of processing times the total weighted completion time is significantly decreased. In addition, results indicate that the problem instances with setup class *B* is relatively easier to solve compared to setup class *A*.

Table 1. Comparative computational results (15 CPU seconds per run)

Setup Class	N	F	**DLTS**		**OCGA**		**RSDM**	
			ARD	MRD	ARD	MRD	ARD	MRD
A	50	4	16.43	35.55	14.92	28.56	18.57	44.00
		8	20.22	44.40	18.86	35.15	21.55	49.04
		12	20.13	41.40	17.94	30.95	20.97	44.05
	100	4	12.66	30.32	11.21	22.53	16.45	42.31
		8	18.51	46.06	16.49	36.18	20.69	53.45
		12	18.65	44.85	16.11	32.44	20.36	48.65
Average			17.77	40.43	**15.92**	**30.97**	19.77	46.92
B	50	4	11.38	27.22	9.75	19.02	13.12	32.64
		8	15.40	35.82	13.05	27.28	16.69	39.73
		12	14.73	38.13	12.41	29.51	15.58	39.55
	100	4	11.59	28.35	10.05	20.21	14.67	37.47
		8	17.33	42.11	14.55	32.46	19.72	51.49
		12	16.42	37.39	13.77	27.82	17.75	39.15
Average			14.48	34.84	**12.26**	**26.05**	16.26	40.01
AVERAGE			16.13	37.64	**14.09**	**28.51**	18.02	43.47

5. Conclusion

This paper presents a genetic algorithm that uses an optimized crossover operator to solve the problem of $1|s_f|\sum w_j C_j$. Various techniques have also been introduced into the proposed algorithm to further enhance the solutions quality. The computational results indicated that the proposed algorithm is able to generate better quality solutions compared to other variants of local search methods.

References.

[1] C. C. Aggarwal, J. B. Orlin, and R. P. Tai, Optimized crossover for the independent set problem, Operations Research, 45(1997), 226-234.

[2] B. H. Ahn and J. H. Hyun, Single facility multi-class job scheduling, Computers & Operations Research, 17(1990), 265-272.

[3] R. K. Ahuja, J. B. Orlin, and A. Tiwari, A greedy genetic algorithm for the quadratic assignment problem, Computers & Operations Research, 27(2000), 917-934.

[4] A. Allahverdi, J. N. D. Gupta, and T. Aldowaisan, A review of scheduling research involving setup considerations, OMEGA, International Journal of Management Science, 27(1999), 219-239.

[5] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, A survey of scheduling problems with setup times or costs, European Journal of Operational Research, 187(2008), 985-1032.

[6] E. Balas and W. Niehaus, Finding large cliques in arbitrary graphs by bipartite matching, In: Clique, Coloring and Satisfiability: Second DIMACS Implementation Challenge, D.S., Johnson and M.A., Trick, Eds., (1996), 29-53.

[7] E. Balas and W. Niehaus, Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems, Journal of Heuristics, 4(1998), 107-122.

[8] J. Bruno and P. Downey, Complexity of task sequencing with deadlines, set-up times and changeover costs, SIAM Journal on Computing, 7(1978), 393-404.

[9] H. A. J. Crauwels, C. N. Potts, and L. N. Van Wassenhove, Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion time, Annals of Operations Research, 70(1997), 261-279.

[10] H. A. J. Crauwels, A. M. A. Hariri, C. N. Potts, and L. N. Van Wassenhove, Branch and bound algorithm for single machine scheduling with batch set-up times to minimize total weighted completion time, Annals of Operations Research, 83(1998), 59-76.

[11] S. Dunstall, A. Wirth, and K. Baker, Lower bounds and algorithms for flowtime minimization on a single machine with set-up times, Journal of Scheduling, 3(2000) 51-69.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics, 5(1979), 287-326.

[13] J. N. D. Gupta, Single facility scheduling with multiple job classes, European Journal of Operational Research, 33(1998), 42-45.

[14] I. Kacem and C. Chu, Worst-case analysis of the WSPT and MWSPT rules for single machine scheduling with One planned setup period, European Journal of Operational Research, 187(2008), 1080-1089.

[15] C. Koulamas and G.J. Kyparisis, Single-machine scheduling problems with past-sequence-dependent setup times, European Journal of Operational Research, 187(2008), 1045-1049.

[16] J. Y-T. Leung, Handbook of scheduling: Algorithms, models, and performance analysis, Chapman & Hall/CRC, US, (2004).

[17] L. S. Lee, C. N. Potts, and J. A. Bennell, A genetic algorithms for single machine family scheduling problem, Proceedings of 3rd IMT-GT 2007 Regional Conference on Mathematics, Statistics and Applications, (2007), 488-493.

[18] A. J. Mason, Genetic algorithms and scheduling problems, PhD Dissertation, University of Cambridge, UK, (1992).

[19] A. J. Mason and E. J. Anderson, Minimizing flow time on a single machine with job classes and setup times, Naval Research Logistics, 38(1991), 333-350.

[20] C. L. Monma and C. N. Potts, On the complexity of scheduling with batch setup time, Operations Research, 37(1989), 798-804.

[21] H. Nazif and L. S. Lee, A genetic algorithm on single machine scheduling problem to minimise total weighted completion time, European Journal of Scientific Research, 35(2009), 444-452.

[22] C. N. Potts and M. Y. Kovalyov, Scheduling with batching: a review, European Journal of Operational Research, 120(2000), 228-249.

[23] W. E. Smith, Various optimizers for single-stage production, Naval Research Logistics Quarterly, 3(1956), 59-66.

[24] D. Wang, M. Gen, and R. Cheng, Scheduling grouped jobs on single machine with genetic algorithm, Computer and Industrial Engineering, 36(1999), 309-324.

[25] D. Williams and A. Wirth, A new heuristic for a single machine scheduling problem with set-up times, Journal of the Operational Research Society, 47(1996), 175-180.