

Job Shop Scheduling Based on ACO with a Hybrid Solution Construction Strategy

Shih-Pang Tseng
Department of Computer Science
and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan, R.O.C.
Email: tsp@mail.tajen.edu.tw

Chun-Wei Tsai
Department of Applied
Geoinformatics
Chia Nan University of Pharmacy & Science
Tainan, Taiwan, R.O.C.
Email: cwtsai0807@gmail.com

Jui-Le Chen
Department of Electrical
Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
Email: reler@mail.tajen.edu.tw

Ming-Chao Chiang
Department of Computer Science
and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan, R.O.C.
Email: mcchiang@cse.nsysu.edu.tw

Chu-Sing Yang
Department of Electrical
Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
Email: csyang@ee.ncku.edu.tw

Abstract—This paper presents a novel ant colony optimization (ACO) based on an efficient solution construction strategy (transition operator) for improving the quality of the end results of job shop scheduling problem (JSSP). Inspired by the observation that the quality of the end results of ACO is largely affected by their operators—especially the transition operator, a novel solution construction strategy is presented in this paper. The proposed algorithm uses two different strategies to compute the probability of solution construction to improve the end results. Our experimental results show that the proposed algorithm outperforms all state-of-the-art job shop scheduling algorithms evaluated in this paper and can significantly improve the quality of ant colony optimization for JSSP.

Keyword: ant colony optimization; job shop scheduling problem; transition operator.

I. INTRODUCTION

As one of the most well known combinatorial optimization problems, the job shop scheduling problem (JSSP) [1], [2], [3], [4], [5] is to assign a finite set of jobs, each of which consists of a sequence of operations, to a finite set of machines in such a way that the maximum completion time or makespan is minimized while assuming that each operation of a job is assigned to a different machine in the order given and is processed to completion without being preempted. Formally, the JSSP is usually defined as follows: Given a set of m machines and a set of n jobs—each of which consists of a sequence of operations $\langle O_{1j}, O_{2j}, \dots, O_{mj} \rangle$, with O_{ij} to be processed on a given machine μ_{ij} for p_{ij} time where $\mu_{ij} \neq \mu_{i+1,j}$ for $i = 1, 2, \dots, m-1$ —the problem is to find an assignment of the n jobs to the m machines that minimizes the maximum completion time or makespan. In symbols,

$$C_{\max} = \min \max_{1 \leq j \leq n} C_j$$

where C_j denotes the completion time of the j th job.

Recently, the JSSP has become an important research topic for several reasons. The first is that it has become extremely important to minimize the computation time, or maximize the throughput, in this era because it is important to conserve the energy so as to protect the environments or reduce the cost. For example, the cost can be reduced by fabricating the same products with less energy. The second is that an efficient solution to the JSSP can be used to solve many other problems. As such, several variants of JSSP have been introduced in recent years to solve the nurse scheduling problem, the flexible job shop scheduling problem, the fuzzy job shop scheduling problem, and the flow shop scheduling problem. However, even after all these efforts, the NP-hard job shop scheduling problem [6] is still very hard to be solved to optimality even if the size of benchmark is small. For this reason, it has always been a popular research topic of combinatorial optimization problems because a better solution can be used to enhance the work efficiency or reduce the energy cost of a factory.

One of the most well known examples to explain the difficulty of JSSP was introduced by Fisher and Thompson [1] in 1960s, which is a 10×10 instance (i.e., $n = 10$ and $m = 10$), and its optimal solution was not found until 20 years after this problem was introduced. The good news is that the optimum solution has been found by Carlier and Pinson [7]. They used a branch and bound algorithm to solve the JSSP in 1989. However, although the exhaustive search algorithm can find the optimum solution of this benchmark (i.e., the 10×10 instance), it is still very hard to find the optimum solution when the size of benchmark becomes 20×20 [8]. For this reason, since 1990's, approximation algorithms (some of which are called metaheuristics) such as priority dispatch, shifting bottleneck procedure [9], tabu search (TS) [10], [11], simulated annealing (SA) [12], genetic algorithm (GA) [13], [14], and ant colony

optimization (ACO) [15], [16], [17], [18], [19] have been used to solve the JSSP in a reasonable time. In addition to using the original version of the above metaheuristic algorithms, many studies have tried to integrate these algorithms to find a better result. A good example is the TSSA by Zhang et al. [8], which combines TS with SA.

Among these researches, ACO is apparently one of the most promising metaheuristic algorithms to solve the JSSP because its search abilities are very efficient for the combinatorial optimization problems. More important, ACO is not sensitive to the initial solutions. In this paper, we present an efficient algorithm, called hybrid selection strategy for ant colony optimization (HSSACO), to solve the JSSP that provides a better result than traditional metaheuristics can. The new transition operator of the proposed algorithm is designed to modify the solution construction strategy which integrates more than one objective compared to the traditional ACO for JSSP, which considers only one objective at a time (e.g., earliest starting time; or EST for short) so as to improve the quality of the end results. To evaluate the performance of the proposed algorithm, we compare it with four ACO-based algorithms for several well known benchmarks for the JSSP. Experimental results show that the proposed algorithm, HSSACO, outperforms all state-of-the-art algorithms for the JSSP compared in this paper.

The remainder of the paper is organized as follows. Section II gives a brief introduction to the genetic algorithm and approaches taken to enhance its performance. Section III provides a detailed description of the proposed algorithm. Also given in Section III is a simple example to demonstrate how the proposed algorithm works. Performance evaluation of the proposed algorithm is presented in Section IV. Analysis of the proposed algorithm is given in Section IV. Conclusion is drawn in Section V.

II. RELATED WORK

Proposed by Dorigo et al. in the 90s, the ant colony optimization (ACO) [20], [21], [22] is a promising metaheuristic algorithm inspired by the foraging behavior of real ants. The focus of the early version of ACO has been on solving the combinatorial optimization problems such as the traveling salesman problem and the quadratic assignment problem. Basically, ACO is made up of two required operators *SolutionConstruction()* and *PheromoneUpdate()* as well as one optional operator *LocalSearch()*.

The *SolutionConstruction()* operator plays the role of calculating the probability of choosing the next sub-solution of i , which is defined as follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{j \in \mathcal{N}_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta} & \text{if } j \in \mathcal{N}_i^k, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $\mathcal{N}_i^k$ denotes the set of feasible (or candidate) sub-solutions that can be the next sub-solution of i ; τ_{ij} the pheromone value between the sub-solutions i and j ; and η_{ij} the quality of the sub-solution j that will affect each ant's determination to move to j when at i . The parameters α and

β are used to adjust the weight of exploration and exploitation. The *PheromoneUpdate()* operator is employed in updating the pheromone value τ_{ij} on each edge e_{ij} , which is defined as follows:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho \sum_{k=1}^m \Delta\tau_{ij}^k, \quad (2)$$

where m denotes the number of ants; $\Delta\tau_{ij}^k = 1/L^k$; L^k the quality of solution created by ant k . Moreover, $\rho \in (0, 1]$ denotes the evaporation rate of pheromone values on the pheromone table. The optional operator *LocalSearch()* is employed in improving the quality of the solution of ACO. A good example is problem-specific permutation method.

Recently, several studies [15], [16], [18], [19] have attempted to use ACO to solve the JSSP because of its potential to find solutions that are closer to the optimum solution than all traditional heuristic algorithms. In 1994, Colormi et al. [15] used the early version of ACO, ant system (AS), to solve the JSSP in which the way the solution is represented is similar to that of the traveling salesman problem, which consists of $n+1$ nodes and $n(n-1)/2+n$ edges. Their simulation results showed that AS can find approximate solution to the JSSP that is about 7% to 10% away from the optimum solution of the benchmark they tested. In 1999, Zwaan and Marques [16] first analyzed the performance of the parameter settings of AS and then used the information so obtained to fine-tune the parameter values of AS. More precisely, the interaction between α and β was first tested, followed by the values of ρ and the numbers of ants. Then, Zwaan and Marques suggested that the number of ants of AS be close to $n \times m$ where n and m are the number of jobs and the number of machines, respectively.

More recently, Heinonen and Pettersson [18] combined ACO with efficient local search method to improve the results of ACO in solving the JSSP. The local search method first identifies the critical path in the schedule constructed by ACO. The critical path can be decomposed into a number of blocks each of which is a maximal sequence of adjacent operations that require to be processed by the same machine. Then, the local search method is used to swap operations to improve the makespan of the schedule. Their simulation results showed that ACO with efficient local search method can find a near-optimum solution to the MT10 problem the relative error of which is about 3.9%. In [19], Huang and Liao presented a hybrid algorithm, called ACO combined with fast TS (ACOFT), that uses ant colony optimization to find the global search directions and tabu search to fine-tune the solution. Their simulation results showed that ACOFT can find a near-optimum solution the relative error of which is 1.798% on average. In addition to using ACO to solve the JSSP, some studies [23], [24] also apply ACO to solve the variants of JSSP. Examples are the flexible job shop scheduling problem and the fuzzy job shop scheduling problem.

III. THE PROPOSED ALGORITHM

In this section, we will begin with the concept, followed by a detailed description of the proposed algorithm.

A. The concept

Generally speaking, most ACO-based algorithms for JSSP use only one kind of strategy to optimize the makespan such as earliest starting time (EST), most remaining operations (MRO), shortest processing time (SPT), and most work remaining (MWR). More precisely, $\eta(o_{ij})$ can be computed as EST, MRO, or the others. However, even though all the heuristic methods are aimed to minimize the makespan, the strategies they use are very different. For example, the EST is aimed to perform the jobs at the earliest possible starting time whereas the SPT is aimed to perform the shortest jobs first.

In this paper, we integrate two different heuristics into the solution construction method of the ACO-based algorithms for solving JSSP. As shown in Eq. (3), the probability, p_{ij}^k , of the proposed algorithm is made up of two different search strategies. One is the earliest starting time (as shown in Eq. (5)) while the other is the most remaining operations (as shown in Eq. (6)).

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{em}(o_{ij})]^\beta}{\sum_{o_{ij} \in \mathcal{N}_i} [\tau_{ij}]^\alpha [\eta_{em}(o_{ij})]^\beta}, \quad (3)$$

$$\eta_{em}(o_{ij}) = \eta_{est}^{w_1}(o_{ij}) \times \eta_{mro}^{w_2}(o_{ij}), \quad (4)$$

$$\eta_{est}(o_{ij}) = \frac{\frac{1}{s_{ij}+1}}{\sum_{o_{ij} \in \mathcal{N}_i} \frac{1}{s_{ij}+1}}, \quad (5)$$

$$\eta_{mro}(o_{ij}) = \frac{r_{ij} + 1}{\sum_{o_{ij} \in \mathcal{N}_i} r_{ij} + 1}, \quad (6)$$

where s_{ij} denotes the possible starting time of o_{ij} ; r_{ij} the number of remaining operations of o_{ij} ; $\mathcal{N}_i$ the remaining operations that satisfy the processing order restriction but have not started yet; w_1 and w_2 (where $w_1, w_2 \in [0, 1]$ and $w_1 + w_2 = 1$) the weights of the two makespan optimization strategies. As shown in Eq. (4), more than one makespan optimization strategy is considered at a time to avoid the ACO-based algorithms from falling into local minima at early iterations. Moreover, $\eta_{em}(o_{ij})$ is the weighted geometric mean of these two strategies $\eta_{est}^{w_1}(o_{ij})$ and $\eta_{mro}^{w_2}(o_{ij})$. The proposed algorithm not only considers searching the possibility of earliest possible starting time of the job, it also takes into account performing the job that which has a larger number of operations that are not started yet. For instance, suppose $w_1 = w_2 = 0.5$. Then, $\eta_{em} = \eta_{est}^{0.5} \times \eta_{mro}^{0.5} = \sqrt{\eta_{est} \times \eta_{mro}}$. Moreover, to avoid converging to a single direction, we also normalize the values of these two makespan optimize strategies, as shown in Eqs. (5) and (6).

B. The hybrid selection strategy for ant colony optimization

In the previous section, we present a new solution construction strategy (also called the transition operator) for the ACO-based algorithm. As Eqs. (3)–(6) show, the hybrid selection strategy for ant colony optimization is as follows.

- 1) Input the dataset and its parameters.
- 2) Initialization:

- a) Set the values of all paths τ_{ij} in the pheromone table to τ_{init} .
- b) Initialize the parameters of ACO.
- c) Put all the ants on the start node.

3) Solution Construction:

For each ant,

- a) Compute the probability distribution by using Eq. (3).
- b) Construct the solution by using the probability distribution p_{ij}^k .
- c) Update the local pheromone by using Eq. (2).
- 4) Update the pheromone of the best solution.
- 5) Perform local search to improve the solution.
- 6) If the stop condition is not met, go to 2.
- 7) Otherwise, output the best so far solution as the final solution.

Because our aim is to apply the proposed algorithm to all the ACO-based algorithms, the description of the proposed algorithm herein is for the general design of ACOs. First, the proposed algorithm will read the dataset and its parameters. Next, set all the paths in the pheromone table (an $m \times n$ array) to the initial value τ_{init} . Then, initialize all the other parameters of ACO such as α , β , ρ , w_1 , and w_2 . In the solution construction stage, HSSACO will first use Eq. (3) to compute the probability distribution of ant k . Then, ant k will use the probability distribution p_{ij}^k to determine which will be the next sub-solution. For the JSSP, it means the next operation. After the next sub-solution is determined, the proposed algorithm will then perform the local pheromone update if required by the ACO we use. In addition to the local pheromone update, the global pheromone update will then be performed. In this paper, the proposed algorithm also performs local search to improve and fine-tune the final results. The above operators will be repeatedly performed until convergence or the stop condition is met.

IV. SIMULATION RESULTS

To simplify the discussion of the simulation results, the following conventions are employed in this paper.

$$\Delta = \frac{\Delta_{HSS+\psi} - \Delta_\psi}{\Delta_\psi} \times 100\%, \quad (7)$$

where Δ denotes the enhancement of $\Delta_{HSS+\psi}$ compared with Δ_ψ in percentage of makespan. That is, ψ denotes either earliest starting time (EST) or most remaining operations (MRO) used by the ant system (AS) [15], elitist ant system (EAS) [25], [20], rank ant system (RAS) [26], and ant colony system (ACS) [21].

A. Parameter settings and datasets

The empirical analysis was conducted on an SUN X4150 machine with 2.5 GHz Xeon CPU and 4GB of memory running CentOS 5.4 with Linux 2.6.18. Moreover, all the programs are written in C++ and compiled using g++ (GNU C++ compiler).

TABLE II
IMPACT OF w_1 AND w_2 .

	0.1/0.9	0.2/0.8	0.3/0.7	0.4/0.6	0.5/0.5	0.6/0.4	0.7/0.3	0.8/0.2	0.9/0.1
AS	-2.48	-2.62	-2.59	-2.52	-2.56	-2.45	-1.98	-1.59	-1.15
EAS	-2.21	-2.29	-2.40	-2.39	-2.45	-2.48	-2.29	-2.06	-1.41
RAS	-2.02	-2.08	-2.19	-2.24	-2.23	-2.27	-2.15	-1.94	-1.36
ACS	-4.84	-4.89	-4.88	-4.89	-4.93	-4.96	-5.02	-5.03	-4.74

TABLE III
SIMULATION RESULTS OF ACO WITHOUT LOCAL SEARCH

Dataset		BKS	AS		HSS+AS			EAS		HSS+EAS		
			EST	MRO	ESTMRO	δ_{est}	δ_{mro}	EST	MRO	ESTMRO	δ_{est}	δ_{mro}
la01	10× 5	666	725	737	706	-2.64	-4.14	716	722	701	-1.99	-2.89
la05	10× 5	593	595	593	593	-0.27	0.00	593	593	593	-0.06	0.00
la10	15× 5	958	981	958	958	-2.32	0.00	969	958	958	-1.12	0.00
la15	20× 5	1,207	1,394	1,378	1,364	-2.11	-1.02	1,366	1,377	1,365	-0.08	-0.87
la20	10×10	902	1,070	1,028	1,014	-5.27	-1.43	1,056	1,020	995	-5.81	-2.51
la25	15×10	977	1,228	1,182	1,165	-5.13	-1.49	1,209	1,179	1,155	-4.50	-2.06
la30	20×10	1,355	1,668	1,591	1,570	-5.88	-1.32	1,645	1,584	1,554	-5.52	-1.94
la31	30×10	1,784	2,082	1,951	1,935	-7.04	-0.81	2,026	1,943	1,906	-5.93	-1.91
la32	30×10	1,850	2,203	2,088	2,070	-6.05	-0.90	2,157	2,071	2,039	-5.47	-1.54
la33	30×10	1,719	2,008	1,950	1,916	-4.57	-1.74	1,987	1,927	1,884	-5.15	-2.22
la34	30×10	1,721	2,078	1,977	1,945	-6.37	-1.63	2,034	1,958	1,924	-5.42	-1.73
la35	30×10	1,888	2,243	2,226	2,164	-3.53	-2.79	2,181	2,222	2,142	-1.79	-3.60
la36	15×15	1,268	1,624	1,506	1,497	-7.82	-0.60	1,592	1,490	1,464	-8.03	-1.73
la37	15×15	1,397	1,735	1,744	1,695	-2.35	-2.83	1,705	1,720	1,680	-1.46	-2.33
la38	15×15	1,196	1,538	1,522	1,499	-2.58	-1.56	1,513	1,507	1,479	-2.24	-1.85
la39	15×15	1,233	1,577	1,537	1,505	-4.60	-2.11	1,554	1,513	1,478	-4.91	-2.33
la40	15×15	1,222	1,561	1,498	1,474	-5.63	-1.66	1,520	1,479	1,450	-4.60	-1.95
Average						-4.36	-1.53				-3.77	-1.85

Dataset		BKS	RAS		HSS+RAS			ACS		HSS+ACS		
			EST	MRO	ESTMRO	δ_{est}	δ_{mro}	ESTMRO	MRO	HSS	δ_{est}	δ_{mro}
la01	10× 5	666	706	723	700	-0.89	-3.19	802	765	740	-7.78	-3.39
la05	10× 5	593	593	593	593	-0.01	0.00	603	593	593	-1.72	0.00
la10	15× 5	958	971	958	958	-1.30	0.00	999	958	958	-4.13	0.00
la15	20× 5	1,207	1,372	1,373	1,356	-1.19	-1.25	1,389	1,432	1,374	-1.02	-3.99
la20	10×10	902	1,046	1,004	996	-4.77	-0.75	1,182	1,105	1,068	-9.58	-3.34
la25	15×10	977	1,199	1,177	1,148	-4.26	-2.47	1,262	1,168	1,151	-8.83	-1.51
la30	20×10	1,355	1,623	1,583	1,553	-4.33	-1.88	1,674	1,615	1,537	-8.19	-4.87
la31	30×10	1,784	2,020	1,935	1,910	-5.48	-1.30	2,004	1,923	1,835	-8.43	-4.60
la32	30×10	1,850	2,144	2,066	2,032	-5.21	-1.61	2,210	2,081	1,973	-10.71	-5.16
la33	30×10	1,719	1,978	1,924	1,882	-4.88	-2.17	1,974	1,876	1,810	-8.28	-3.52
la34	30×10	1,721	2,038	1,953	1,916	-5.97	-1.91	2,056	2,000	1,886	-8.26	-5.68
la35	30×10	1,888	2,185	2,217	2,143	-1.90	-3.31	2,179	2,223	2,128	-2.34	-4.26
la36	15×15	1,268	1,578	1,484	1,457	-7.67	-1.81	1,628	1,489	1,461	-10.24	-1.84
la37	15×15	1,397	1,706	1,716	1,678	-1.66	-2.21	1,632	1,732	1,645	0.81	-4.98
la38	15×15	1,196	1,503	1,509	1,473	-2.02	-2.42	1,533	1,533	1,444	-5.79	-5.77
la39	15×15	1,233	1,555	1,513	1,481	-4.73	-2.11	1,485	1,493	1,430	-3.71	-4.21
la40	15×15	1,222	1,519	1,481	1,445	-4.85	-2.40	1,483	1,492	1,428	-3.71	-4.28
Average						-3.60	-1.81				-6.00	-3.61

TABLE I
PARAMETER SETTINGS OF THE ACO ALGORITHMS.

Algorithm	α	β	ρ	τ
AS	1	10	0.1	$1/C$
ACS	—	10	0.1	1.0
EAS	1	10	0.1	a_n/C
AS-rank	1	10	0.1	$0.5w(w-1)/C$

Moreover, all the simulations are carried out for 30 runs, and all the ACO-based algorithms tested in this paper use 25

ants and 100 iterations. The parameter κ of EAS is set to 25. C denotes the makespan produced randomly; a_n the number of ants. The number of elitist ants w of AS-rank is set to 6. For the other settings, some of them are repeated below in Table I for easy reference.

The datasets used in this paper are the la series from la01 up to la40. These datasets contain from 10 up to 30 jobs and from 5 up to 15 machines. Moreover, these datasets are available for download from OR-Library at <http://people.brunel.ac.uk/~mastjib/jeb/info.html>.

TABLE IV
SIMULATION RESULTS OF ACOs WITH LOCAL SEARCH

Dataset		BKS	AS		HSS+AS			EAS		HSS+EAS		
			EST	MRO	ESTMRO	δ_{est}	δ_{mro}	EST	MRO	ESTMRO	δ_{est}	δ_{mro}
la01	10× 5	666	666	681	677	1.60	-0.63	670	676	671	0.15	-0.68
la05	10× 5	593	593	593	593	0.00	0.00	593	593	593	0.00	0.00
la10	15× 5	958	958	958	958	0.00	0.00	958	958	958	0.00	0.00
la15	20× 5	1,207	1,323	1,279	1,274	-3.67	-0.34	1,297	1,274	1,268	-2.22	-0.42
la20	10×10	902	97 5	944	946	-2.95	0.25	970	939	939	-3.19	-0.04
la25	15×10	977	1,134	1,092	1,084	-4.39	-0.67	1,108	1,079	1,079	-2.66	0.01
la30	20×10	1,355	1,551	1,460	1,462	-5.72	0.14	1,507	1,463	1,454	-3.53	-0.63
la31	30×10	1,784	1,919	1,811	1,807	-5.82	-0.20	1,887	1,803	1,795	-4.85	-0.45
la32	30×10	1,850	2,062	1,914	1,920	-6.90	0.30	1,995	1,904	1,901	-4.71	-0.16
la33	30×10	1,719	1,927	1,784	1,784	-7.44	-0.04	1,857	1,766	1,767	-4.84	0.04
la34	30×10	1,721	1,946	1,835	1,829	-6.00	-0.30	1,901	1,819	1,812	-4.68	-0.41
la35	30×10	1,888	2,039	2,037	2,010	-1.41	-1.34	2,045	2,034	1,997	-2.35	-1.83
la36	15×15	1,268	1,536	1,409	1,413	-7.99	0.32	1,475	1,398	1,387	-5.99	-0.75
la37	15×15	1,397	1,632	1,627	1,612	-1.22	-0.94	1,600	1,602	1,590	-0.61	-0.70
la38	15×15	1,196	1,404	1,403	1,404	-0.02	0.06	1,407	1,392	1,386	-1.47	-0.42
la39	15×15	1,233	1,398	1,392	1,401	0.22	0.62	1,445	1,387	1,380	-4.46	-0.49
la40	15×15	1,222	1,394	1,395	1,382	-0.86	-0.95	1,419	1,376	1,358	-4.28	-1.30
Average						-3.09	-0.22				-2.92	-0.48

Dataset		BKS	RAS		HSS+RAS			ACS		HSS+ACS		
			EST	MRO	ESTMRO	δ_{est}	δ_{mro}	EST	MRO	ESTMRO	δ_{est}	δ_{mro}
la01	10× 5	666	668	673	671	0.41	-0.38	719	693	689	-4.16	-0.58
la05	10× 5	593	593	593	593	0.00	0.00	593	593	593	-0.02	0.00
la10	15× 5	958	958	958	958	0.00	0.00	958	958	958	0.00	0.00
la15	20× 5	1,207	1,294	1,275	1,267	-2.12	-0.60	1,328	1,321	1,314	-1.04	-0.53
la20	10×10	902	960	937	937	-2.34	0.09	1,098	992	998	-9.12	0.66
la25	15×10	977	1,102	1,083	1,075	-2.46	-0.71	1,168	1,095	1,098	-5.98	0.28
la30	20×10	1,355	1,504	1,454	1,452	-3.50	-0.14	1,578	1,470	1,453	-7.90	-1.16
la31	30×10	1,784	1,881	1,802	1,800	-4.30	-0.10	1,925	1,787	1,784	-7.32	-0.18
la32	30×10	1,850	1,984	1,899	1,897	-4.36	-0.09	2,125	1,919	1,866	-12.18	-2.77
la33	30×10	1,719	1,844	1,764	1,762	-4.42	-0.11	1,917	1,745	1,728	-9.84	-0.92
la34	30×10	1,721	1,900	1,821	1,814	-4.53	-0.39	1,987	1,823	1,795	-9.67	-1.51
la35	30×10	1,888	2,030	2,029	1,996	-1.64	-1.63	2,073	2,001	1,971	-4.93	-1.52
la36	15×15	1,268	1,470	1,394	1,382	-5.99	-0.89	1,564	1,420	1,409	-9.90	-0.76
la37	15×15	1,397	1,592	1,598	1,586	-0.39	-0.73	1,582	1,615	1,604	1.39	-0.64
la38	15×15	1,196	1,394	1,387	1,383	-0.78	-0.32	1,454	1,399	1,396	-3.95	-0.18
la39	15×15	1,233	1,438	1,381	1,377	-4.22	-0.25	1,431	1,378	1,356	-5.23	-1.56
la40	15×15	1,222	1,403	1,375	1,356	-3.35	-1.43	1,409	1,392	1,377	-2.27	-1.12
Average						-2.59	-0.45				-5.42	-0.73

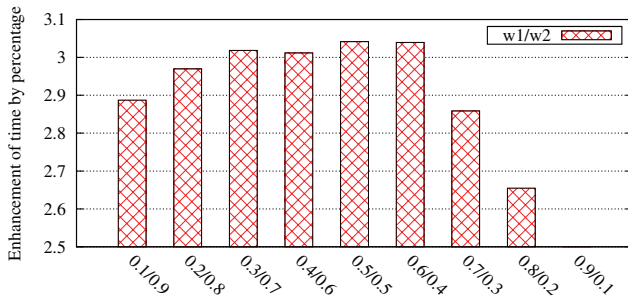


Fig. 1. Impact of w_1 and w_2 on average.

B. Results and Analysis

Table II shows the impact of w_1 and w_2 on the performance of different ACO-based algorithms. Moreover, Table II also shows the results of setting $w_1 = 1.0$ and $w_2 = 0.0$ for the 40 la series datasets from la10 up to la40 on average. As shown in Fig. 1, the settings of $w_1 = w_2 = 0.5$ and $w_1 = 0.6, w_2 = 0.4$ provide better results than the other settings. For this reason,

we set $w_1 = w_2 = 0.5$ for all the simulations to be given later in this paper. Tables III and IV show the results of applying the proposed algorithm to four ACO-based algorithms—AS, EAS, RAS, and ACS. For each of these algorithms, two heuristic methods, EST and MRO, are used to compute η . In other words, δ_{est} and δ_{mro} denote, respectively, the enhancement of the proposed algorithm compared with the ACO-based algorithms with EST and MRO. Also, BKS denotes the best known solution of each benchmark.

Table III shows the results of ACO and ACO with the proposed algorithm but no local search. For all the benchmarks tested, the results show that the proposed algorithm can improve the makespan by from 1.53% up to 6.00% on average, by 0.00% in the worst case, and by 10.71% in the best case. Some of the results show that the improvement of HSS is 0.00%. That is because like AS and HSS+AS with MRO for la05 and la10, HSS and MRO get the BKS. In addition, Table III shows that the proposed algorithm can find the better results compared with the original ACO for all the benchmarks tested. Table IV gives the results with local search

proposed by Nowicki and Smutnicki [27]. The results show that the ACO-based algorithms can find solutions closer to the optimum value. However, even in this case, the proposed algorithm can still improve the makespan by from 0.22% up to 5.42% on average, by -1.60% in the worst case, and by 12.18% in the best case. This shows that the proposed algorithm can effectively improve the makespan.

According to our observation, the proposed algorithm, HSSACO, can improve the final results of ACO-based algorithms, especially for the large scale datasets. This characteristic is shown in using the AS to solve la01 with the EST method. The proposed algorithm cannot improve the final results of the original version of AS for some of the benchmarks. It is because the results are very close to the optimum solution. However, the good news is that the proposed algorithm can improve the final results as the size of the dataset increases such as AS for la33. The proposed algorithm can improve the results by about 7.44% compared with AS using the EST method. Moreover, the proposed algorithm can also improve the results of EAS, RAS, and ACS. In summary, even though some of the final results of these simulations are not very close to the BKS, these comparisons have shown that the proposed algorithm has the potential to improve the quality of the end results of the ACO-based algorithms, especially when the solutions are not close to the optimum solution.

V. CONCLUSION

This paper presents a novel technique for improving the quality of ACO-based algorithms, called hybrid selection strategy for ant colony optimization (HSSACO). The proposed algorithm uses two different heuristic strategies to compute the probability for the solution construction operator so as to maintain the diversity of the solutions. To evaluate the performance of the proposed algorithm, we apply it to four state-of-the-art ACO-based algorithms—AS, EAS, RAS, and ACS. Moreover, we also use it to solve 10 JSSP datasets, the size of which are from 10 up to 20 jobs and from 5 up to 20 machines. The simulation results show that the proposed algorithm can effectively improve the results of ACO and its variants from 1.23% up to 4.05% on average. Our future focus will be on speeding up the proposed algorithm.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council, Taiwan, R.O.C., under Contract Nos. NSC100-2218-E-041-001-MY2 and NSC99-2220-E-006-008.

REFERENCES

- [1] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling* (J. F. Muth and G. L. Thompson, eds.), pp. 225–251, Prentice-Hall, 1963.
- [2] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 4, pp. 287–326, 1979.
- [3] F. A. Rodammer and J. K. Preston White, "A recent survey of production scheduling," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 6, pp. 841–851, 1988.
- [4] A. S. Jain and S. Meeran, "A state-of-the-art review of job-shop scheduling techniques," *European Journal of Operations Research*, no. 113, pp. 390–434, 1999.
- [5] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd ed., 2008.
- [6] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, pp. 117–29, 1976.
- [7] J. Carlier and E. Pinson, "An algorithm for solving the job-shop problem," *Management Science*, vol. 35, pp. 164–176, 1989.
- [8] C. Y. Zhang, P. Li, Y. Rao, and Z. Guan, "A very fast ts/sa algorithm for the job shop scheduling problem," *Computers & Operations Research*, vol. 35, pp. 282–294, 2008.
- [9] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, pp. 391–401, 1988.
- [10] É. D. Taillard, "Parallel taboo search techniques for the job shop scheduling problem," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 108–117, 1994.
- [11] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Annals of Operations Research*, vol. 41, pp. 231–252, 1993.
- [12] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40, pp. 113–125, 1992.
- [13] T. Yamada and R. Nakano, "Conventional genetic algorithm for job shop problems," in *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 474–479, 1997.
- [14] F. D. Croce, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," *Computers & Operations Research*, vol. 22, no. 1, pp. 15–24, 1995.
- [15] A. Colnori, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant System for Job-shop Scheduling," *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 34, no. 1, pp. 39–53, 1994.
- [16] S. van der Zwaan and C. Marques, "Ant Colony Optimisation for Job Shop Scheduling," in *Proceedings of the 3rd Workshop on Genetic Algorithms and Artificial Life*, 1999.
- [17] J. Zhang, X.-M. Hu, X. Tan, J. hui Zhong, and Q. Huang, "Implementation of an ant colony optimization technique for job scheduling problem," *Transactions of the Institute of Measurement and Control*, vol. 28, no. 1, pp. 93–108, 2006.
- [18] J. Heinonen and F. Pettersson, "Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem," *Applied Mathematics and Computation*, vol. 187, no. 2, pp. 989–998, 2007.
- [19] K.-L. Huang and C.-J. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1030–1046, 2008.
- [20] M. Dorigo, V. Maniezzo, and A. Colnori, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, vol. 26, no. 1, pp. 29–41, 1996.
- [21] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
- [22] M. Dorigo and T. Stützle, *Ant Colony Optimization*. The MIT Press, 2004.
- [23] N. Liouane, I. Saad, S. Hammadi, and P. Brone, "Ant systems & local search optimization for flexible job shop scheduling production," *International Journal of Computers, Communications & Control*, vol. 2, no. 2, pp. 174–184, 2007.
- [24] J. Šilc, P. Korosec, and B. Robič, "Combining vector quantization and ant-colony algorithm for mesh-partitioning," in *Proceedings of 5th International Conference Parallel Processing and Applied Mathematics (PPAM)*, pp. 113–118, 2003.
- [25] M. Dorigo, *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.
- [26] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system - a computational study," *Central European Journal for Operations Research and Economics*, vol. 7, pp. 25–38, 1997.
- [27] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *MANAGEMENT SCIENCE*, vol. 42, no. 6, pp. 797–813, 1996.