A Multiobjective State Transition Algorithm for Single Machine Scheduling

Xiaojun Zhou, Samer Hanoun, David Yang Gao, and Saeid Nahavandi

Abstract In this paper, a discrete state transition algorithm is introduced to solve a multiobjective single machine job shop scheduling problem. In the proposed approach, a non-dominated sort technique is used to select the best from a candidate state set, and a Pareto archived strategy is adopted to keep all the non-dominated solutions. Compared with the enumeration and other heuristics, experimental results have demonstrated the effectiveness of the multiobjective state transition algorithm.

Keywords Discrete state transition algorithm • Multiobjective optimization • Single machine scheduling

1 Introduction

The multiobjective optimization is encountered in many real-world applications [1]. For a specific policy, the decision maker may find it advantageous for one goal but disadvantageous for others. A traditional way to deal with this issue is to impose a priori preference reflecting the relative importance of different objectives; however, the final solution just indicates a decision maker's satisfaction, and it might be dissatisfactory for other decision makers.

To ameliorate the problem, the concept of *Pareto optimality* and other relevant concepts are introduced. These are defined as follows:

(1) *Pareto dominance*: A feasible solution $\boldsymbol{x} = (x_1, \cdots, x_n)$ is said to Pareto dominate another feasible solution $\boldsymbol{y} = (y_1, \cdots, y_n)$, denoted as $\boldsymbol{x} \prec \boldsymbol{y}$, if

$$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}), \forall i \in \{1, \cdots, k\}, \text{ and } \exists j \in \{1, \cdots, k\}, f_j(\boldsymbol{x}) < f_j(\boldsymbol{y}), \quad (1)$$

where $f_i(\boldsymbol{x})$ is the ith objective function, k is the number of objectives.

X. Zhou (✉) • D.Y. Gao
School of Science, Information Technology and Engineering,
University of Ballarat, Ballarat, VIC 3353, Australia
e-mail: tiezhongyu2010@gmail.com; d.gao@ballarat.edu.au

S. Hanoun • S. Nahavandi
Centre for Intelligent Systems Research, Deakin University, Geelong, VIC, Australia
e-mail: samer.hanoun@deakin.edu.au; saeid.nahavandi@deakin.edu.au

D. Gao et al. (eds.), *Advances in Global Optimization*, Springer Proceedings in Mathematics & Statistics 95, DOI 10.1007/978-3-319-08377-3_9

(2) *Pareto optimality*: A feasible solution $\boldsymbol{x}^*$ is said to be Pareto optimal if and only if

$$\neg\exists \boldsymbol{x} \in S, \boldsymbol{x} \prec \boldsymbol{x}^*, \tag{2}$$

where S is the feasible space.

(3) *Pareto optimal set*: The Pareto optimal set, denoted as P^*, is defined by

$$P^* = \{\boldsymbol{x}^* \in S | \neg\exists \boldsymbol{x} \in S, \boldsymbol{x} \prec \boldsymbol{x}^*\}. \tag{3}$$

(4) *Pareto front*: The Pareto front, denoted as Pf^*, is defined by

$$Pf^* = \{(f_1(\boldsymbol{x}^*), \cdots, f_k(\boldsymbol{x}^*)) | \boldsymbol{x}^* \in P^*\}. \tag{4}$$

The introduction of *Pareto optimality* allows us to find a set of Pareto optimal solutions simultaneously, independent of the decision maker's priori preference.

In the past few decades, evolutionary-based and nature-inspired multiobjective optimization techniques have drawn considerable attention for scheduling problems [2–7]. In this paper, we introduce a recently new heuristics called state transition algorithm [8–11] as the basic search engine for the multiobjective optimization. A non-dominated sort approach is used to select the best from a candidate state set, and the best state is stored using a Pareto archive strategy. Experimental results have testified the effectiveness of the proposed algorithm.

2 Problem Description

In the field of joinery manufacturing, jobs with similar materials can be scheduled together to minimize the amount of materials used; therefore, reducing the cost.

For example, based on the cost savings matrix shown in Table 1, pairing Job1 and Job2 will provide saving in the cost equivalent to 4 units.

Additionally, based on the jobs' processing times and due dates as shown in Table 2, and for any given sequence and pair of jobs, not only the total cost saving C is affected but also the total tardiness time T, which is calculated as:

Table 1 The cost savings matrix for five jobs having the same material

	Job1	Job2	Job3	Job4	Job5
Job1	0	4	2.64	4.08	3.9
Job2	4	0	3.64	4.72	4.23
Job3	2.64	3.64	0	2.65	2.87
Job4	4.08	4.72	2.65	0	3.84
Job5	3.9	4.23	2.87	3.84	0

Table 2 Due dates and processing times for a set of five jobs

Job	Due date (days)[a]	Processing Time (h)
Job1	8	17:40
Job2	2	24:00
Job3	11	19:20
Job4	3	25:00
Job5	3	14:40

[a]Number of operational hours = 8 h per day

$$T = \sum_{j=1}^{n} \max\{0, c_j - d_j\} \tag{5}$$

where c_j and d_j are the completion time and the due time of job j, respectively.

The goal of this paper is to determine the optimal sequence with pairing, in order to maximize the total cost savings and minimize the total tardiness time.

It is obvious that finding the permutation of the sequence $\{1, 2, \cdots, n\}$ with pairing becomes a solution to the multiobjective single machine scheduling problem; however, not without the necessity to discuss the number of pairs for any fixed sequence of jobs.

Given a sequence $s = (1, 2, \cdots, n)$, for $n = 3$, we have two possible pairing options (1-2)-3 and 1-(2-3); for $n = 4$, we have two possible pairing options (1-2)-(3-4) and 1-(2-3)-4, as pairing options (1-2)-3-4 and 1-2-(3-4) are discarded; for $n = 5$, we have three possible options (1-2)-(3-4)-5, (1-2)-3-(4-5) and 1-(2-3)-(4-5), as options (1-2)-3-4-5, 1-2-(3-4)-5, 1-2-3-(4-5) and 1-(2-3)-4-5 are discarded.

If $P_1(n)$ denotes the number of pairs with the first two jobs pairing, and $P_2(n)$ denotes the complement of $P_1(n)$, then we have the following theorem:

Theorem 1.

$$P_1(n+1) = P(n-1), P_2(n+1) = P_1(n), n \geq 3 \tag{6}$$

where $P(n) = P_1(n) + P_2(n)$ *is the total number of pairs. For example,* $P_1(2) = 1$, $P_2(2) = 0$, $P_1(3) = 1, P_2(3) = 1$, $P_1(4) = 1, P_2(4) = 1$, $P_1(5) = 2, P_2(5) = 1$, *we have* $P_1(4) = P(2), P_2(4) = P_1(3)$, $P_1(5) = P(3), P_2(5) = P_1(4)$.

Figure 1 shows the growth trend of the number of pairs with the sequence size; however, only small size job scheduling problems are considered in this study. Considering that $P(10) = 12 \ll 10! = 3{,}628{,}800$, a complete enumeration approach is used for pairing and only the permutation of a sequence is focused.

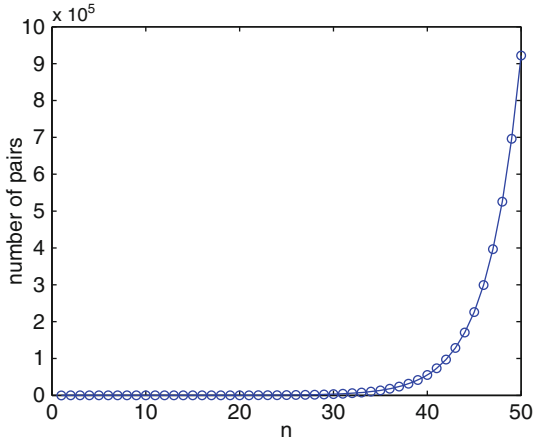

Fig. 1 Growth trend relative to the sequence size

3 Discrete State Transition Algorithm

In the case a solution to a specific optimization problem is described as a state, then the transformation to update the solution becomes a state transition. Without loss of generality, the unified form of discrete state transition algorithm can be described as:

$$\begin{cases} \boldsymbol{x}_{k+1} = A_k(\boldsymbol{x}_k) \bigoplus B_k(\boldsymbol{u}_k) \\ y_{k+1} = f(\boldsymbol{x}_{k+1}) \end{cases}, \tag{7}$$

where $\boldsymbol{x}_k \in \mathscr{X}^n$ stands for a current state, corresponding to a solution of a specific optimization problem; $\boldsymbol{u}_k$ is a function of $\boldsymbol{x}_k$ and historical states; $A_k(\cdot)$, $B_k(\cdot)$ are transformation operators, which are usually state transition matrixes; $\bigoplus$ is an operation, which is admissible to operate on two states; and f is the cost function or evaluation function.

The following three transformation operators are defined to permute current solution [10]:

(1) Swap Transformation

$$\boldsymbol{x}_{k+1} = A_k^{swap}(m_a)\boldsymbol{x}_k, \tag{8}$$

where $A_k^{swap} \in \mathbb{R}^{n\times n}$ is the swap transformation matrix, m_a is the swap factor, a constant integer used to control the maximum number of positions to be exchanged, while the positions are random. Figure 2 shows an example of the swap transformation with $m_a = 2$.

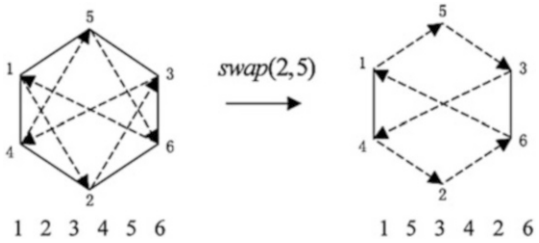

Fig. 2 Illustration of the swap transformation

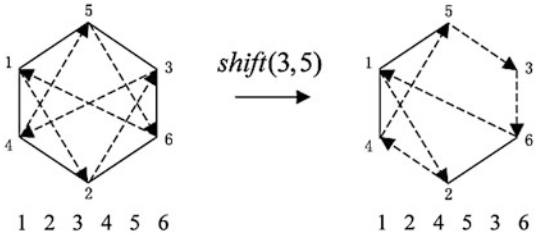

Fig. 3 Illustration of the shift transformation

(2) Shift Transformation

$$\boldsymbol{x}_{k+1} = A_k^{shift}(m_b)\boldsymbol{x}_k, \tag{9}$$

where $A_k^{shift} \in \mathbb{R}^{n\times n}$ is the shift transformation matrix, m_b is the shift factor, a constant integer used to control the maximum length of consecutive positions to be shifted. Note that both the selected position to be shifted after and positions to be shifted are chosen randomly. Figure 3 shows an example of the shift transformation with $m_b = 1$.

(3) Symmetry Transformation

$$\boldsymbol{x}_{k+1} = A_k^{sym}(m_c)\boldsymbol{x}_k, \tag{10}$$

where $A_k^{sym} \in \mathbb{R}^{n\times n}$ is the symmetry transformation matrix, m_c is the symmetry factor, a constant integer used to control the maximum length of subsequent positions as center. Note that both the component before the subsequent positions and consecutive positions to be symmetrized are created randomly. Figure 4 shows an example of the symmetry transformation with $m_c = 0$.

4 Pareto Archived Strategy Based on DSTA

In state transition algorithm, the times of transformation are called search enforcement (*SE*); as a result, after each transformation operator, a candidate state set S is generated.

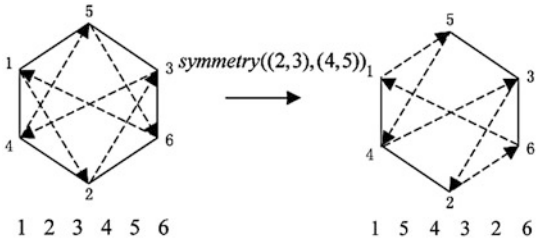

Fig. 4 Illustration of symmetry transformation

4.1 Non-dominated Sort

We use a sorting approach similar to the fast-non-dominated-sort proposed in [12], described as follows:

```
for each s ∈ S do
    n_s ← 0
    for each t ∈ S do
        if t ≺ s then
            n_s ← n_s + 1
        end if
    end for
end for
```

where n_s is the domination count, representing the number of solutions dominating solution s. After the non-dominated sort, the state with the least count will be stored as incumbent *best* for the next transformation operator.

4.2 Pareto Archived Strategy

We adopt a simple Pareto archived strategy to select current *best* as follows:

```
for each A_i ∈ 𝒜 do
    if best ≺ A_i then
        𝒜 ← 𝒜 − A_i
    else if A_i ≺ best then
        𝒜 ← 𝒜
    else
        𝒜 ← 𝒜 ⋃ best
    end if
end for
```

where $\mathscr{A}$ is the archive keeping all non-dominated solutions.

4.3 Pseudocodes of the Proposed Algorithm

The core procedure of the proposed algorithm can be outlined in pseudocodes:

1: **repeat**
2: $State \leftarrow operator(best, SE, n)$
3: $best \leftarrow update\_best(best, SE, n, data)$
4: $Paretoset \leftarrow update\_archive(Paretoset, best)$
5: **until** the maximum number of iterations is met

where $State$ is the state set; $operator$ stands for the three transformation operators, which are carried out sequentially; $update\_best$ is corresponding to the non-dominated sort, and $update\_archive$ corresponds to the Pareto archived strategy. The $data$ is the known information (cost saving matrix, due dates, and processing times) about a specific scheduling problem.

5 Experimental Results

In order to test the performance of the proposed multiobjective state transition algorithm, two typical examples are used for comparison. In the following experiments, $SE = 20, m_a = 2, m_b = 1, m_c = 0$ are adopted for parameter settings. The maximum number of iterations for are 100 and 1,000, respectively, for the two examples. The known data for Example 1, 2 are given in Tables 1 and 2, Tables 3 and 4, respectively, and the corresponding results can be found in Tables 5 and 6. It is worth noting that the pairing methodology used with complete enumeration and Cuckoo Search (CS) is based on a greedy approach by first selecting the pair that produces the highest cost savings, and then repeating the same procedure for the remaining set of pairs in the sequence [7]. We can find that for Example 1,

Table 3 The cost savings matrix for ten jobs having the same material

	Job1	Job2	Job3	Job4	Job5	Job6	Job7	Job8	Job9	Job10
Job1	0	2.73	2.1	2.16	2.66	3.6	2.46	2.7	2.46	2.8
Job2	2.73	0	2	1.6	4.3	3.69	2.3	3.5	2.76	3.6
Job3	2.1	2	0	1.4	3.51	3.33	2.52	3.68	2.52	2.46
Job4	2.16	1.6	1.4	0	2.17	2.32	2.72	3.04	2.04	2.97
Job5	2.66	4.3	3.51	2.17	0	3.6	4.05	4.41	2.7	2.64
Job6	3.6	3.69	3.33	2.32	3.6	0	2.58	4.7	3.44	2.94
Job7	2.46	2.3	2.52	2.72	4.05	2.58	0	2.6	2.88	2.82
Job8	2.7	3.5	3.68	3.04	4.41	4.7	2.6	0	3.64	3.57
Job9	2.46	2.76	2.52	2.04	2.7	3.44	2.88	3.64	0	3.76
Job10	2.8	3.6	2.46	2.97	2.64	2.94	2.82	3.57	3.76	0

Table 4 Due dates and processing times for a set of ten jobs

Job	Due date (days)	Processing time (h)
Job1	11	14:00
Job2	2	18:00
Job3	13	15:00
Job4	14	8:20
Job5	11	17:20
Job6	9	16:00
Job7	4	19:40
Job8	6	23:20
Job9	10	20:00
Job10	10	19:20

Table 5 Comparison results for the set of jobs presented in Tables 1 and 2

Approach	Optimal solutions	T	C
Complete Enumeration	(2-5)-(1-4)-3	13	8.31
	(5-2)-(1-4)-3	13	8.31
	(2-5)-(4-1)-3	13	8.31
	(2-4)-(5-1)-3	15	8.62
CS [7]	(2-5)-(1-4)-3	13	8.31
	(5-2)-(1-4)-3	13	8.31
	(2-5)-(4-1)-3	13	8.31
	(2-4)-(5-1)-3	15	8.62
STA	(5-2)-(1-4)-3	13	8.31
	(2-5)-(1-4)-3	13	8.31
	(2-5)-(4-1)-3	13	8.31
	(5-2)-(4-1)-3	13	8.31
	(2-4)-(5-1)-3	**15**	**8.62**

STA obtained a solution which can dominate the optimal solutions by enumeration and CS. From both examples, it is easy to find that some additional optimal solutions are achieved by STA, as indicated by the bold values.

6 Conclusion

A multiobjective state transition algorithm is presented for a single machine job shop scheduling problem. In this paper, a complete enumeration approach is used for pairing the jobs in a fixed sequence. Compared with a greedy-based approach

Table 6 Comparison results for the set of jobs presented in Tables 3 and 4

Approach	Optimal solutions	T	C
Complete Enumeration	(5-7)-(2-6)-(1-3)-(4-10)-(8-9)	39	16.45
	(5-7)-(2-6)-(1-3)-(4-8)-(10-9)	40	16.64
	(5-7)-(2-6)-(1-3)-(4-8)-(9-10)	40	16.64
	(5-7)-(2-6)-(1-4)-(3-8)-(10-9)	41	17.34
	(5-7)-(2-6)-(1-4)-(3-8)-(9-10)	41	17.34
	(5-2)-(7-4)-(6-1)-(3-8)-(10-9)	43	18.06
	(5-2)-(7-4)-(6-1)-(3-8)-(9-10)	43	18.06
	(2-5)-(7-4)-(6-1)-(3-8)-(10-9)	43	18.06
	(2-5)-(7-4)-(6-1)-(3-8)-(9-10)	43	18.06
CS [7]	2-(7-5)-(6-1)-3-(4-10)-(8-9)	39	14.26
	(5-7)-(2-6)-(1-3)-(4-8)-(9-10)	40	16.64
	(5-7)-(2-6)-(1-4)-(3-8)-(10-9)	41	17.34
	(2-5)-(7-4)-(6-1)-(3-8)-(10-9)	43	18.06
	(2-5)-(7-4)-(6-1)-(3-8)-(9-10)	43	18.06
	(5-2)-(7-4)-(6-1)-(3-8)-(10-9)	43	18.06
STA	(5-7)-(2-6)-(1-3)-(4-10)-(8-9)	39	16.45
	(5-7)-(2-6)-(1-3)-(4-10)-(9-8)	**39**	**16.45**
	(5-7)-(2-6)-(1-3)-(4-8)-(10-9)	40	16.64
	(5-7)-(2-6)-(1-3)-(4-8)-(9-10)	40	16.64
	(5-7)-(2-6)-(1-4)-(3-8)-(10-9)	41	17.34
	(5-7)-(2-6)-(1-4)-(3-8)-(9-10)	41	17.34
	(5-2)-(7-4)-(6-1)-(3-8)-(10-9)	43	18.06
	(5-2)-(7-4)-(6-1)-(3-8)-(9-10)	43	18.06
	(2-5)-(7-4)-(6-1)-(3-8)-(10-9)	43	18.06
	(2-5)-(7-4)-(6-1)-(3-8)-(9-10)	43	18.06

used with both the complete enumeration method and the CS, experimental results show the effectiveness of the proposed algorithm in obtaining the true set of all Pareto optimal solutions.

Acknowledgements This research work is conducted between Deakin University and Ballarat University under the Collaboration Research Network (CRN) initiative. The problem studied in this paper is related to the Australian Research Council (ARC) linkage project number LP0991175.

References

1. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Struct. Multidiscip. Optim. **26**(6), 369–395 (2004)
2. Coello, C.A.C.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl. Inf. Syst. **1**(3), 129–156 (1999)

3. Lei, D.M.: A Pareto archive particle swarm optimization for multi-objective job shop scheduling. Comput. Ind. Eng. **54**(4), 960–971 (2008)
4. Al-Anzi, F.S., Allahverdi, A.: A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. Eur. J. Oper. Res. **182**(1), 80–94 (2007)
5. Xia, W.J., Wu, Z.M.: An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput. Ind. Eng. **48**(2), 409–425 (2005)
6. Hanoun, S., Nahavandi, S., Kull, H.: Pareto archived simulated annealing for single machine job shop scheduling with multiple objectives. In: The Sixth International Multi-Conference on Computing in the Global Information Technology (ICCGI), pp. 99–104 (2011)
7. Hanoun, S., Creighton, D., Nahavandi, S. Kull, H.: Solving a multiobjective job shop scheduling problem using Pareto Archived Cuckoo Search. In: Proceedings of the 2012 17th IEEE International Conference on Emerging Technologies & Factory Automation, pp. 1–8 (2012)
8. Zhou, X.J., Yang, C.H., Gui, W.H.: Initial version of state transition algorithm. In: International Conference on Digital Manufacturing and Automation (ICDMA), pp. 644–647 (2011)
9. Zhou, X.J., Yang, C.H., Gui, W.H.: A new transformation into state transition algorithm for finding the global minimum. In: International Conference on Intelligent Control and Information Processing (ICICIP), pp. 674–678 (2011)
10. Yang, C.H., Tang, X.L., Zhou, X.J., Gui, W.H.: State transition algorithm for traveling salesman problem. In: the Proceedings of the 31st Chinese Control Conference (CCC), pp. 2481–2485 (2012)
11. Zhou, X.J., Yang, C.H., Gui, W.H.: State transition algorithm. J. Ind. Manag. Optim. **8**(4), 1039–1056 (2012)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)