A Comparison of Two Chromosome Representations Used in ACO to Solve A Scheduling Problem

Chen-Fu Chen and Muh-Cherng Wu
Department of Industrial Engineering and Management
National Chiao Tung University, Taiwan

E-mail : mcwu@mail.nctu.edu.tw

Abstract - This research attempts to justify a research claim—developing new chromosome (solution) representation schemes might be able to improve the performance of existing meta-heuristic algorithms. A scheduling problem, called permutation manufacturing-cell flow shop (PMFS), is used to justify the research claim. We compare the effectiveness of two chromosome representation schemes (S_{old} and S_{new}) while they are embedded in a particular meta-heuristic algorithm, ACO (ant colony optimization), to solve the scheduling problem. Denote a tested meta-heuristic algorithm by X_Y, where X represents an algorithmic mechanism and Y represents a chromosome representation. Experiment results indicate that ACO_S_{new} outperforms ACO_S_{old}. These findings advocate the attention of developing new chromosome representations in the research of meta-heuristic algorithms.

Keywords: Chromosome Representation, Ant Colony Optimization, Scheduling,

I. INTRODUCTION

Meta-heuristic algorithms have been widely used in solving complex space-search problems. A meta-heuristic algorithm is essentially composed of two parts: (1) *algorithmic mechanism* and (2) solution representation (also called *chromosome representation*). Most prior studies focused on how to enhance algorithmic mechanisms. Yet, the investigation of how to improve *chromosome representation* has been rarely noticed.

This paper advocates that developing new chromosome representation might be very important in the application of meta-heuristic algorithms. Such a research claim is justified by solving a scheduling problem, called permutation manufacturing-cell flow shop (PMFS). We compare the effectiveness of two chromosome representation schemes (S_{old} and S_{new}) while they are embedded in a particular meta-heuristic algorithm, ACO (ant colony optimization), to solve the scheduling problem.

A tested meta-heuristic algorithm is denoted by *X_Y*, where *X* represents an algorithmic mechanism and *Y* represents a chromosome representation. Experiment results indicate that *ACO_*S_{new} outperforms *ACO_*S_{old}. This finding sheds a light on the track of developing new chromosome representations in the application of meta-heuristic algorithms.

II. PROBLEM AND LITERATURE REVIEW

The concerned scheduling context is a flow shop with four distinct features. Firstly, the shop follows a *family-based scheduling paradigm*. That is, all jobs are pre-grouped into various families. Jobs within a family are similar in process requirements and needs no setup time. In contrast, significant setup times are required if the two jobs are of different families.

Secondly, each job is *individually transported*. Once a job completes its operation at a stage, it is *immediately* and *individually* transported to the buffer of the next stage. Thirdly, its setup times among families are *sequence-dependent*. Fourthly, the shop is a *permutation flow* shop with *no breakdown*. That is, while traveling through each stage of the flow shop, the job sequence within each family and the sequence among families keep the same. Each machine is so reliable and involves no breakdown in the scheduling horizon.

The scheduling problem leads to two sequencing decisions: *among-family* sequencing and *within-family* sequencing. Within-family sequencing denotes the sequence of jobs within each family. In contrast, among-family sequencing denotes the sequence of families.

Such a scheduling problem is NP-hard in the strong sense [1,2]. Exact optimization procedures are not suitable for solving realistic–size problems due to the tremendous requirement of computational efforts. Approximation algorithms developed for solving such a scheduling problem are either heuristic approach or meta-heuristic approach [1,2,3].

III. CHROMOSOME REPRESENTATIONS

As stated, we attempt to use two chromosome representation schemes to solve the scheduling problem. One (called S_{old}) is adopted from prior studies [1,2], and the other (called S_{new}) is developed by us. Before introducing S_{old} and S_{new}, readers are reminded that the scheduling problem includes two decisions—the job sequence within each family and the sequence among families. Therefore, a chromosome representation must be eligible for accommodating the two sequencing decisions.

A. S_{old} Chromosome Representation

To accommodate the two sequencing decisions, S_{old} chromosome representation has two distinct features: *clustering* and *multiple-segments*. Consider a scheduling context with n jobs (i.e., $J_1, J_2, \ldots, J_n$) that have been grouped into q job families (i.e., $f_1, f_2, \ldots, f_q$). The *multiple-segments* feature denotes that a chromosome includes q+1 segments. The clustering feature denotes that the q+1 segments are categorized into two clusters. The first cluster involves *only one* segment, which represents the sequence among the q families. The second cluster involves q segments, each of which represents the job sequence within a particular family.

S_{old} chromosome representation is explained by an

example problem with 10 jobs and 3 families. See Fig. 1, the chromosome involves two clusters. The first cluster comprises only one segment, which implies that the sequence among families is $f_3 \rightarrow f_2 \rightarrow f_1$. The second cluster comprises three segments; the first one implies that family f_1 comprises 3 jobs and their processing sequence is $J_1 \rightarrow J_3 \rightarrow J_2$. Accordingly, the second and the third segments respectively represent the job sequence within family f_2 and f_3.

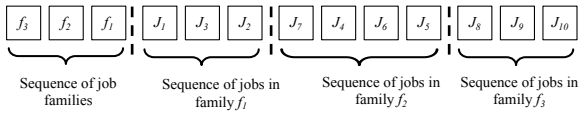

Fig. 1. S_{old} chromosome representation.

B. S_{new} Chromosome Representation

In contrast, S_{new} has two other distinct features: *single segment* and *decoding mechanism*. That is, an S_{new} chromosome is a *single segment* which comprises a sequence of jobs. By *decoding* the S_{new} chromosome, we can obtain the two scheduling decisions—job sequence within each family and the sequence among families.

Consider a scheduling context with n jobs (i.e., $J_1, J_2, \ldots, J_n$) that are grouped into q families (i.e., $f_1, f_2, \ldots, f_q$). An S_{new} chromosome is a *single segment* comprising a sequence of the n jobs. To obtain the among-family sequence decision, the decoding mechanism is designed by traveling through the S_{new} chromosome (i.e., the job sequence) and reviewing each job's affiliated family. While traveling through the chromosome, we would observe that a particular family may appear several times. Then, each family only when it appears the first time is recorded. As a result, the recorded sequence of these families denotes the *among-family* sequence. Alternatively, for jobs within a particular family, their sequencing decision is determined by each job's appearing sequence in the S_{new} chromosome.

As shown in Fig. 2, an example with 10 jobs and 3 families is used to explain S_{new} chromosome representation scheme. The S_{new} chromosome indicates that the sequence of the first four jobs is $J_8 \rightarrow J_7 \rightarrow J_4 \rightarrow J_1$ and their affiliated family sequence is $f_3 \rightarrow f_2 \rightarrow f_2 \rightarrow f_1$. This implies that the resulting family sequence is $f_3 \rightarrow f_2 \rightarrow f_1$. By conforming to the job precedence relationships of the S_{new} chromosome, the job sequence within each family can be easily obtained. Namely, the job sequence within family f_3 is $J_8 \rightarrow J_9 \rightarrow J_{10}$, that within f_2 is $J_7 \rightarrow J_4 \rightarrow J_6 \rightarrow J_5$, and that within f_1 is $J_1 \rightarrow J_3 \rightarrow J_2$.

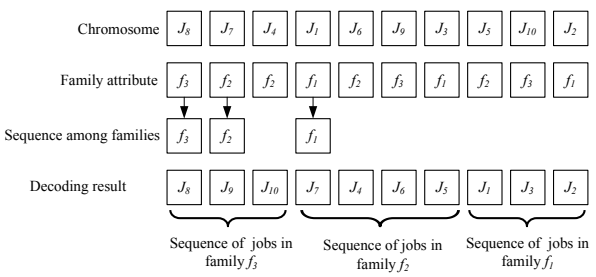

Fig. 2. S_{new} chromosome representation.

IV. ACO ALGORITHMS

A. ACO_S_{new} Algorithm

The purpose of ACO_S_{new} algorithm is to obtain a good job sequence. To do so, finding a good job sequence is considered as a traveling salesman problem (TSP). Consider a problem with N jobs to be scheduled. Given a virtual start node (say, Node_0), we have to travel through N existing nodes. The purpose is to find a good traveling route (i.e., a traveling sequence of these N nodes), which also denotes a good chromosome (a sequence of N jobs).

To illustrate the generation of a traveling route, we model the N+1 nodes (including Node_0) as a network. In the network, there exists a path between any two nodes as shown in Fig. 3. Each path has an *aggregate attribute* (λ_{ij}), which denotes the degree of preference for traveling through the path. The larger is λ_{ij}, the higher is the probability of traveling through the path. Noticeably, λ_{ij} is the combination of two other attributes τ_{ij} and η_{ij} that shall be explained later.

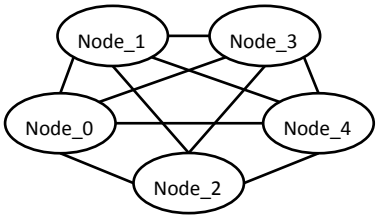

Fig. 3. An example of ACO network.

Given such an ACO network, the procedure of generating a traveling route is explained below. Consider that a salesman is now at node i, and there are m remaining nodes to be traveled through. This implies that the salesman has m alternatives (paths) for choosing the next node. In ACO_S_{new}, we use the tournament method [4] to select the next node; that is, the probability of choosing node k as the next node is $p_{ik} = \lambda_{ik} / (\sum_{j=1}^{j=m} \lambda_{ij})$. Suppose there are S salesmen designated to travel the network, we *very likely* generate S different traveling routes, due to the *probabilistic* nature of the tournament method. Such a salesman is also called an *ant*, which explains why such an algorithm is named ACO (*ant* colony optimization).

In ACO_S_{new}, we denote the updated ACO network by W_t, where W_0 represents the initial network and t represents total number of network updating. Given W_t, the idea for generating W_{t+1} is based on the feedback of multiple travelers. That is, we first send S ants (salesmen) to travel through the W_t network. This yields S different traveling routes; each route is very likely with different performance (makespan). Then, the performances of these S traveling routes are aggregated to change λ_{ij} for obtaining W_{t+1} as explained below.

As stated, λ_{ij} is the combination of two other attributes τ_{ij} and η_{ij}. The attribute η_{ij} is a *static* attribute, which denotes the relative importance of path ij and shall not be changed during the update of W_t. In ACO_S_{new}, we define $\eta_{ij} = 1/(s_{ij} + p_j)$ where p_j denotes the processing time of job j, and s_{ij} denotes the

setup time required for the transition from processing job *i* to job *j*. That is, if jobs *i* and *j* are of the same family, then $s_{ij} = 0$.

In contrast, τ_{ij} is a dynamic attribute, which shall be *dynamically* changed during the update of W_t. Denote the performance of *s*-th traveling route by L_s. By the inclusion of L_s, the attribute τ_{ij} (also called *pheromone*) is updated as follows.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{s=1}^{S} \Delta\tau_{ij}^{s}, \qquad (1)$$

where

$$\Delta\tau_{ij}^{s} = \begin{cases} \frac{Q}{L_s} & \text{when } s\text{-th traveler passes through path } ij \\ 0 & \text{otherwise} \end{cases}$$

Q: a constant (parameter, $Q > 0$)
ρ: evaporation rate (parameter, $1 > \rho > 0$)

In the above formulation (1), $\tau_{ij}(t+1)$ denotes the *pheromone* on path *ij* in network W_{t+1}. Notice that the *pheromone* on each path of W_t shall evaporate with a certain percentage (ρ) while we transit from W_t to W_{t+1}. The term $(1-\rho)\tau_{ij}(t)$ denotes the remaining *pheromone* after such evaporation. The term $\Delta\tau_{ij}^{s}$ denotes the feedback of *s*-th ant about the preference path *ij*. While *s*-th ant did travel through path *ij*, the ant suggests the *pheromone* on path *ij* shall be increased by this amount $\Delta\tau_{ij}^{s} = Q/L_s$. The smaller is L_s (smaller makespan), the larger is $\Delta\tau_{ij}^{s}$. In contrast, if *s*-th ant did not travel through path *ij*, then the *pheromone* on path *ij* shall not be increased ($\Delta\tau_{ij}^{s} = 0$).

The above formula for updating $\tau_{ij}(t)$ is essentially a recursive form. To carry out the recursive form, we need to define $\tau_{ij}(0)$, the initial value of *pheromone* on path *ij*. In *ACO_S*$_{new}$, we use the criterion of SPT (shortest processing time) to generate a job sequence. That is, in sequencing *N* jobs, the shorter is the processing time, the higher is the sequence priority. We call such a chromosome the *initial chromosome* ω_0, denote its makesan by L_0, and set $\tau_{ij}(0) = 1/(S \cdot L_0)$ where *S* is the total number of ants (travelers).

Given $\tau_{ij}(t)$ and η_{ij}, the aggregate attribute $\lambda_{ij}(t)$ is defined as follows.

$\lambda_{ij}(t) = [\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}$, where α, β are positive integers.

Given W_t, in which $\lambda_{ij}(t)$ is available, an ant travels through the network in a *probabilistic* manner. As stated, while *s*-th ant is now at node *i*, its probability of proceeding to node *k* can be defined below.

$$p_{ik}^{s}(t) = \begin{cases} \dfrac{\lambda_{ik}(t)}{\sum_{j \in J_s} \lambda_{ij}(t)} & if\ node\ k \in J_s \\ 0 & if\ node\ k \notin J_s \end{cases}$$

where J_s denotes the set of nodes that have not been travelled through by *s*-th ant. Due to the probabilistic feature of the tournament method, for any two ants *s* and *q*, we would find that $J_s \neq J_q$ in most cases; in turn, this leads to $p_{ik}^{s}(t) \neq p_{ik}^{q}(t)$. The variable p_{ik}^{s} is also called *transition probability* for ant *s*.

While W_t is obtained in the ACO network updating process, the best chromosome ever found must be recorded. We denote such a chromosome $\omega_{best}(t)$ and its makespan $L_{best}(t)$. The ACO updating process shall terminate while the best solution keeps unchanged for T_f generations; that is, $L_{best}(t_n) = L_{best}(t_n+1) = \cdots = L_{best}(t_n+T_f)$.

Procedure ACO_S$_{new}$

Step 1: Initialization
- Input parameters $\rho, \alpha, \beta, S, T_f$
- Set $t = 0$;

Step 2: Compute $\tau_{ij}(0)$
- Generate *initial chromosome* ω_0;
- Compute its makespan L_0; Set $\tau_{ij}(0) = 1/(S \cdot L_0)$
- Create ACO network W_0;

Step 3: Update ACO network W_t
- Send *S* ants to travel through network W_t;
- Obtain *S* traveling routes (chromosomes) and their makespans;
- Obtain W_{t+1} by using the ACO network updating method;
- Record the best chromosome $\omega_{best}(t)$ and its makespan $L_{best}(t)$.

Step 4: Termination Check
- If $L_{best}(t_n) = L_{best}(t_n+1) = \cdots = L_{best}(t_n+T_f)$, STOP;
- Else, Go to Step 3.

B. ACO_S$_{old}$ Algorithm

See Fig. 1, an S_{old} chromosome is a *F+1* segment. In *ACO_S*$_{old}$, a chromosome is modeled by a *composite* ACO network, which is composed of *F+1 sub-networks*. That is, we model each segment by a sub-network; the resulting *F+1* sub-networks are then connected together to yield a composite ACO network (see Fig. 4).

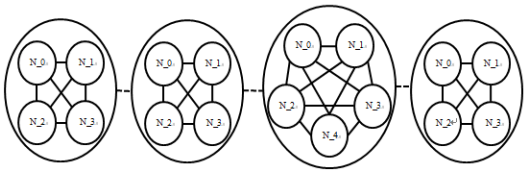

Fig. 4. A composite ACO network for the chromosome

In *ACO_S*$_{old}$, a chromosome is obtained by requesting an ant travel through the composite ACO network. That is, the ant has to travel through all sub-networks in order. Once traveling through a sub-network, the ant is automatically sent to the *start node* of the next sub-network. Therefore, the ultimate traveling route of the composite ACO network represents a solution.

V. NUMERICAL EXPERIMENTS

Numerical experiments are carried out to compare the two meta-heuristic algorithms, which are coded in C++ programming languages, running on personal computers equipped with AMD Athlon(tm) II *4640 3.0GHz CPU and 4G RAM.

Data sets for the experiments are adopted from prior studies [1,2,3]. The data sets are concerned with *large-scale*

problems, which are categorized into three scenarios. Each scenario includes 30 problem instances; each problem instance denotes a unique scheduling problem. In each problem *instance*, 15 experiments *runs* are carried out.

Of the three *scenarios*, each one is designated by *X*-*F*-*m*, where *X* denotes the type of setup times (LSU, MSU, SSU), *F* is the number of families, and *m* is the number of machines. In addition, SSU denotes small setup times, MSU denotes medium setup times, and LSU denotes large setup times (Table 1).

Table 1
Results of the Numerical experiments for ACO

Scenario	*Makespan*			*Computation Time*	
	Lnew (min.)	*Lold (min.)*	*γ (%)*	*Tnew (sec.)*	*Told (sec.)*
SSU1010	551.92	554.87	0.56	98.01	29.51
MSU1010	687.61	698.2	1.52	164.46	40.3
LSU1010	947.02	972.93	2.66	295.34	61.76
Average	728.85	742.00	1.58	185.94	43.86

The performance difference between ACO_S_{old} and ACO_S_{new} is denoted by $\gamma = (L_{old} - L_{new})/L_{old}$. Where L_{old} and L_{new} respectively denote the makespan of the two algorithms. Notice that $\gamma > 0$ implies that ACO_S_{new} outperforms ACO_S_{old}. The parameters of the two ACO algorithms are set as follows: $\rho = 0.8,\ \ S = \mathrm{C}(N, 2)$, where *N* is the total number jobs, $\alpha = 1,\ \beta = 2,$ and T_f = 10,000.

Table 1 shows a comparison between the experiment results of ACO_S_{new} and ACO_S_{old}. The first index γ ranges from 0.56% to 2.66%, and its average is 1.58%. This implies that ACO_S_{new} outperforms ACO_S_{old}. The average of T_{old} is 43.86 sec., and the average of T_{new} is 185.94 sec, which indicate that the two ACO algorithms are computationally inexpensive. This finding advocates the value of developing new chromosome representation scheme in the application of meta-heuristic algorithms.

VI. CONCLUDING REMARKS

This paper attempts to highlight the importance of developing new chromosome representation in the application of meta-heuristic algorithms. Such a research claim is justified by solving a scheduling problem, called permutation manufacturing-cell flow shop (PMFS). We compare the effectiveness of two chromosome representation schemes (S_{old} and S_{new}) while they are embedded in a particular meta-heuristic algorithm (ACO) to solve the scheduling problem.

Experiment results indicate that ACO_S_{new} outperforms ACO_S_{old}. This implies that developing appropriate chromosome representations might be very important in the application of meta-heuristic algorithms. This further raises the research interests for comparing the effectiveness of some other meta-heuristic algorithms while they are embedded with S_{old} and S_{new}

ACKNOWLEDGEMENTS

This work is financially supported by a research contract NSC99-2221-E-009-110-MY3.

REFERENCES

[1] J. E Schaller, J. N. D. Gupta, A. J. Vakharia, ―Scheduling a flowline manufacturing cell with sequence dependent family setup times," *European Journal of Operational Research*, 125(2) (2000) 324-339.

[2] S.W. Lin, K.C. Ying, Z.J. Lee, ―Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times," *Computers & Operations Research*, 36 (2009) 1110-1121.

[3] P.M. Franca, J.N.D. Gupta, A. S. Mendes, ―Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups," *Computers and Industrial Engineering,* 48(3) (2005) 491–506.

[4] M. Dorigo , L. M. Gambardella, ―Ant colony system: A cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, 1(1) (2002) 53-66.