Journal of Southeast University (English Edition) Vol. 24, No. 4, pp. 450 – 454 Dec. 2008 ISSN 1003—7985

Hybrid evolutionary algorithm for no-wait flow shops to minimize makespan and total flowtime

Liao Xiaoping[1] Liu Yougen[2] Li Xiaoping[2]

([1]College of Computer and Information Engineering, Hohai University, Nanjing 210098, China)
([2]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: The NP-hard no-wait flow shop scheduling problems with makespan and total flowtime minimization are considered. Objective increment properties of the problems are analyzed. A non-dominated classification method is introduced to class population individuals into Pareto fronts to improve searching efficiency. Besides investigating the crowding distance and the elitist solution strategy, two effective bi-criteria local search procedures based on objective increments are presented to improve searching effectiveness. Based on the properties and methods, a hybrid evolutionary algorithm is proposed for the considered problems and compared with the best existing algorithms. Experimental results show that the proposed algorithm is effective with high efficiency.

Key words: no-wait flow shop; objective increment; makespan; total flowtime; evolutionary algorithm

The no-wait flow shop is a typical constrained scheduling problem in which the operations of each job should be processed without interruption between consecutive machines, i. e. , the start of a job must be delayed on the first machine when necessary so that the job need not wait for processing on subsequent machines. No-wait flow shops exist in such settings as chemical processing, metal processing and canning operations in food processing. Modern manufacturing systems such as just-in-time, flexible manufacturing environments, and robotic cells can also be modeled as no-wait flow shop scheduling problems.

For decades, more attention has been attracted on no-wait flow shop scheduling problems with a single criterion optimization, especially on makespan[1-6]. However, multi-criteria are taken into account at the same time in real practice. For the multi-objective no-wait flow shop scheduling problems, there are only a few algorithms currently available. Allahverdi and Aldowaisan[7] proposed a heuristic algorithm PAAH for minimizing a weighted sum of makespan and total flowtime. Later, they also proposed two meta-heuristic algorithms HAS and HG[8] to optimize a weighted sum of makespan and maximum lateness. Recently, Reza et al. [9] proposed the meta-heuristic algorithm HMOIA to optimize the weighted mean completion time and weighted mean tardiness. The considered objectives often conflict with each other. In other words, optimizing one objective often deteriorates other ones. Therefore, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. It is desirable to get a set of candidate solutions rather than just one solution for decision-makers; these are called Pareto solutions. A Pareto optimal set is a set of solutions in which none is dominated by another. A decision-maker can choose his preferred solution as the final solution according to some trade-off. The genetic algorithm (GA) is a popular method for multi-objective optimization problems. Jones et al. [10] reported that 90% of the approaches to multi-objective optimization aimed at approximating the true Pareto front for the considered problem.

In this paper, a hybrid evolutional algorithm based on objective increments[5-6] is designed for searching locally the Pareto-optimal frontier for no-wait flow shops in order to minimize makespan and total flowtime simultaneously.

1 Preparation

1.1 Problem description

A no-wait flow shop is a constrained flow shop scheduling problem with n jobs $\{J_1, J_2, \ldots, J_n\}$ being processed on m machines $M_1, M_2, \ldots, M_m$. Each job is processed sequentially on m machines without delay between adjacent machines. The start of a job must be delayed on the first machine when necessary so that the job need not wait for processing on subsequent machines. Let $O_{i,j}$ be the operation of job $j(j = 1, 2, \ldots, n)$ processed on machine $i(i = 1, 2, \ldots, m)$; $t_{i,j}$ be the processing time of $O_{i,j}$; $S_{i,j}$ and $C_{i,j}$ be the starting time and finish time of $O_{i,j}$. The optimization objectives are to find the schedule with both the minimum makespan and the total flowtime, i. e. finding the job sequence $\pi = (\pi_{[1]}, \pi_{[2]}, \ldots, \pi_{[n]})$ ($\pi_{[i]} \in \{J_1, J_2, \ldots, J_n\}$ is the i-th job of π with minimum $C_{\max}$ and minimum $\sum_{i=1}^{n} C_{m,i}$ (denoted as $F_n(\pi)$).

1.2 Non-dominated solutions

In a bi-objective optimization problem, a feasible solution x dominates another feasible solution $y(x \succ y)$, if and only if $z_i(x) \leqslant z_i(y)\ \forall i \in \{1, 2, \ldots, k\}$ and $z_j(x) < z_i(y)$ for at least one objective function j. A solution is said to be Pareto optimal if it is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective. The set of all feasible non-dominated solutions in X is referred to as the Pareto optimal set, and for a given Pareto optimal set, the corresponding objective function values in the objective space are called the Pareto front.

For a sequence S, the corresponding value of makespan and total flowtime are denoted by $Z_1(S)$, $Z_2(S)$ in this pa-

Received 2008-04-15.
Biography: Liao Xiaoping(1965—), male, lecturer, xpliao@ sina. com.
Foundation items: The National Natural Science Foundation of China(No. 60504029, 60672092), the National High Technology Research and Development Program of China(863 Program)(No. 2008AA04Z103).
Citation: Liao Xiaoping, Liu Yougen, Li Xiaoping. Hybrid evolutionary algorithm for no-wait flow shops to minimize makespan and total flowtime [J]. Journal of Southeast University (English Edition), 2008, 24(4): 450 – 454.

per. If $Z_r(S) \leqslant Z_r(S')\ (r=1,2)$ and $Z_r(S) < Z_r(S')$ for at least one r, sequence S dominates sequence S'. Otherwise, the two sequences non-dominate each other.

1.3 Objective increment

According to Refs. [5 – 6], in most heuristics for flow shops, there exist two fundamental operations, insertion and pair-wise exchange. In these heuristics for flow shop scheduling problems, both constructive and improvement phases generate new partial sequences from a parent partial sequence. In traditional heuristics, all new partial sequences are evaluated by computing the finish times of all the jobs. However, this is not necessary for a no-wait flow shop because there are only a few slots that are changed in a new partial sequence from its parent. Therefore, only the changed partial sequences are needed to be calculated. The difference of the objective function value between the new offspring sequence and its parent sequence is called the objective increment. It is obvious that the computational time can be considerably reduced just by computing objective increments.

Let $D_{i,j} = \max\limits_{k=1,\cdots,m}\left\{\sum\limits_{h=k}^{m}(t_{h,j}-t_{h,i})+t_{k,i}\right\}$ be the distance between the finish times of the two adjacent jobs J_i and J_j. $F_n(\pi) = \sum\limits_{i=0}^{n-1}(n-i)D^{\pi}_{[i],[i+1]} = \sum\limits_{i=0}^{n-1}(n-i)d^{\pi}_{[i]}$ denotes the value of the total flowtime of sequence π. Similar to Ref. [5], the following properties are true.

Theorem 1 If job J_q is inserted into the position between slot i and j of $\pi(0 \leqslant i < j \leqslant n)$, the makespan increment is (J_q is excluded from sequence π) $\delta(i,j,q) = D_{[i],q} + D_{q,[j]} - D_{[i][j]}$.

Theorem 2 When job $J_{[i]}$ and $J_{[j]}\ (0<i<j<n)$ of π are exchanged, the makespan increment is

$$\omega_e(i,j) = \begin{cases} \delta_{[i-1],[i],[i+1]} - \delta_{[i],[i+2],[i+1]} & j = i+1 \\ \delta_{[i-1],[i+1],[j]} + \delta_{[j-1],[i+1],[i]} - & \\ \quad \delta_{[i-1],[i+1],[i]} - \delta_{[j-1],[j+1],[j]} & \text{otherwise} \end{cases}$$

For simplicity, let $x_{i,j}(\pi)$, $s(k)$ and $y_{i,j}(\pi)$ be

$$x_{i,j}(\pi) = \begin{cases} 0 & i = n \\ (n-i)(D^{\pi}_{[i],[j]} - d^{\pi}_{[i]}) & \text{otherwise} \end{cases}$$

$$y_{i,j}(\pi) = \begin{cases} 0 & j = n \\ (n-j)(D_{i,\pi_{(j+1)}} - d^{\pi}_{[j]}) & \text{otherwise} \end{cases}$$

$$s(\pi) = \begin{cases} 0 & k < 0 \\ \sum\limits_{i=0}^{k} d^{\pi}_{[i]} & k \geqslant 0 \end{cases}$$

Theorem 3 If job J_q is inserted into slot j of $\pi(0 \leqslant j \leqslant n)$, the total flowtime increment is

$$\Delta_{T1}(n,j) = s(j-1) + (n-j+1)D_{\pi_{(j)},q} + y_{q,j}(\pi)$$

Theorem 4 When $J_{[i]}$ and $J_{[j]}\ (0<i<j<n)$ of π are exchanged, the total flowtime increment is

$$\omega_T(i,j) = \begin{cases} x_{i-1,j}(\pi) + y_{\pi_{(i)},j}(\pi) + (n-i)(D^{\pi}_{[j],[i]} - d^{\pi}_{[i]}) & j = i+1 \\ x_{i-1,j}(\pi) + y_{\pi_{(i)},i}(\pi) + x_{j-1,i}(\pi) + y_{\pi_{(i)},j}(\pi) & \text{otherwise} \end{cases}$$

In this paper, the above objective increment properties are adopted to evaluate the makespan and the total flowtime of any new generated solution in order to reduce computation time.

2 Population Classification

Generally, the Pareto sorting procedure applies to each generation of a multi-objective genetic algorithm, which is one of the key factors to the algorithm efficiency. In other words, the computational time of the algorithm can be reduced greatly if the efficiency of the Pareto sorting procedure is improved. During the sorting procedure, an obtained population of solutions is classified into many non-dominated fronts. All solutions on the first (i. e., the best) non-dominated front are non-dominated with respect to each other, and at least one of its solutions dominates another solution on the remaining non-dominated fronts. Similarly, all solutions on the second non-dominated front are non-dominated with respect to each other, and at least one of its solutions dominates another solution on the remaining fronts except the first front. All solutions in a population are classified by such a non-dominating relationship. All solutions on different fronts are assigned to different priorities. The solutions on the first non-dominated front are assigned with front 1, the highest priority. Fig. 1 shows the classification of an instance.

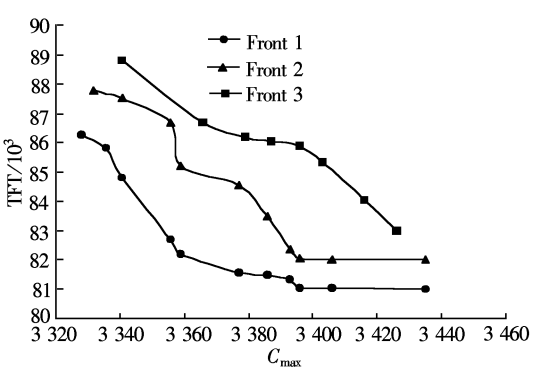

Fig. 1 Non-dominated solutions classification

All the solutions in a Pareto front(front 1, front 2, front 3, …) are arranged in increasing order of one performance measure and in non-increasing order of the other. So all solutions can be sorted by the divide-and-conquer method. And the main computation time focuses on sorting solutions by C_{max}.

3 Proposed Hybrid Evolutionary Algorithm

In this paper, a hybrid evolutionary algorithm based on the GA is proposed. To accelerate convergence to the proposed algorithm, NEH[11], GR[1], IH7[12], RAJ[2] and PAAH[7] are adopted to generate several solutions for the initial population. For a population with size N_p, the remaining $N_p - 5$ individuals are generated randomly. A binary-tournament selection is adopted and the tournament size is two. Two chromosomes are chosen randomly from the parent population. The more appropriate chromosome (the rank is smaller), the more chance to be selected. The chromosome with fewer fitness function values can also be accepted with a probability (e. g., 0.1). The two selected chromosomes

cannot be identical and are subsequently subjected to crossover and mutation. If the two selected chromosomes are on the same front, the crowding distance is introduced to another metric. This crowding distance measure is a tiebreaker in selection. If the solutions are on the same non-dominated front, the solution with a higher crowding distance is the winner. Otherwise, the solution with the lowest priority is selected. The crowding distance d_i of chromosome i with respect to all other chromosomes on the same front is used, which is defined as $d_i = \sum_{\substack{j \in \{F_i\} \\ i \neq j}} D_{ij}$. Here, $D_{ij} = \sqrt{\sum_{r=1}^{R} \left(\frac{Z_r(i) - Z_r(j)}{\max_{i' \in \{F_i\}} \{Z_r(i')\} - \min_{i' \in \{F_i\}} \{Z_r(i')\}} \right)^2}$ is the crowding distance between chromosome i and j. f_i denotes the front on which chromosome i lies, and $\{F_i\}$ denotes the set of chromosomes that lie on the same front as that of chromosome i.

Single-point crossover is adopted with crossover probability 1. Though the mutation probability is 0.1, mutation conducted or not is determined by a random number u. Mutation is performed only when $u \leqslant 0.1$. During a mutation, two random positions i and j are chosen, the corresponding jobs are exchanged.

The parent population p_t and the offspring population O_t with size N_p is combined as C_t, i.e. $C_t = p_t \cup O_t$. C_t is also classified into fronts. The next generation is determined by: 1) Choose all the best chromosomes(on front 1) in the combined population C_t and copy them to the archive; 2) Choose N_p chromosomes from the combined population C_t and copy them to the next generation population p_{t+1}.

A new archive is used to maintain the elitist solutions. In order to find more and better non-dominated solutions on the best front, a light local stir strategy is adopted in the archive. An update procedure is first presented.

1) For sequence π_i, after a local stir operation(i.e., insertion and swap), compute the objective increments of makespan Δ_C and total flowtime Δ_T.

2) If ($\Delta_C \geqslant 0$ and $\Delta_T \geqslant 0$) then Halt.

3) Generate a new sequence π_i' by applying a stir operation on π_i.

4) If ($\Delta_C \leqslant 0$ and $\Delta_T \leqslant 0$) then

Replace π_i with π_i'.

Else/ * (non-dominated solution) * /

If (| Archive | $< N_e$) then

Insert π_i' into the archive;

Else

Calculate $d(\pi_i')$, the crowding distance of sequence π_i'.

If $d(\pi_i') > d(\pi_i)$ then Replace π_i with π_i'.

The archive local search(ALS) is a procedure to update all the chromosomes in the archive. For any of the chromosomes, calculate Δ_{C1} and Δ_{T1} for two random positions. Call update procedure to update the archive. Then calculate ω_{C1} and ω_{T1} for two random positions. Also call update procedure to update the archive.

The obtained solutions by the above procedure can be further improved by the following ISOA (improving the solutions of the archive) procedure. To improve the efficiency of the algorithm, an objective increment method is adopted.

1) The size of the archive is initialized as N_e.

2) For $i = 1$ to N_e

① For $j = 2$ to n

For the selected chromosome π_i, consider the current position j and each possible slot of π_i. Then calculate Δ_{C1} and Δ_{T1}.

Call update procedure to update the archive.

② For $j = 2$ to $n - 1$

For the selected chromosome π_i, consider the current position j, $j + 1$ and the each possible slot of π_i. Then calculate Δ_{C2} and Δ_{T2}.

Call update procedure to update the archive.

③ For $j = 2$ to n

For the selected chromosome π_i, consider the current position j and each possible slot of π_i. Then calculate ω_{C1} and ω_{T1}.

Call update procedure to update the archive.

3) Halt.

Based on the above statements, the proposed multi-objective genetic algorithm is described below.

1) $N_p \leftarrow$the size of population, $N_e \leftarrow \varnothing$, itNum$\leftarrow$0, maxitNum$\leftarrow$the maximal iteration number, $p_c \leftarrow 1$, $p_m \leftarrow 0.1$.

2) Generate the initial population p_t by the algorithm given above.

3) Calculate the value of makespan and the total flowtime of each chromosome in the population.

4) Sort the initial population and calculate the values of the rank of each chromosome in the population.

5) If itNum $\geqslant$ maxitNum go to 6).

① Apply crossover and mutation to population and generate offspring population O_t with N_p individuals.

② Calculate the two objectives of each chromosome in O_t.

③ $C_t \leftarrow P_t \cup O_t$(size being $2N_p$). Sort population C_t and calculate the crowding distance of each chromosome in the C_t.

④ Select N_p chromosomes from C_t to generate p_{t+1}. Copy all the best solutions on the front 1 to the archive and remove those dominated solutions.

⑤ Call ALS to update the archive. itNum$\leftarrow$itNum + 1.

6) Apply ISOA procedure to improve the solutions of the archive.

7) The solutions in the archive are the Pareto-optimal solutions. Stop.

4 Experimental Results

The performance of the proposed multi-objective genetic algorithm is compared with three well-known heuristics, PAAH[7], PGA-ALS[8] and NSGA-II[13]. All algorithms have been coded in Java and executed on a Pentium(R)4, 3 GHz, and Window XP with 512 MB RAM.

All the test data used in all the algorithms come from the 50 benchmark problems (Ta071 to Ta120) given by Taillard[14]. The population size is set as 200 and the archive size is 100 with the number of iterations being 2 000. Also, to generate the same size solution set with other genetic algorithms, parameters of PAAH are initialized with $\alpha = 0$ and step = 0.006 25.

All the compared algorithms are measured on effectiveness and efficiency by optimal (ratio of obtained non-dominated solutions) and time (consumed computation time), respectively. Tab. 1 shows the results for the compared algorithms.

Tab. 1 shows that OPT of NSGA-II is zero in all the instances. In other words, the effectiveness of NSGA-II is the worst among the compared algorithms. The proposed algorithm obtains the best OPT for most instances. In fact, the other two algorithms outperform the proposal only in five instances(Ta075, Ta085, Ta109, Ta114 and Ta116). NSGA-II needs the least computation time. Time taken by PGA-ALS is similar to that of the proposal and Time of the proposal is usually less than that of PGA-ALS. PAAH is too time-con-

Tab. 1 Comparison results of the algorithms

Instances	PGA-ALS		NSGA-Ⅱ		PAAH		Proposed algorithm	
	Optimal	Time/s	Optimal	Time/s	Optimal	Time/s	Optimal	Time/s
Ta071	0.0	60.5	0.0	106.9	13.6	210.6	86.4	41.8
Ta072	0.0	61.6	0.0	101.9	12.5	210.6	87.5	39.5
Ta073	0.0	63.0	0.0	109.5	13.6	211.3	86.4	36.3
Ta074	0.0	72.5	0.0	103.9	16.7	211.3	83.3	41.3
Ta075	0.0	60.9	0.0	108.9	57.1	211.3	42.9	31.8
Ta076	10.0	54.1	0.0	94.0	0.0	208.8	90.0	37.2
Ta077	0.0	55.5	0.0	103.4	16.7	208.8	83.3	39.2
Ta078	0.0	50.3	0.0	101.6	11.1	160.6	88.9	35.1
Ta079	14.3	54.3	0.0	101.9	21.4	208.8	64.3	37.3
Ta080	0.0	55.1	0.0	101.3	18.2	208.8	81.8	32.8
Ta081	25.0	56.1	0.0	103.7	18.8	218.1	56.3	38.3
Ta082	68.8	57.4	0.0	104.7	6.3	218.8	25.0	36.7
Ta083	2.7	53.2	0.0	103.9	8.1	219.4	89.2	37.2
Ta084	0.0	68.1	0.0	105.8	10.0	216.3	90.0	35.6
Ta085	69.2	61.3	0.0	99.2	23.1	216.9	7.7	34.4
Ta086	0.0	69.5	0.0	105.4	10.0	216.3	90.0	45.3
Ta087	33.3	61.2	0.0	101.3	9.5	216.9	57.1	32.3
Ta088	10.5	54.3	0.0	104.2	15.8	216.3	73.7	32.7
Ta089	36.4	58.5	0.0	100.1	27.3	216.3	36.4	36.5
Ta090	22.2	54.6	0.0	105.7	33.3	216.9	44.4	40.8
Ta091	13.6	100.5	0.0	113.7	4.5	1 862.5	81.8	100.1
Ta092	10.5	101.8	0.0	110.6	10.5	1 861.9	78.9	81.8
Ta093	0.0	94.4	0.0	116.2	38.5	1 851.3	61.5	74.5
Ta094	5.6	100.5	0.0	111.1	16.7	1 872.5	77.8	86.7
Ta095	0.0	101.7	0.0	114.4	21.4	1 871.3	78.6	77.1
Ta096	30.8	101.9	0.0	108.5	15.4	1 866.9	53.8	80.1
Ta097	0.0	105.2	0.0	113.2	5.6	1 866.9	94.4	99.9
Ta098	13.0	102.4	0.0	113.9	13.0	1 866.9	73.9	80.8
Ta099	0.0	102.4	0.0	109.7	10.0	1 852.5	90.0	70.4
Ta100	11.1	108.6	0.0	114.3	22.2	1 852.5	66.7	83.0
Ta101	0.0	115.0	0.0	118.1	20.0	1 923.1	80.0	73.4
Ta102	22.7	116.4	0.0	117.6	9.1	1 921.3	68.2	94.4
Ta103	0.0	111.3	0.0	117.6	26.7	1 891.3	73.3	78.1
Ta104	0.0	119.5	0.0	118.5	20.0	1 925.6	80.0	80.6
Ta105	0.0	111.5	0.0	120.9	14.3	1 868.1	85.7	76.7
Ta106	0.0	115.3	0.0	118.1	46.2	1 918.1	53.8	84.5
Ta107	13.0	120.9	0.0	120.9	8.7	1 927.5	78.3	91.1
Ta108	0.0	107.1	0.0	126.2	30.0	1 996.3	70.0	75.2
Ta109	52.9	112.9	0.0	127.3	23.5	2 068.8	23.5	92.9
Ta110	0.0	126.2	0.0	126.9	33.3	2 031.3	66.7	85.6
Ta111	30.4	604.7	0.0	198.6	8.7	33 065.0	60.9	685.8
Ta112	21.4	599.9	0.0	194.1	17.9	33 300.0	60.7	664.8
Ta113	42.3	602.4	0.0	194.8	7.7	31 873.8	50.0	672.8
Ta114	44.0	599.7	0.0	206.9	20.0	31 903.1	36.0	700.8
Ta115	44.9	600.2	0.0	208.2	8.2	32 125.0	46.9	774.8
Ta116	13.3	590.7	0.0	197.2	60.0	32 154.4	26.7	678.5
Ta117	36.0	599.4	0.0	203.9	0.0	32 205.0	64.0	718.7
Ta118	13.9	604.5	0.0	204.3	5.6	32 186.3	80.6	704.2
Ta119	12.2	608.6	0.0	207.5	0.0	31 785.6	87.8	710.8
Ta120	13.6	604.9	0.0	200.0	27.3	31 945.4	59.1	683.4

suming. For example, it takes NSGA-Ⅱ only 27.3 s; PGA-ALS takes 604.9 s and the proposed algorithm 683.4 s, while PAAH takes 31 945.4 s for Ta120.

5 Conclusion

This paper presents a genetic algorithm based on objective increments for solving no-wait flow shop scheduling problems with the objective of the minimization of the makespan and total flowtime. Non-dominated classification and crowding distance, coupled with the use of the elitist solution strategy, are proposed. A fast Pareto sorting approach and two effective bi-criteria local search procedures based on objective increments are introduced. Experimental results indicate that the proposed algorithm outperforms the other compared algorithms in most instances. The proposed algorithm obtains the best ratio of the Pareto solution in 45 out of the 50 compared instances. Though PAAH is the best among the existing algorithms, it outperforms the proposed algorithm in only four instances while it consumes more than 40 times computation time than the proposed algorithm. Therefore, the proposed algorithm is more suitable for the considered problems.

References

[1] Gangadharan R, Rajendran C. Heuristic algorithms for scheduling in the no-wait flow shop [J]. *International Journal of Production Economics*, 1993, **32**(3): 285 − 290.

[2] Rajendran C. A no-wait flow shop scheduling heuristic to minimize makespan [J]. *Journal of the Operational Research Society*, 1994, **45**(4): 472 − 478.

[3] Aldowiasan T, Allahverdi A. New heuristics for no-wait flow shops to minimize makespan [J]. *Computers and Operations Research*, 2003, **30**(8): 1219 − 1231.

[4] Grabowski J, Pempera J. Some local search algorithms for no-wait flow-shop problem with makespam criterion [J]. *Computers and Operations Research*, 2005, **32**(8): 2197 − 2213.

[5] Li Xiaoping, Wu Cheng. Heuristic for no-wait flow shops with makespan minimization based on total idle-time increments [J]. *Science in China, Series F*: *Information Sciences*, 2008, **51**(7): 896 − 909.

[6] Li Xiaoping, Wang Qian, Wu Cheng. Heuristic for no-wait flow shops with makespan minimization [J]. *International Journal of Production Research*, 2008, **46**(9): 2519 − 2530.

[7] Allahverdi A, Aldowaisan T. No-wait flow shops with bicriteria of makespan and total completion time [J]. *Journal of the Operational Research Society*, 2002, **53**(9): 1004 − 1015.

[8] Allahverdi A, Aldowaisan T. No-wait flow shops with bicriteria of makespan and maximum lateness [J]. *European Journal of Operational Research*, 2004, **152**(1): 132 − 147.

[9] Reza Tavakkoli-Moghaddam, Alireza Rahimi-Vahed, Ali Hossein Mirzaei. A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness [J]. *Information Sciences*, 2007, **177**: 5072 − 5090.

[10] Jones D F, Mirrazavi S K, Tamiz M. Multiobjective meta-heuristics: an overview of the current state-of-the-art [J]. *European Journal of Operational Research*, 2002, **137**(1): 1 − 9.

[11] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-Ⅱ [J]. *IEEE Transactions on Evolutionary Computation*, 2002, **6**(2): 182 − 197.

[12] Taillard E. Benchmarks for basic scheduling problems [J]. *European Journal of Operational Research*, 1993, **64**(2): 278 − 285.

[13] Nawaz M, Enscore E E Jr, Ham I. A heuristic algorithm for the *m*-machine, *n*-job flow shop sequencing problem [J]. *Omega*, 1983, **11**(1): 91 − 95.

[14] Allahverdi A, Aldowaisan T. New heuristics to minimize total completion time in *m*-machine flow shops [J]. *International Journal of Production Economics*, 2002, **77**(1): 71 − 83.

最小化最长完工时间和总完工时间的无等待流水调度混合进化算法

廖小平[1] 刘有根[2] 李小平[2]

([1] 河海大学计算机及信息工程学院,南京 210098)

([2] 东南大学计算机科学与工程学院,南京 210096)

摘要:针对 NP 难的最小化最长完工时间和总完工时间无等待流水双目标调度优化问题,分析相应的目标增量性质,提出用非支配划分方法将种群划分为具有不同优先级的 Pareto 面以提高搜索解的效率. 除建立拥挤距离的概念和最优解策略外,提出 2 个基于目标增量的双目标局部搜索过程,以提高搜索解的性能. 根据得到的性质和方法,构建一个求解所考虑问题的混合进化算法,并与目前最好的算法比较. 实验结果表明所提出的算法在性能上优于所比较算法,并具有较高的效率.

关键词:无等待流水调度;目标增量;最长完工时间;总完工时间;进化算法

中图分类号:TP278