Local Search Algorithms for Multiobjective Scheduling Problem

Dr. Adawiya A. Mahmood Al-Nuaimi
al_nuaimi85@yahoo.com
Diyala University - College of Science
Department of Mathematics

Abstract: *This paper presents local search algorithms for finding approximation solutions of the multiobjective scheduling problem within the single machine context, where the problem is the sum of the three objectives total completion time, maximum tardiness and maximum late work.*

Late work criterion estimates the quality of a schedule based on durations of late parts of jobs. Local search algorithms descent method (DM), simulated annealing (SA) and genetic algorithm (GA) are implemented. Based on results of computational experiments, conclusions are formulated on the efficiency of the local search algorithms.

Keywords: Local search, multiobjective scheduling, late work criterion, genetic algorithm.

1. Introduction

Real world problems arising in various application domains are usually strictly related to time [1]. Time constraints are important from two points of view. They determine feasibility conditions and they make it possible to evaluate the quality of feasible solutions. In scheduling theory, time restrictions are usually modeled by due dates or deadlines and the quality of schedules is estimated with reference to these parameters. Performance measures involving due dates model, the informal criteria that are applied by practitioners. This makes these objective functions a very attractive and widely explored subject of research. A significant part of scheduling literature is devoted to classical objective functions, such as minimizing the maximum lateness, mean or total tardiness and the number of tardy jobs or mean earliness – tardiness (cf. e.g. [2],[3],[4]). Late work performance measures are not so widely explored.

The scheduling problem is defined as a problem of assigning a set of jobs to a set of machines in time under given constraints ([2],[3],[4]). Jobs j ($j=1,2,\ldots,n$) are mainly characterized by processing times (p_j) due dates (d_j), define expected completion times ($C_j = \sum_{i=1}^{j} p_i$) for particular schedule of jobs.

The late work criterion estimates the quality of a solution (a schedule) on the basis of the duration of late parts of particular jobs. In the non- preemptive case the late work parameter for job j in a given schedule is defined as

$$V_j = \begin{cases} 0 & \text{if } C_j \le d_j ,\ j = 1,2,\ldots,n \\ C_j - d_j & \text{if } d_j < C_j < d_j + p_j ,\ j = 1,2,\ldots,n \\ p_j & \text{if } C_j \ge d_j + p_j ,\ j = 2,\ldots,n \end{cases}$$

The phrase " late work " was proposed by Potts and Van Wassenhove [5]. Some researchers, e.g. Hochbaum and Shamir [6], use a descriptive name for this schedule parameter – the number of tardy job units.

Leung [7] pointed out application of late work scheduling in computerized control systems, where data are collected and processed periodically.

The multiobjective problem that is considered concerns the minimization of a linear function of total completion time $\sum_{j=1}^{n} C_j$, maximum tardiness (T_{max}) and maximum late work (V_{max}). This problem belongs to the class of simultaneous multiobjective problems. In the simultaneous problems, two or more than two criteria are considered simultaneously. Sen and Gupta [8] solve the problem $1//L_{max}+\sum_{j=1}^{n} C_j$ by a branch and bound (BAB) algorithm.

The organization of this paper is as follows. Section 2 presents the problem formulation. Section 3 provides the local search approximation algorithms. Section 4 summarizes results of computational experiments and it is followed by conclusions are given in section 5.

2. Problem Formulation

A set of n independent jobs N = {1,2,…,n} are available for processing at time zero, job j (j=1,2,…,n) is to be processed without interruption on a single machine that can be handle only one job at a time, requires processing time p_j and due date d_j. For a given schedule σ of the jobs, completion time $C_{\sigma(j)} = \sum_{i=1}^{j} p_{\sigma(i)}$, maximum tardiness $T_{max}(\sigma) = \max\{T_{\sigma(1)}, T_{\sigma(2)}, \dots, T_{\sigma(n)}\}$, where $T_{\sigma(j)} = \max\{0, C_{\sigma(j)} - d_{\sigma(j)}\}$ and maximum late work $V_{max}(\sigma) = \max\{V_{\sigma(1)}, V_{\sigma(2)}, \dots, V_{\sigma(n)}\}$ can be computed where

$$V_{\sigma(j)} = \begin{cases} 0 & \text{if } C_{\sigma(j)} \le d_{\sigma(j)}, \quad j=1,2,\dots,n \\ C_{\sigma(j)} - d_{\sigma(j)} & \text{if } d_{\sigma(j)} < C_{\sigma(j)} < d_{\sigma(j)} + p_{\sigma(j)}, \quad j=1,2,\dots,n \\ p_{\sigma(j)} & \text{if } C_{\sigma(j)} \ge d_{\sigma(j)} + p_{\sigma(j)}, \quad j=2,3,\dots,n \end{cases}$$

The problem of minimizing a linear function of total completion time $\sum_{j=1}^{n} C_j$, maximum tardiness (T_{max}) and maximum late work (V_{max}) is denoted by $1//\sum_{j=1}^{n} C_j + T_{max} + V_{max}$ and called it (P). The problem (P) can be formulated as follows:

$$\left.\begin{aligned} & Z = \min_{\sigma \in S}\{ \textstyle\sum_{j=1}^{n} C_{\sigma(j)} + T_{max}(\sigma) + V_{max}(\sigma) \} \\ & \text{Subject to} \\ & C_{\sigma(j)} \ge p_{\sigma(j)} \\ & C_{\sigma(j)} = C_{\sigma(j-1)} + p_{\sigma(j)} \qquad j = 2,\dots,n \\ & T_{\sigma(j)} \ge C_{\sigma(j)} - d_{\sigma(j)} \qquad \forall j,\ j = 1,2,\dots,n \\ & T_{\sigma(j)} \ge 0 \\ & V_{\sigma(j)} \le C_{\sigma(j)} - d_{\sigma(j)} \\ & V_{\sigma(j)} \le p_{\sigma(j)} \\ & V_{\sigma(j)} \ge 0 \end{aligned}\right\} \text{-------(P)}$$

Where S is the set of all schedules.

The aim in problem (P) is to find a processing order of the jobs on a single machine to minimize the sum of total completion time, maximum tardiness and the maximum late work (i.e. to minimize the $1//\sum_{j=1}^{n} C_j + T_{max} + V_{max}$).

3. Local Search Approximation Algorithms

Local search algorithms can find the best approximation solution within a reasonable running time. Local search is a family of methods that iteratively search through the set of solutions. Starting from an initial solution which is obtained by applying some constructive heuristic, a local search procedure moves from one feasible solution to a neighboring solution until some stopping criteria are met. The choice of suitable neighborhood function has an important influence on the performance of local search. These neighborhood functions define the set of solutions to which the local search procedure may move to a single iteration [9]. This is formalized in the following definition:

Definition I [9]: An instance of a combinatorial optimization problem is a pair (S , f), where the solution set S is the set of all feasible solutions and the cost function f is a mapping $f:S \longrightarrow R$. The problem is to find a globally optimal (minimal) solution, i.e. an $s^* \in S$, such that $f(s^*) \leq f(s)$ for all $s \in S$.

3.1 Solution Representation [9]

Solution representation depends on the problem specification. In a scheduling problem of n jobs, a solution is represented by a permutation of the integers 1,2,…,n.

Definition II[10]: A neighborhood function N* is a mapping

N*:S $\longrightarrow$ P(S) which specifies for each s ∈ S a subset N*(s) of S neighbors of s.

For sequencing problems, the natural representation is a permutation of the integers 1, 2,…, n with n the number of jobs. With this representation, two basic neighborhoods can be defined [9]. Each of which is illustrated by considering a typical neighbor of the sequence (1, 2, 3, 4, 5, 6, 7, 8).

1. Insert(shift): Remove a job from position i in the sequence and insert it at position j (either before (left insert) or after (right insert) the original position). Thus (1,**6**,2,3,4,5,7,8) and (1,2,3,4,5,7,**6**,8) are both neighborhoods.
2. Swap(interchange): Swap two jobs (i,j) which may not be adjacent. Thus (1,**6**,3,4,5,**2**,7,8) is a neighbor.

Definition III [9]: Let (S , f) be an instance of a combinatorial optimization problem and let N* be a neighborhood function. A solution s*∈ S is called a local optimal (minimal) solution with respect to N* if f(s*) ≤ f(s) for all s ∈ N*(s*). The neighborhood function N* is called exact if every local minimum with respect to N* is also a global minimum.

3.2 Descent approximation method (DM)[10]

It is the simplest type of neighborhood search, which is sometimes known as iterative local improvement. In this method, only moves that result in an improvement in the objective function value are accepted.

The main components of a descent method are as follows:

1. Initialization

The search has to be initialized with an initial solution. This solution can be constructed by some heuristic method or it can be chosen at random.

2. Generating of neighborhood

To generate a neighborhood, the two basic neighborhoods insert or swap which are illustrated in section 3.1 can be applied.

3. Stopping criterion

There are many ways for stopping criterion of the method, the one that is used in this paper is a fixed number of iterations, in more precisely the method is terminated after 18000 iteration at a near optimal solution.

3.3 Simulated annealing approximation (SA) algorithm [10]

Simulated annealing (SA) is a probabilistic algorithm that is applied in combinatorial optimization problems. In simulated annealing, probabilistic acceptance rule is used. More precisely, any move that results in an improvement in the objective function value, or leaves the value unchanged, is accepted. On the other hand, a move that increases the objective function value by Δ is accepted with probability $\exp(-\Delta / T)$, where T is a parameter known as the temperature. The value of T changes during the course of the search; typically T starts at a relatively high value and then gradually decreases.

The main components of simulated annealing are as follows:

1. **Initialization**

The initial solution can be constructed by some heuristic method or it can be chosen at random. This solution will be the initial current solution for simulated annealing method with objective function value Z.

2. **Neighborhood generation**

To generate a neighborhood, the two basic neighborhoods insert or swap which are illustrated in section 3.1 can be applied.

3. **Accepting test**

In this step the difference value between the initial current solution Z and the new value Z', $\Delta = Z' - Z$ is calculated and evaluated as follows:

a) If $\Delta \le 0$, then Z' is accepted as the new current solution and set Z = Z'.

b) If $\Delta > 0$, then Z' is accepted with $p(\Delta) = \exp(-\Delta / T)$, which is the probability of accepting a move, where T is a known temperature.

4. **Termination test**

The algorithm is terminated after 18000 iterations at near optimal solution.

3.4 Genetic approximation algorithm (GA)[11]

Genetic approximation algorithms are search techniques based on simplification of natural evolutionary process, genetic approximation algorithms operate on a population of solutions rather than a single solution and employ heuristics such as selection, crossover and mutation to evolve better solutions. A population of solutions is represented by a string of a finite set of parameters called chromosomes. In scheduling problem, a chromosome includes genes that are a number of jobs that should be applied on one machine with permutation.

The main components of a genetic algorithm are as follows:

1. Initialization

Initially many individual solutions are generated randomly or produced with a good heuristic to form an initial population. The chromosomes should be encoded. There are many ways to encode the initial generation, for scheduling problems the natural representation is usually used. The natural representation consists of permutation of the jobs 1,2,…,n which defines the processing order of the jobs.

2. Fitness (evolution)

The evolution function plays an important role in genetic algorithm. The fitness function is equivalent to the objective function in traditional optimization methods.

3. Selection

It is the stage of a genetic algorithm in which individual chromosomes are chosen from a population for later breeding (recombination or crossover). The process of selection is used to form each subsequent population during the evolutionary process. This process is done by computing the fitness function of each candidate solution and then selecting the individuals (according to fitness value usually) that will form the next generation parents.

4. Crossover

Crossover is the process of making new candidate solutions from previous candidate solutions. This process uses the principles of biological evolution to form children (new candidates) from parents (old candidates). We used homogeneous mixture crossover (HMX) on each pair of parent solutions to generate two new solutions. This crossover is given by the mixture of the two parents uniformly by making a set (m) from genes, the odd position from the first parent and the even position from the second parent. Then separate genes without repetition of the gene, since we read the set (m) from the left, if the gene g does not exist in the child then keep it and put (o) in (m), otherwise we keep gene g in the second child and put (1) in (m), until (m) genes are exhausted. This way also gives two new children. This crossover preserves the absolute positions taken from one parent and the relative positions of those from the other parent. For example:

	Parents	Mixture	Exchanging
Parent1	798251634	799586245813623741	795862413 child1
Parent2	956483271	001000001100111111	958623741 child2

5. Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. Mutation randomly changes one or more element of the string (permutation).

6. Termination

The termination operator is the mechanism for stopping the evolutionary process of a genetic algorithm. Termination can occur when the algorithm has reached a predetermined number of generations (iterations), an acceptable solution has been found, or when no solution improvement has occurred for a period of time. In this search GA is terminated if the best chromosome of the population doesn't change for 100 consecutive generations.

4. Test Problems and Computational Results with the Local Search Approximation Algorithms

Test problems are generated as follows: for each job j, an integer processing time p_j is generated from the discrete uniform distribution [1, 10]. Also, for each job j, an integer due date d_j is generated from the discrete uniform distribution [P (1-TF-RDD/2), P (1-TF+RDD/2)], where $P = \sum_{j=1}^{n} p_j$, depending on the relative range of due date (RDD) and on the average tardiness factor (TF). For both parameters, the values 0.2, 0.4, 0.6, 0.8, 1.0 are considered. For each selected value of n, two problems are generated for each of the five values of parameters producing 10 problems for each value of n, where the number of jobs n=50,100,200,300.

The local search algorithms (i.e. descent method(DM), simulated annealing(SA) and genetic algorithm(GA)) for the problem $(1//\sum_{j=1}^{n} C_j + T_{max} + V_{max})$ (P) are tested by coding them in Matlab R2009b and running on a personal computer hp with RAM 2.50 GB. Each algorithm stops when it attends to a fixed number of iteration. The (DM) and (SA) stop after 20,000 iteration and (GA) stops after 100 new generation of population.

The computational results of local search algorithms for the problem (P) are given in tables (4.1) to (4.4) that show the values of each local search algorithm and how many times each of them catches the best value. Each table contains 10 problems where:

EX = example number.

DM = descent method value for the problem P.

SA = simulated annealing value for the problem P .

GA = genetic algorithm value for the problem P .

Best = the best value.

No. best = number of examples that catch the best value.

Av. Time = the average of time in second for getting solution.

Table (4.1): The results of local search algorithms for n =50.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	6395	6423	0.582676	6462	0.506278	6395	2.3971
2	4765	4780	0.499135	4774	0.500460	4765	2.4087
3	4304	4349	0.494884	4325	0.494979	4304	2.3592
4	6019	6060	0.489996	6055	0.488871	6019	2.4186
5	4164	4214	0.486802	4219	0.502979	4164	2.4088
6	5280	5299	0.494951	5308	0.494046	5280	2.3756
7	5838	5874	0.495978	5890	0.502825	5838	2.4129
8	5191	5227	0.506029	5215	0.494777	5191	2.4127
9	5097	5114	0.480152	5122	0.495186	5097	2.3820
10	5288	5328	0.491072	5312	0.489667	5288	2.3719
No. best		0	Av. Time 0.5021675	0	Av. Time 0.4970068	10	Av. Time 2.39475

Table (4.2): The results of local search algorithms for n= 100.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	18987	19390	0.734482	19401	0.658349	18987	5.2709
2	20723	21235	0.661327	21226	0.653215	20723	5.5581
3	19211	19590	0.674532	19552	0.660796	19211	5.4432
4	20409	20830	0.655261	20790	0.649451	20409	5.4264
5	19317	19700	0.652724	19698	0.649839	19317	5.4152
6	20497	20812	0.649166	20797	0.672833	20497	5.3507
7	18659	19071	0.652451	19080	0.654630	18659	5.6460
8	17551	17892	0.647136	17899	0.658415	17551	5.4470
9	20078	20389	0.631262	20382	0.693263	20078	5.3957
10	20943	21298	0.649257	21300	0.654339	20943	5.3995
No. best		0	Av. Time 0.6607598	0	Av.Time 0.660513	10	Av. Time 5.43527

Table (4.3): The results of local search algorithms for n= 200.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	71105	76133	1.198537	76053	0.955301	71105	15.9462
2	78611	82815	0.967703	82779	0.973503	78611	16.9266
3	85370	89810	0.970000	89711	0.971836	85370	16.0068
4	78768	84688	0.961153	84605	0.975451	78768	15.9601
5	74506	79357	0.958148	79363	0.958554	74506	15.9958
6	84066	88457	0.983067	88502	0.976251	84066	16.9431
7	80723	86142	0.962179	85949	0.970662	80723	16.1494
8	87814	93941	0.964851	93936	0.979546	87814	15.9670
9	76343	81607	0.958335	81788	0.966122	76343	15.2053
10	78499	83276	0.966904	83214	0.975118	78499	15.9035
No. best		0	Av. Time 0.9890877	0	Av. Time 0.9702344	10	Av.Time 16.10038

Table (4.4): The results of local search algorithms for n=300.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	119294	128396	1.263755	128360	1.239453	119294	34.8934
2	120367	130709	1.216018	130907	1.186773	120367	34.6639
3	123204	133171	1.880526	133294	1.205262	123204	33.2739
4	110212	119518	1.188934	119518	1.254122	110212	34.0288
5	123287	134628	1.199343	134525	1.174999	123287	33.3615
6	120395	129182	1.353742	129174	1.328534	120395	34.9723
7	125349	131618	1.225027	131618	1.243116	125349	34.7649
8	122456	132182	1.870435	131726	1.885144	122456	33.4627
9	112351	122439	1.199825	122439	1.263254	112351	34.1293
10	127426	138746	1.399265	138927	1.397453	127426	33.4537
No. best		0	Av. Time 1.379687	0	Av. Time 1.317811	10	Av.Time 34.10064

The average computation time of (DM) is close to that of (SA) while for (GA) is large. The results of the above tables show that the (GA) performs very well.

5. Conclusions

In this paper local search algorithms (descent method (DM), simulated annealing (SA) and genetic algorithm (GA)) are proposed to find approximation solutions for the problem of minimizing a multiobjective $\sum_{j=1}^{n} C_j + T_{max} + V_{max}$. Computational experiments for the local search algorithms on a large set of test problems are given.

The main conclusion to be drawn for our computational results is that the genetic algorithm (GA) is more effective, whereas the average computation time of descent method (DM) is close to that of simulated annealing (SA) while for genetic algorithm (GA) is large.

An interesting future research topic would involve experimentation with the following multiobjective problems :

1. $1//F(T_{max}, \sum_{j=1}^{n} V_j, \sum_{j=1}^{n} T_j)$.
2. $1//F(\sum_{j=1}^{n} V_j, V_{max}, \sum_{j=1}^{n} T_j)$.

References

[1] Sterna M., "A survey of scheduling problems with late work criteria", Omega 39: 120-129; 2011.

[2] Blazewicz J., Ecker K., Pesch E., Schmidt G., Weglarz J., "Handbook on scheduling. From theory to applications". Berlin - Heidelberg -New York: Springer; 2007.

[3] Brucker P., "Scheduling algorithms", Berlin-Heidelberg-New York: Springer; 2007.

[4] Pinedo M., "Scheduling: theory, algorithms and systems", New York: Springer; 2008.

[5] Potts CN., Van Wassenhove LN., "Single machine scheduling to minimize total late work", Operations Research 40 (3): 586-95; 1991.

[6] Hochbaum DS, Shamir R., "Minimizing the number of tardy job units under release time constraints", Discrete Applied Mathematics 28 (1): 45-57; 1990.

[7] Leung JY-T., "Minimizing total weighted error for imprecise computation tasks and related problems. In: JY-T Leung. Editor. Handbook of scheduling: algorithms, models and performance analysis", Boca Raton : CRC Press 34: 1-16; 2004.

[8] Sen T. and Gupta S. K., "A branch and bound procedure to solve a bicriterion scheduling problem", IIE Transactions 15: 84-88; 1983.

[9] Aarts E. H. L. and Lenstra J. K., "Local search in combinatorial optimization", John Wiley and Sons. Chichest; 1997.

[10] Vaessens R. J. M., Arats E. H. L. and Lenstra J. K., "A local search template", Computers and Operations Research 25 : 969-979; 1998.

[11] Wall M. B., "A genetic algorithm for resource constrained scheduling", Ph.D. in Mechanical Engineering at the Massachusetts Institute of Technology, June; 1996.

خوارزميات بحث محلية لمسألة جدولة متعددة الأهداف

م.د.عدوية علي محمود النعيمي
al_nuaimi85@yahoo.com
جامعة ديالى - كلية العلوم - قسم الرياضيات

المستخلص

ان هذا البحث يقدم خوارزميات بحث محلية لإيجاد حلول تقريبية لمسألة جدولة متعددة الأهداف على ماكنة واحدة حيث المسألة هي المجموع للأهداف الثلاثة مجموع أوقات الإتمام الكلي ، أعظم تأخير لاسالب وأعظم تأخير لوحدات عمل متأخر.

مقياس العمل المتأخر يخمن كفاءة الجدولة بالاعتماد على فترات زمنية للأجزاء المتأخرة للأعمال . أقترحت خوارزميات البحث المحلية وهي طريقة النزول ، طريقة تقوية المحاكاة والخوارزمية الجينية . بالاعتماد على نتائج التجارب الحسابية تم صياغة استنتاجات حول كفاءة خوارزميات البحث المحلية.

الكلمات الرئيسية: بحث محلي، جدولة متعددة الأهداف، مقياس عمل متأخر، خوارزمية جينية.