Hybrid Particle Swarm Optimization and Convergence Analysis for Scheduling Problems

Xue-Feng Zhang
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0382
xuefeng@ar.is.kyushu-u.ac.jp

Miyuki Koshimura
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0382
koshi@inf.kyushu-u.ac.jp

Hiroshi Fujita
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0382
fujita@inf.kyushu-u.ac.jp

Ryuzo Hasegawa
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819–0382
ryuzo@inf.kyushu-u.ac.jp

ABSTRACT

This paper proposes a hybrid particle swarm optimization algorithm and for solving Flow Shop Scheduling Problems (FSSP) and Job Shop Scheduling Problems (JSSP) to minimize the maximum makespan. A new hybrid heuristic, based on Particle Swarm Optimization (PSO), Tabu Search (TS) and Simulated Annealing (SA), is presented. By reasonably combining these three different search algorithms, we develop a robust, fast and simply implemented hybrid optimization algorithm HPTS (Hybrid of Particle swarm optimization, Tabu search and Simulated annealing). On the other hand, we analyze the convergence of PSO algorithm with an optimum keeping strategy and TS, SA algorithms by Markov chain theory at a different aspect in this paper, and HPTS algorithm is proved to be convergent. This hybrid algorithm is applied to the standard benchmark sets and compared with other approaches. The experimental results show that the proposed algorithm could obtain the high-quality solutions within relatively short computation time. Meanwhile, the convergence of HPTS is proved. For example, in 30 and 43 benchmarks, 7 new upper bounds and 6 new upper bounds are obtained by the HPTS algorithm, respectively.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis:**]: Constrained optimization, Global optimization, Stochastic programming

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.

General Terms

Theory, Algorithms, Performance.

Keywords

Flow Shop Scheduling Problem, Job Shop Scheduling Problem, Particle Swarm Optimization

1. INTRODUCTION

Flow-Shop Scheduling Problem (FSSP) is a combinatorial optimization problem and NP-complete [1]. The FSSP is one of the most studied problems in the area of scheduling theory and applications, known by its intractability from theoretical and computational aspects. The seminal paper from Johnson [34] was the first one to define the concept of flow shop scheduling. The main task of FSSP is to find a permutation schedule which minimizes the maximum completion time of a sequence of $N = \{J_1, \cdots J_n\}$ jobs in an M-machine flow-shop.

1. Every job has M operations, and every machine has N jobs.

2. Every job executes its operations on every machine in the order of $M = \{M_1, \cdots M_m\}$.

3. Every operation of a job cannot be preempted.

4. Every machine can execute only a single operation of a job at a time [2].

The Job Shop Scheduling Problem (JSSP) is also one of the existing combinatorial optimization problems and it has been demonstrated to be an NP-hard problem [1]. The JSSP consists of n jobs and m machines. Each job must go through m machines to complete its work. We consider one job consists of m operations. Each operation uses one of m machines to complete one job's work for a fixed time interval. Once one operation is processed on a given machine, it can not be interrupted before it finishes the job's work. The sequence of operations of one job should be predefined and may be different for any job. In general, one job be-

ing processed on one machine is considered as one operation noted as, then every job has a sequence of m operations. Each machine can process only one operation during the time interval. The objective of JSSP is to find an appropriate operation permutation for all jobs that can minimize the makespan C_{max}, i.e., the maximum completion time of the final operation in the schedule of $n \times m$ operations.

In the past few decades, these problems have attracted many researchers. Many heuristic algorithms have been proposed, such as Taillard's tabu search method [4], Ogbu and Smith's simulated annealing algorithm [5], shifting bottleneck approach (SB) [6], tabu search algorithm(TS) [7], genetic algorithm(GA) [3], simulated annealing(SA) [8], particle swarm optimization(PSO) [9], and ant colony(ACO) [10]. Particle swarm optimization (PSO) is an evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995, which is inspired by social behavior of bird flocking. It is endowed with properties of easy implementation and fast convergence. In recent years, there have been a lot of reported works focused on the modification of PSO to solve continuous optimization problems [11].

PSO algorithm is a kind of random optimization algorithm based on continuous optimization problems. Swarm intelligent of PSO is produced by cooperation and competition between particles, which is used for guiding optimization search. PSO has been studied widely in many applications due to its good global searching ability. Moreover, PSO algorithm is less studied to solve FSSP and JSSP. The PSO algorithm design of solving FSSP and JSSP is very difficult, and the efficient PSO algorithm design of solving JSSP is more difficult. Leticia etc. construct the single machine scheduling algorithm based on random coding of JSSP, and the algorithm is a kind of retardation minimum time algorithm. The algorithm utilizes the dynamic mutation operators to ensure the diversity of particle populations. The algorithm has been tested respectively with 40 jobs and 50 jobs, and the algorithm achieves better results. Lina etc construct PSO algorithm based on operation code to solve JSSP. They apply the crossover and mutation operation of GA in place of the update operations of velocity and position of PSO algorithm.

In the hybrid particle swarm optimization, Jerald. J etc apply GA, SA and PSO algorithms to solve the scheduling problems of flexible manufacturing systems. The hybrid algorithm optimizes machine idle time and reduces the cost of production tardiness. Liu etc. combine PSO and VNS. The hybrid algorithm minimizes the makespan of the flexible JSSP. Xia etc. design the hybrid PSO algorithm based on SA local search algorithm. The hybrid algorithm can solve multi-objective flexible JSSP. In order to minimize the makespan, Sha etc. construct the hybrid algorithm based on Hash table to solve JSSP. In the hybrid algorithm, Giffler-Thompson algorithm is adopted to construct the feasible solution from the particle location of Hash table, and SWAP operation updates the particle velocity. The hybrid algorithm combines with TS algorithm based on block Structure.

In this paper, we propose a new multi-structural hybrid evolutionary framework, and derive HPTS algorithm to solve FSSP and JSSP. The remainder of the paper is organized as follows. The remainder of the paper is organized as follows. In Section 2, we present an introduction for particle swarm optimization and discuss our approach to solve the FSSP and JSSP. we also prove the convergence of PSO algorithm with an optimum keeping strategy and TS, SA algorithms by Markov chain theory at a different aspect in Section 3. Section 4 gives experimental results of the HPTS algorithm and other competitive approaches. Finally, the main conclusions are drawn.

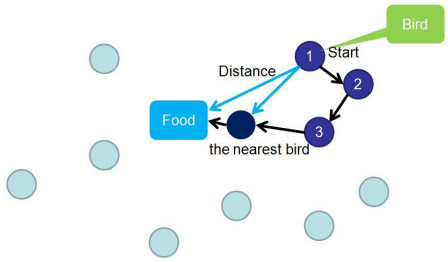

Figure 1: A PSO algorithm mimics the behavior of flying birds

2. THE PROPOSED ALGORITHM

PSO algorithm is introduced by Kennedy and Eberhart [26] [27] based on the social behavior metaphor. In PSO, a group of particles, without quality and volume, fly through a D-dimensional space, adjusting their positions in search space according to their own experience and their neighbors. It can be imagined a scene in Figure 1. Although approximation methods do not guarantee achieving optimal solutions, they are able to attain near-optimal solutions. PSO has been introduced as an optimization technique in real-number spaces. But many optimization problems are set in a space featuring discrete components. Typical examples include problems that require ordering and route planning, such as in scheduling and routing problems [13].

Recently, the theorem of No Free Lunch (NFL) is proposed for evaluating optimization algorithms by professor Wolpert and Macready of Stanford University [22]. It is shown that there is not a single solution that adapts to all problems effectively. In theory and practice, adopting a single intelligent algorithm is not enough to solve job shop scheduling problems, it is hard to improve the local optimization after some running time of the algorithm, it is necessary to find out a method to escape from this local optimization. Therefore, a hybrid particle swarm optimization algorithm based on TS, SA local search algorithms is proposed.

Some issues are in applying PSO algorithm to solve scheduling problems. The original PSO design is developed to solve continuous functions. The PSO algorithm is problem-independent, which means little specific knowledge revelent to a given problem is required. All we have to know is the fitness evaluation of each solution. This advantage makes PSO more robust than many other search algorithms. However, since PSO is a stochastic search algorithm, it is prove to inadequate global search at the end of a run. PSO may fail to find the required optimum in cases when the problem to be solved is too complicated and complex. TS and SA can employ certain probability to avoid becoming trapped in a local optimum, and the search process can be controlled by the cooling schedule. by designing the neighborhood structure and cooling schedule of TS and SA, the motivation as the Figure 2, we can control the search process and avoid

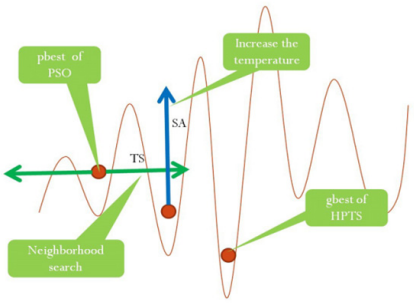

Figure 2: The Motivation of HPTS.

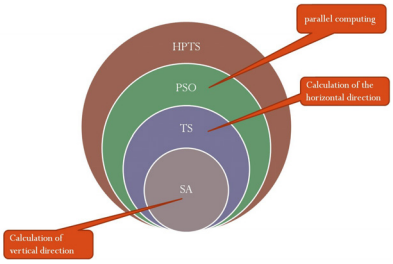

Figure 3: The structure of HPTS.

individuals being trapped in local optimum more efficiently. Therefore, a hybrid algorithm of PSO, TS and SA, named HPTS, is presented in Figure 3.

In the HPTS, three different operations are defined: PSO operation, TS operation and SA operation. PSO combines local search (by self-experience) with global search (by neighboring experience), achieving a high search efficiency. TS uses a memory function to avoid being trapped at a local minimum. It has emerged as an effective algorithmic approach for the FSSP and JSSP. This method can also be referred to as calculation of the horizontal direction. SA employs certain probability to avoid being trapped in a local optimum and the search process can be controlled by the cooling schedule (also known as calculation of vertical direction). The groups use a random initialization, which generates initial particle's initial position and velocity in the search space. Each particle's best position is set to the current location. The current position of the corresponding fitness value is calculated for each particle, according to the evolutionary structure. The hybrid HPTS framework can be converted to the traditional PSO that provides initial solution for TS and SA during the hybrid search process. Such hybrid HPTS strategy retains the advantages of TS and SA, which provides a promising methodology to solve FSSP and JSSP. In addition, HPTS can be applied to many combinatorial optimization problems by simple modification. The description of HPTS is shown in algorithm 1 and algorithm 2.

1) Encoding scheme and initial solution: One of the key issues in applying PSO successfully to FSSP and JSSP is how to encode a schedule to a search solution. Coding scheme also represents the position of particle, including the velocity of particle. In this algorithm, we set up a search space of $n \times m$ dimensions for a problem of n jobs on m machines,

Algorithm 1 PSO

1: **while** the maximum of generation is not met **do**
2: Generation++;
3: Generate next swarm;
4: Find a new local optimum (*gbest*) and a global optimum (*pbest*);
5: Update *gbest* and *pbest*;
6: **end while**

Algorithm 2 TS (SA)

1: Set iteration $iter = iter + 1$, generate neighbors of the current solution $s*$ by a neighborhood structure. If the $s*$ is optimal, then stop;
2: Select the best neighbor which is not tabu or satisfies the aspiration criterion, and store it as the new current solution $s*$, update the tabu list;
3: **for** *gbest* particle s of swarm **do**
4: Tempreture $T_k = T_0$;
5: **while** $T_k \geq T_{end}$ **do**
6: Generate a neiborhood solution $s*$ from s by pair-exchange method;
7: Compute the fitness of $s*$;
8: Evaluate $s*$, $\Delta = f(s^*)\text{-}f(s)$;
9: **if** min[1,exp(-Δ/T_k)] < random[0,1] **then**
10: Accept $s*$;
11: Update the best solution found so far if possible;
12: **end if**
13: **end while**
14: T_k= B * T_{k-1} (weight value parameters are included in B);
15: **end for**
16: If $iter \leq improveiter$ then go to loop;
17: If a termination criterion is satisfied then stop. Otherwise 'pop' a solution on top of the solution stack L, shift the solution to active schedule and install the active solution as the current solution s*, set $iter = 0$, and empty tabu list. Go to step 2;

we adopt job-based representation to represent the position vector of one particle, also named one schedule of all jobs. Suppose that the searching space is D-dimensional and m particles form the swarm. The ith particle represents a D-dimensional vector X_i ($i = 1, 2,..., m$). It means that the ith particle locates at $X_i = (x_{i1}, x_{i2},..., x_{iD})$ ($i = 1, 2,..., m$). The position of each particle is a potential result. We could calculate the particle's fitness by putting its position into a designated objective function. The ith particle "flying" velocity is also a D-dimensional vector, denoted as $V_i = (v_{i1}, v_{i2},..., v_{iD})$ ($i = 1, 2,..., m$). Denote the best position of the ith particle as $P_i = (p_{i1}, p_{i2},..., p_{iD})$, and the best position of the swarm as $G_i = (g_{i1}, g_{i2},..., g_{iD})$, respectively. In addition, initial swarm and initial particle velocities are generated randomly.

2) Parameters setting: The performance of PSO is very sensitive to properly setting the parameters. PSO parameter selection schemes introduced in literature can be classified in some categories. All parameters of PSO are selected empirically. Although some approach may lead to the suboptimal results for a single problem, it would not be helpful as a general approach to solve optimization problems. In this paper, Inertia weight and acceleration coefficient, two

approaches for selecting the value of a parameter is adapted and applied for both classes of proposed algorithms. Which are described as follows:

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{iter_{max}} \times iter \quad (1)$$

Inertia weight (ω) is an important parameter for the search ability of a PSO algorithm. A large inertia weight facilitates searching a new area, while a small inertia weight facilitates fine searching in the current search area. Suitable selection of the inertia weight provides a balance between global exploitation and local exploitation, on average, reduces the number of iterations to find an adequate solution. Therefore, by linearly decreasing the inertia weight form a relatively large value to a relatively small value over the course of operation, the PSO tends to have more global search ability at the beginning of the run. Where ω_{max} is the initial value of weighting coefficient, $\omega_{min}$is the final value of weighting coefficient, $iter_{max}$ is the maximum number of iterations or generation, and $iter$ is the current iteration or generation number.

$$\begin{aligned} V_{id} &= \varphi\{\omega \times V_{id} + C_1 \times \text{Rand}() \times \left(\text{P}_{\text{id}}^{\text{best}} - \text{P}_{\text{id}}\right) \\ &+ C_2 \times \text{Rand}() \times \left(\text{P}_{\text{gid}}^{\text{best}} - \text{P}_{\text{id}}\right)\} \end{aligned} \quad (2)$$

$$\varphi = \frac{2}{|2 - C - \sqrt{C^2 - 4C}|}, C = C_1 + C_2, C > 4 \quad (3)$$

In a swarm, acceleration coefficient C_1 and C_2, know as cognitive and social parameter respectively, control range of particle movement in a single iteration. Although they are not critical for convergence of PSO, assigning different values to C_1 and C_2 sometimes leads to faster convergence and improved performance [17], Typically, these are set to a value of 2.0 [18], but experimental results indicate that alternative configurations, depending on the problem, may produce superior performance. An extended study of the acceleration coefficents in the earliest version of PSO, is given in [19]. M.Clerc [20] introduced a constriction factor to PSO. The constriction factor modified the former Equation (6) to Equation (9) and (10).

3) The neighborhood search: The neighborhood search strategy is directly effective on the efficiency of the local search for the FSSP and JSSP, and unnecessary and infeasible moves must be eliminated if it is possible. Currently, the most well-known neighborhood structures are all based on the concept of blocks. In the HTPS algorithm, taking into account a balance of the effectiveness and efficiency, we ultimately adopt $N6^1$ neighborhood structure, which is introduced by Balas and Vazacopoulos [16].

4) Tabu list: The main task is to avoid the search process turning back to the solutions visited in the previous steps. The elements stored in the tabu list are the attributes of moves, rather than the attributes of solutions for the considered problem. The main purpose of using this attributive representation is to reduce the computational cost. In H-PTS, the size of the list is set the same as the swarm size. During the algorithm running, the tabu list will be updated dynamically.

5) Fitness function: Fitness function is usually used as the performance evaluation of particles in the swarm, which is represented to map an original objective function value to a fitness function that represents relative superiority of particles. In HPTS, each particle's fitness function is expressed by the makespan of the corresponding schedule.

6) Cooling schedule: The SA process can be controlled by the cooling schedule. In general, the cooling schedule is specified by several parameters and/or methods, namely the initial temperature T_0, the rule designated how to lower the temperature, and the termination condition. In proposed algorithm, the initial temperature T_0 is set by $T_0 = \Delta\ f_{max}$, where $\Delta\ f_{max}$ is the maximal difference in fitness values between any two neighboring solutions. We specify the temperature with $T_k = B \times T_{k-1}$ during the kth epoch (k = 1,2,3,...), where B is a parameter called decreasing rate and has a value less than 1.

7) Termination criterion: If one of the following conditions is satisfied: whether the algorithm stops when the algorithm has performed a given total number of iterations, or the elite solution stack has been exhausted, or the solution is proved to be optimal.

3. THE CONVERGENCE OF HPTS

The convergence of intelligent optimization algorithm is an important problem for the application of intelligent optimization algorithm. It is very necessary that we discuss the convergence of the proposed algorithm. We analyze the convergence of PSO algorithm with an optimum keeping strategy and TS, SA algorithms by Markov chain theory at a different aspect in this paper, and HPTS algorithm is proved to be convergent. First of all, we will give an introduction of Markov chain theory as follows.

Markov first studied the stochastic processes that came to be named after him in 1906 [28]. Approximately a century later, there is an active and diverse interdisciplinary community of researchers using Markov chains in comuputer science, physics, statistics, bioinformatics, engineering, and many other areas. Markov chain is a mathematical system that undergoes transitions from one state to another (from a finite or countable number of possible states) in a chainlike manner. It is a random process characterized as memoryless: the next state depends only on the current state and not on the sequence of events that preceded it. This specific kind of "memorylessness" is called the Markov property.

We describe a Markov chain as follows: We have a set of states $S = (s_1, s_2, ..., s_r)$. The process starts in one of these states and moves successively from one state to another. Each move is called a step. If the chain is currently in state s_i, then it moves to state s_j at the next step with a probability denoted by p_{ij} (as well known as transition probabilities), and this probability does not depend upon which states the chain was in before the current state.

Definition 1 (Markov chain) A stochastic sequence (X_n, $n \in T$) and a discrete series $T = 0, 1, 2, ...$ are given, all states values corresponding to each X_n constitute the set of discrete state$S = (s_0, s_1, ..., s_r)$. The stochastic sequence is called Markov chain as soon as the conditional probability satisfies the formula as follow:

$$\begin{aligned} P\{X_{n+1} = S_{n+1} | X_0 = S_0, ...X_n = S_n\} = \\ P\{X_{n+1} = S_{n+1} | X_n = S_n\} \end{aligned} \quad (4)$$

Definition 2 (Transition probability matrix) The conditional probability p_{ij} = P(X_{n+1}= j | X_n= i) is called

transition probability of Markov chain $(X_n,\ n \in T)$, where $ij \in S$. The matrix is called transition probability matrix.

Definition 3 (Finite homogeneous Markov chain) Markov chain is called finite homogeneous Markov chain if conditional probability $p_{ij}(n)$ has nothing to do with n and its set of state S is finite, where. Then $p_{ij}(n)$ is always regarded as p_{ij}.

Theorem 1 The hybrid algorithm PSO, TS and SA algorithm is finite homogeneous Markov chain.

Proof: Since the probability of group in next state rests with the current state, which is independent of the past state, HPTS algorithm with the set of finite state S belongs to Markov chain characteristic. Suppose that P_S, P_A, P_C and P_T are independent of time intervals, then the searching of HPTS can be noted by a transition probability matrix with one step P=$P_T[P_C(P_SP_A)]$, which is independent of time intervals as well. Therefore, the whole searching is finite homogeneous Markov chain.

The neighborhood search and cooling schedule are key factors to impact on the quality and efficiency of HPTS for TS, SA. Moreover, we first give two assumptions about the neighborhood search structure as follows to ensure the convergence of TS and SA.

Assumption 1: The neighborhood search structure is supposed to be symmetrical.

Assumption 2: On the point view of the graph theory, the graph G_N is defined as strongly connected. There must be a path from S_i to S_j for any $S_i, S_j \in S$, where $i, j = 0, 1, 2, ...k$.

Assumption 3: SA algorithm is a homogeneous Markov chain S_{Ti} in which the temperature Ti is held at a slowly decreasing cooling schedule.

Theorem 2 HPTS algorithm with the optimum keeping strategy is global asymptotic convergence when time is endless. Namely the HPTS algorithm will convergence to the optimal group.

proof: Compared with the standard PSO velocity update equation, which has abandoned the previous velocity of particle i, and will make at least one particle of the particle swarm stop evolution of each generation due to its best history position. The optimal strategy algorithm is adopted in the hybrid algorithm. For convenience, the optimal individual reserved from each generation is saved in the left side of the population, but it does not participate in the evolutionary process. The state which contains the same optimal solution is arranged in order as same as which in the original state space, and the one which contains the different optimal solution is arranged in order according to the fitness value. Moreover, The new social collaboration transition probability matrix, self adapting transition probability matrix and competition transition probability matrix can be presented respectively as P_S, P_A, and P_C. After the competition, we will compare the optimal solution of the current population with the optimal solution reserved from the former generation, such an operation is presented. Obviously, we can simply prove that if the search space S of TS is limited, and neighborhood search structure satisfies the above assumption 1 and 2, TS algorithm will converge to optimal solutions inevitably. The transition probability matrix with one step of TS is irreducible stochastic matrix as well. In the optimization context, we can generate an optimal with high probability if we produce a random sample according to the probability distribution. When Ti is very small, the time it takes for the Markov chain S_{Ti} to reach equilibrium can be excessive. SA algorithm will remedy this drawback by using a slowly decreasing cooling schedule. Apparently, the transition probability matrix of HPTS is also irreducible stochastic matrix. Which represents that the probability of individual staying in the non-global optimal solution tends to 0. Therefore, HPTS with the optimum keeping strategy will converge to the optimal group when time is endless.

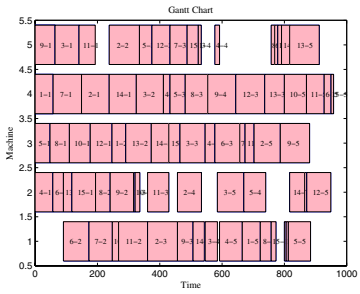

Figure 4: The gantt chart of optimal schedule for LA10.

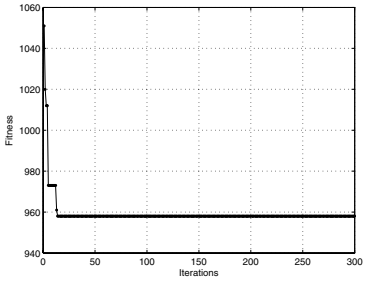

Figure 5: The representative convergence curve for LA10.

4. EXPERIMENTAL RESULTS

To illustrate the effectiveness of the HPTS algorithm for FSSP to minimize the makespan, 30 instances of 10 different sizes taken from Taillard's benchmark (Taillard, 1990) have been selected for simulation experiments. In addition, for JSSP, 43 instances taken from OR-Library [17] as test benchmarks to test our new proposed algorithm. In the 43 instances, FT06, FT10, and FT20 were designed by Fisher and Thompson [18]. Instances LA01-LA40 that were designed by Lawerence [19]. The proposed approach was coded in MATLAB programming language and run on a 2.27 GHz Intel(R) Core(TM)2 Duo CPU, RAM 4GB personal computer. Each instance is executed for 10 runs.

For FSSP, the algorithm ran with the following settings of the control parameters: the population size is 100, which means we use only 100 particles to search solutions for every problem where the number of iterations is 400.

The simulated experimental results are illustrated in Table 1, Figure 6 and Figure 7. In Table 1, PS denotes the problem's size: the number of jobs (Js) multiplies the number of machines (Ms), $FSSP$ represents the Flow-shop Scheduling Problem, WK represents the well-known opti-

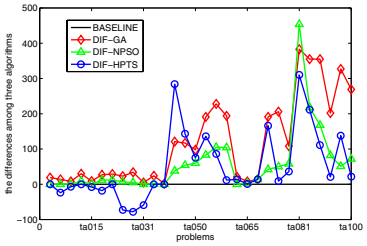

Figure 6: The comparison of three algorithms.

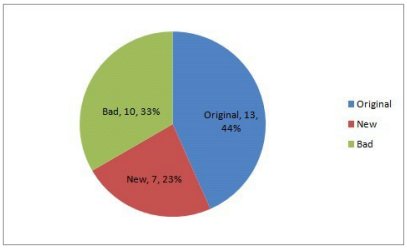

Figure 7: The detailed comparison of experimental results

mal solutions in the benchmark, Min means the best solution found by HPTS, Max means the worst solution found by HPTS, Avg indicates the average value of total run solutions. In Figure 6, these parameters satisfy following conditions: $DIF - HPTS = HPTS$ - WK; $DIF - NPSO = NPSO$ - WK; $DIF - GA = GA$ - WK; $BASELINE = WK$ - WK; $Original$ represents the some results are equal to WK. New indicates the best solutions are better than WK. and Bad means the bad results are worse than WK. The detailed comparison of experimental results is shown in Figure 7.

Table 1 shows the experimental results of different benchmark instances. The comparison of the three different algorithms is shown in Figure 6. According to the values of Min, Max and Avg, and also the curve about GA, NPSO, and HPTS, we have an observation that the HPTS outperforms the NPSO and GA algorithms in the total solution quality thoroughly.

Figure 6 shows the comparison of HPTS with GA and NPSO. It can be observed obviously that our HPTS algorithm outperforms the NPSO and GA algorithms in some solution quality thoroughly (In particular, ta005, ta010, ta015, ta020, ta025, ta030 and ta031). HPTS can achieve the optimal value in the search space quickly with smaller population size, making use of its better local searching ability to get the final optimal solution at the end of evolution. More important is 7, HPTS can find the WK in 20 cases (67 %) among the 30 instances, the better (New) results are found in 7 instances (ta005, ta010, ta015, ta020, ta025, ta030 and ta031). It is within 23 % of the percent average objective value over than WK, which demonstrates HPTS algorithm has the powerful explore-ability. 44 % of the percent average objective value equal to WK.

For JSSP, the algorithm ran with the following settings of the control parameters: $K = 300$ (iterations), $N = 400$ (the number of particles), $w_{max} = 0.4$, $w_{min} = 0.2$ (inertia weight), $c_1 = c_1 = 0.2$ (acceleration coefficient), w_{min} =0.2 (inertia weight), T_0 =10 (initial temperature). The proper values of these parameters were experimentally determined in a pre-processing phase. Specifically, the performance of the proposed HPTS was compared against that of Goncalves' HGA [3], Tsung-Lieh Lin's MPSO [20], Wei-Jun Xia's HPSO [21].

Due to the stochastic behavior of these algorithms, and the fact that none of them has a natural termination point, it was decided to run the algorithms for the same fixed time duration and report the best solution obtained after this time has elapsed. The benchmarks range from small size instances with 6 jobs and 6 machines to large size instances with 30 jobs and 10 machines. Specifically, The LA10 problem is described as an example to illustrate the simulated experiment results more intuitively, in which there are numerous local optima so that the problem is challenging enough. Figure 4 shows the gantt chart of optimal schedule for LA10 and Figure 5 describes the representative convergence curve for LA10.

In Table 2, $Problem$ denotes the JSSP problems, $Size$ represents the Problem size n jobs on m machines. BKS represents the well-known optimal solutions in the benchmark. Among 43 instaces, HPTS can find the well-known optimal solutions with 40 instances that is much better than HGA (33), MPSO (36), and HPSO (35). Moreover, in about 93 % of the instances, and the deviation of the minimum found makespan from the well-known optimal solutions is only on average 0.4 %. The proposed algorithm yields a significant improvement in solution quality with respect to almost all the other algorithms, except for the HGA, MPSO and HPSO approaches that has a better performance in the LA29, LA36, LA37, LA38, LA40 instances mainly. For instances LA24, LA25, LA27, LA28, LA29, LA36, LA37, LA38, LA39, LA40, the deviation between the best minimum founded solution and the best known solution are all less than the deviation results of HGA. HPTS can obtain better solution for instances LA24, LA27, LA29, LA36, LA37, LA38, LA40 than MPSO. Except for LA29, HPTS also can obtain better solution (LA24, LA36, LA37, LA38, LA39, LA46) than HPSO.

Obviously, the two experimental results show that HPTS is more efficient than other existing discrete particle swarm optimization and genetic algorithms, respectively. The superior results also indicate the successful incorporation of the improved PSO, TS and SA, which facilitates the escape from local minimum points and increases the possibility of finding a better solution. Therefore, the computational results show that the proposed algorithm could obtain the high-quality solutions within relatively short computation time. So, it is a robust, fast and simply hybrid optimization algorithm.

5. CONCLUSIONS

There is no argument that the most important feature of FSSP and JSSP are its efficiency. Therefore, the majority of the research work done in the intelligent manufacturing community is about achieving global optimum. We are motivated to find high-quality solutions in a reasonable computation time by exploiting Particle Swarm Optimization (PSO), Tabu Search (TS) and Simulated Annealing (SA). We propose a new multi-structural hybrid evolutionary framework, and derive HPTS algorithms as its extension. Extensive experiments on different scale benchmarks validate the effectiveness of our approaches, compared with other well-

Table 1: The experimental results 1

PS(Js*Ms)	FSSP	WK	NPSO	GA	HPTS
			Min Avg Max	Min Avg Max	Min Avg Max
20*5	ta001	1278	1278/1295.1/1297	1297/1297.0/1297	1278/1278.0/1278
20*5	ta005	1236	1236/1247.9/1250	1250/1251.2/1258	**1212**/1229.7/1247
20*5	ta010	1108	1108/1115.4/1127	1116/1135.6/1161	**1101**/1107.5/1114
20*10	ta011	1582	1591/1604.7/1627	1612/1623.7/1645	1582/1585.7/1589
20*10	ta015	1419	1419/1428.1/1444	1428/1457.4/1478	**1411**/1415.0/1419
20*10	ta020	1591	1603/1621.0/1633	1618/1632.8/1654	**1573**/1582.3/1591
20*20	ta021	2297	2309/2324.8/2339	2326/2344.8/2375	2297/2297.0/2297
20*20	ta025	2291	2298/2312.2/2334	2314/2330.6/2351	**2218**/2255.0/2291
20*20	ta030	2178	2183/2215.1/2244	2212/2237.5/2282	**2100**/2139.0/2178
50*5	ta031	2724	2724/2729.5/2742	2729/2739.6/2752	**2665**/2694.5/2724
50*5	ta035	2863	2864/2864.0/2864	2887/2916.3/2947	2863/2863.0/2863
50*5	ta040	2782	2782/2783.2/2786	2784/2815.7/2832	2782/2782.5/2783
50*10	ta041	3025	3063/3098.6/3138	3146/3186.7/3227	3309/2724.0/3366
50*10	ta045	2986	3040/3078.7/3129	3103/3168.4/3194	3129/3221.0/3311
50*10	ta050	3091	3151/3171.7/3204	3189/3225.5/3280	3166/3170.0/3174
50*20	ta051	3875	3958/3991.5/4011	4066/4105.3/4141	4011/4143.5/4276
50*20	ta055	3635	3740/3773.7/3812	3863/3915.9/3975	3721/3803.5/3886
50*20	ta060	3777	3881/3959.8/4034	3971/4008.6/4093	3789/3895.0/4001
100*5	ta061	5493	5493/5494.0/5495	5514/5524.5/5541	5507/5509.8/5512
100*5	ta065	5250	5253/5256.8/5267	5258/5302.5/5336	5251/5252.0/5253
100*5	ta070	5328	5342/5345.8/5368	5342/5372.3/5403	5342/5342.0/5342
100*10	ta071	5770	5812/5842.8/5869	5961/6027.2/6095	5936/6080.5/6225
100*10	ta075	5468	5518/5578.4/5636	5674/5769.2/5850	5477/5497.5/5518
100*10	ta080	5845	5903/5914.5/5962	5953/6056.3/6106	5881/5892.0/5903
100*20	ta081	6286	6470/6544.4/6615	6669/6763.9/6843	6596/6834.0/7072
100*20	ta085	6377	6595/6653.6/6723	6732/6816.3/6898	6588/6731.0/6874
100*20	ta090	6465	6633/6723.3/6821	6820/6910.3/6979	6576/6824.0/7072
200*10	ta091	10868	10950/10978.9/11005	11070/11112.1/11168	10889/11017.0/11145
200*10	ta095	10524	10575/10664.0/10764	10851/10917.7/11029	10662/10721.5/10781
200*10	ta100	10676	10748/10796.9/10850	10945/11025.6/11101	10698/10735.5/10773

established methods. The experimental results show that new upper bounds of the unsolved problems are achieved in a relatively reasonable time. For example, in 30 and 43 benchmarks, 7 new upper bounds and 6 new upper bounds are obtained by the HPTS algorithm, respectively. On the other hand, we analyze the convergence of PSO algorithm with an optimum keeping strategy and TS, SA algorithms by Markov chain theory at a different aspect in this paper, and HPTS algorithm is proved to be convergent.

6. ACKNOWLEDGMENTS

This paper is supported by the research fund JSPS KAKENHI (20240003, 21300054), Xue-Feng Zhang is sponsored by the China Scholarship Council (CSC).

7. REFERENCES

[1] M. Garey,D. Johnson, R. Sethi, The complexity of flow shop and job shop scheduling. Mathematics of Operations Research, 1: 117-129, 1976.

[2] I-Hong Kuo, Shi-Jinn Horng, An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. Expert Systems with Applications 36: 7027-7032, 2009.

[3] J.F. Goncalves, J.J.D.M Mendes, M.G.C. Resende, A hybrid genetic algorithm for job shop scheduling. European Joural of Operational Research, 167: 77-95, 2005.

[4] E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem. Euopean Journal of Operational Research, 47, 65-74 (1990)

[5] F. A. Ogbu, D. K. Smith, The application of the simulated annealing algorithm to the solution of the n/m/Cmax flow shop problem. Computers and Operations Research. 17: 243-253, 1990

[6] J. Adams, E. Balas, D. Zawack, The shifting bottleneck procedure for job shop scheduling. Management Science, 34: 391-401, 1988.

[7] E. Nowicki, C. Smutnicki, An advanced tabu search algorithm for the job shop problem. Journal of Scheduling, 8: 145-159, 2005.

[8] R.K. Sursh, A. Albrecht, K.C. Wong, Pareto archived simulated annealing for job shop scheduling with multiple objectives. The International Journal of Advanced Manufacture Technology, 29: 184-196, 2005.

[9] H.W. Ge, L. Sun, Y.C. Liang, An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling. IEEE Transactions on Systems, Man and Cybernetics, Part A, 38: 358-368, 2008.

[10] A. Udomsakdigool, V. Kachitvichyanukul, Multiple colony ant algorithm for job-shop scheduling. International Journal of Production Research, 46: 4155-4175, 2008.

[11] S. He, Q. H.Wu, J. Y. Wen, J. R. Saunders, R.C.

Paton, A particle swarm optimizer with passive congregation. BioSystems, 78: 135-147, 2004.

[12] D. Erkan, E. Ferhat, P.S. Mehmet, Optimum Design of Grillage Systems under Code Provisions using Particle Swarm Optimization. GECCO'10, July 7-11, Portland, Oregon, USA, 2010.

[13] D.Y. Sha, L. Hsinghung, A particle swarm optimization for multi-objective flowshop scheduling. Original Article Int J Adv Manuf Technol, (45):749-758, 2009.

[14] Y. Shi, R. Eberhart, Empirical study of particle swarm optimization. In: Proceedings of Congress on Evolutionary Computation, 1945-1950, 1999.

[15] J. Kennedy, The particle swarm: social adaptation of knowledge. In: Proceedings of IEEE International Conference on Evolutionary Computation, 303-308, 1997.

[16] E. Balas, A. Vazacopoulos, Guided local search with shifting bottleneck for job shop scheduling. Management Science, (44): 62-75, 1988.

[17] J.E. Beasley, Or-library: Distributing test problems by electronic mail. Journal of the Operational Research Society, (14): 1069-1072, 1990.

[18] H. Fisher, G.L. Thompson, Probabilistic learning combinations of local job shop scheduling rules. Prentice-Hall, NJ: Englewood Cliffs, 225-251, 1963.

[19] S. Lawrence, An experimental investigation of heuristic scheduling techniques. In supplement to resource constrained project scheduling, GSIA, Pittsburgh, PA: Carnegie Mellon University, 1984.

[20] L. Tsung-Lieh, H. Shi-Jinn, An effiecient job-shop scheduling algorithm based on particle swarm optimization, Expert Systems with Applications, 37: 2629-2636, 2010.

[21] X. Weijun, W. Zhiming, A hybrid particle swarm optimization approach for the job-shop scheduling problem. Int J Adv Manuf Technol. 29: 360-366, 2006.

[22] S. Xiaoyu, Hybrid Particle Swarm Algorithm for job shop scheduling Prolems. Future Manufacturing Systems : 235-268, 2008.

[23] D.E. Knuth, The TEXbook, Addison-Wesley, 1989.

[24] V. Fevrier, M. Patricia, C. Oscar, M. Oscar, "A New Evolutionary Method with a Hybrid Approach Combining Particle Swarm Optimization and Genetic Algorithms using Fuzzy Logic for Decision Making," IEEE Congress on Evolutionary Computation, 2008, pp. 1333-1339

[25] V. Fevrier, M. Patricia, C. Oscar, "An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms," Appl. Soft Comput, 2011, 11(2): 2625-2632

[26] J. Kennedy, R. Eberhart, "Particle swarm optimization," In Proceedings of IEEE international conference on neural network , 1995, pp.1942-1948.

[27] R. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory," In: Proceedings of Sixth International Symposium on Micro Machine and Human Science, Japan, 1995, pp. 39-43.

[28] B. Dimitris, T. John, Simulated Annealing. Statistical Science. 1 : 10-15, 1993.

Table 2: The experimental results 2

Problem	BKS	HGA	MPSO	HPSO	HPTS
FT06	55	55	55	55	55
FT10	930	930	930	930	930
FT20	1165	1165	1165	1178	1165
LA01	666	666	666	666	666
LA02	655	655	655	655	655
LA03	597	597	597	597	597
LA04	590	590	590	590	590
LA05	593	593	593	593	593
LA06	926	926	926	926	926
LA07	890	890	890	890	890
LA08	863	863	863	863	863
LA09	951	951	951	951	951
LA10	958	958	958	958	958
LA11	1222	1222	1222	1222	1222
LA12	1039	1039	1039	1039	1039
LA13	1150	1150	1150	1150	1150
LA14	1292	1292	1292	1292	1292
LA15	1207	1207	1207	1207	1207
LA16	945	945	945	945	945
LA17	784	784	784	784	784
LA18	848	848	848	848	848
LA19	842	842	842	842	842
LA20	902	902	902	907	902
LA21	1046	1046	1046	1046	1046
LA22	927	927	927	927	927
LA23	1032	1032	1032	1032	1032
LA24	935	953	941	938	**935**
LA25	977	986	977	977	977
LA26	1218	1218	1218	1218	1218
LA27	1235	1256	1239	1236	**1236**
LA28	1216	1232	1216	1216	1216
LA29	1152	1196	1173	1164	**1166**
LA30	1355	1355	1355	1355	1355
LA31	1784	1784	1784	1784	1784
LA32	1850	1850	1850	1850	1850
LA33	1719	1719	1719	1719	1719
LA34	1721	1721	1721	1721	1721
LA35	1888	1888	1888	1888	1888
LA36	1268	1279	1278	1269	**1268**
LA37	1397	1408	1411	1401	**1399**
LA38	1196	1219	1208	1208	**1201**
LA39	1233	1246	1233	1240	**1233**
LA40	1222	1241	1225	1226	**1222**