Combining PSO and Local Search to Solve 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

Intelligent manufacturing is associated with a large number of complex optimization problems and for this reason has got a considerable research attention over the last decades. Most of these problems are of combinatorial nature and have been proved to be NP-complete. This paper deals with the flow shop scheduling problem (FSSP) and the Job Shop Scheduling Problem (JSSP). The objective of these problems is to find an appropriate sequence to minimize the makespan, which are defined as the time for completing a final operation. One major challenging issue is how to obtain the high-quality global optimum. In order to refrain from the premature convergence and being easily trapped into local 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 algorithm as its extension. Extensive experiments on different scale benchmarks validate the effectiveness of our approaches, compared with other well-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 Tailland's and 43 OR-Library benchmarks, 7 new upper bounds and 6 new upper bounds are obtained by the HPTS algorithm, respectively.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

Categories and Subject Descriptors

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

General Terms

Theory, Algorithms, Performance.

Keywords

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

1. INTRODUCTION

In today's complex manufacturing setting, with multiple lines of products, each requiring many different steps and machines for completion, the decision maker for the manufacturing plant must find a way to successfully manage resources in order to produce products in the most efficient way possible. The decision maker needs to design a production schedule that promotes on-time delivery, and minimizes objectives such as the flow time of a product. Out of these concerns grew an area of studies known as the scheduling problems. Intelligent manufacturing is associated with a large number of complex optimization problems and for this reason has got a considerable research attention over the last decades. Most of these problems are of combinatorial nature and have been proved to be NP-complete.

Flow-Shop Scheduling Problem (FSSP) is a combinatorial optimization problem and NP-complete [6]. 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. Every job has M operations, and every machine has N jobs. Every job executes its operations on every machine in the order of $M = \{M_1, \cdots M_m\}$. Every operation of a job cannot be preempted. Every machine can execute only a single operation of a job at a time [12]. This problem can be given as

follows:

$$C(1,1) = T(1,1) \quad (1)$$

$$C(1,i) = C(1,i-1) + T(1,i) \quad (2)$$

$$C(r,1) = C(r-1,1) + T(r,1) \quad (3)$$

$$C(r,i) = \max(C(r,i-1), C(r-1,i)) + T(r,i) \quad (4)$$

where $T(r,i)$ denotes the execution time of the rth operation of the ith job on machine M_r, and $C(r,i)$ represents the maximum completion time of the ith job on machine M_r, $1 \leq r \leq m$, and $1 \leq i \leq n$.

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 [6]. 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 maybe different for any job. In general, one job being processed on one machine is considered as one operation as noted above, 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.

Let $J = 0, 1, ..., n, n+1$ denote the set of operations to be scheduled and $M = 0, 1, ..., m, m+1$ the set of machines. The operations 0 and $n+1$ are dummy, have no duration and represent the initial and final operations. The operations are interrelated by two kinds of constraints. Firstly, the precedence constraints, which force each operation j to b scheduled after all predecessor operations, P_i, are completed. Secondly, operation j can only be scheduled if the machine it requires is idle. Further, let dj denote the fixed duration (processing time) of operation j.

Let F_j represent the finish time of operation j. A schedule can be represented by a vector of finish times $(F_1, ...F_m, F_{m+1})$. Let $A(t)$ be the set of operations being processed at time t, and let $r_{jm} = 1$ if operation j requires machine m to be processed and $r_{jm} = 0$ otherwise.

The conceptual model of the JSSP can be described in the following way [8]:

$$Minimize F_{n+1}(C_{max}) \quad (5)$$

$$F_k \leq F_j - d_j, j = 1, ...n+1, k \in P_j \quad (6)$$

$$\sum_{j \in A(t)} r_{jm} \leq 1, m \in M, t \geq 0 \quad (7)$$

$$F_j \geq 0, j = 1, ...n+1 \quad (8)$$

The objective of Eq.5 minimizes the finish time of operation $n+1$ (the last operation), and therefore minimizes the makespan. Constraints Eq.6 impose the precedence relations between operations and constraints in Eq.7 state that one machine can only process one operation at a time. Finally Eq.8 forces the finish times to be nonnegative.

In the past few decades, these problems have attracted many researchers. Many heuristic algorithms have been proposed, such as Taillard's tabu search method [20], Ogbu and Smith's simulated annealing algorithm [16], shifting bottleneck approach (SB) [1], tabu search algorithm(TS) [15], genetic algorithm(GA) [8], simulated annealing(SA) [19], particle swarm optimization(PSO) [7], and ant colony(ACO) [21]. 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 [9].

Most studies indicate that a single technique can not solve this stubborn problem. Much work has been done on hybrid methods involving GA,SA,TS,and SB techniques as hybrid methods can provide high-quality solution within reasonable computing time. A comprehensive survey of scheduling techniques can be seen from Blazevicz et al. and Jain and Meeran. In this paper, we focus on exploiting particle swarm optimization algorithm to achieve the better solution for FSSP and JSSP.

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. In Section 2, we present and analyze the multi-structural hybrid evolutionary framework. Section 3 gives experimental results of the HPTS and other competitive approaches. Finally, the main conclusions are drawn.

2. PROBLEM FORMULATION

2.1 Particle Swarm Optimization

The particle swarm concept was based on the premise of social behavior. The original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock. A PSO algorithm mimics the behavior of flying birds and their means of information exchange to solve optimization problems. It is a simulator of social behavior that is used to realize the movement of a birds' flock [4]. 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 [17]. They are described as follows [18]:

$$\begin{aligned} V_{id} &= \omega \times V_{id} + C_1 \times \text{Rand}() \times \left(\text{P}_{\text{id}}^{\text{best}} - \text{P}_{\text{id}}\right) \\ &+ C_2 \times \text{Rand}() \times \left(\text{P}_{\text{gid}}^{\text{best}} - \text{P}_{\text{id}}\right) \end{aligned} \quad (9)$$

$$P_{id} = P_{id} + V_{id} \quad (10)$$

where V_{id} represents the velocity of particle i. It also can be regarded as the distance to be traveled by particle i from its current position. P_{id} represents the particle position, P_{id}^{best} called "pbest", the local best solution, represents particle i's best previous position, and P_{gid}^{best} called "gbest", the global best solution, represents the best position among all particles in the swarm. ω is an inertia weight. It regulates the trade-off between the global exploration and local exploitation abilities of the swarm. The acceleration constants C_1

and C_2 represent the weights of the stochastic acceleration terms that pull each particle toward "pbest" and "gbest" positions. Rand() is a random function with range $[0, 1]$.

For Eq.9, the first part represents the inertia of previous velocity; the second part is the "cognition" part, which represents individuals thinking independently; and the third part is the "social" part, which represents cooperation among the particles [11].

2.2 The Multi-Structural Hybrid Evolutionary Framework

In the framework, 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. 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 multi-structural hybrid evolutionary framework can be converted to the traditional PSO that provides initial solution for TS and SA during the hybrid search process. Such hybrid strategy retains the advantages of TS and SA, which provides a promising methodology. In addition, it can be applied to many combinatorial optimization problems by simple modification. The description of the framework is shown in algorithm 1 and algorithm 2. The process of implementing the framework is as follows:

Step 1: Initialize a swarm of particles with random position and velocities in the D-dimensional problem space.

Step 2: For each particle, evaluate the desired optimization fitness function.

Step 3: Individual particle's fitness value are with D-dimensional problem space . If the current value is better than it, then set the position equal to the current value, and the position equal to the current position in the D-dimensional problem space.

Step 4: Compare the fitness evaluation value with the swarm's best fitness value obtained so far. If current value is better than , then reset to the current particle's fitness value.

Step 5: Change the velocity and position of the particle according to Eq.9 and Eq.10, respectively.

Step 6: Loop to step (2) until a termination criterion is met, usually a sufficiently good fitness or a specified number of generations.

Algorithm 1 PSO

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

Algorithm 2 TS (SA)

Set iteration $iter = iter + 1$, generate neighbors of the current solution $s*$ by a neighborhood structure. If the $s*$ is optimal, then stop;
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;
for *gbest* particle s of swarm **do**
  Tempreture $T_k = T_0$;
  while $T_k \geq T_{end}$ **do**
    Generate a neiborhood solution $s*$ from s by pair-exchange method;
    Compute the fitness of $s*$;
    Evaluate $s*$, $\Delta = f(\text{s*})\text{-}f(\text{s})$;
    if $\min[1,\exp(-\Delta/T_k)] < \text{random}[0,1]$ **then**
      Accept $s*$;
      Update the best solution found so far if possible;
    end if
  end while
  $T_k = \text{B} * T_{k-1}$ (weight value parameters are included in B);
end for
If $iter \leq improveiter$ then go to loop;
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;

2.3 Implementation of Proposed Algorithms

Some issues are in applying PSO algorithm to solve scheduling problems. The original PSO design is developed to solve continuous function. The PSO algorithm is problem-independent, which means little specific knowledge relevent 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 prone to inadequate global search ability 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, we can control the search process and avoid individuals being trapped in local optimum more efficiently. Therefore, a hybrid algorithm of PSO, TS and SA, named HPTS, is presented in Figure 1.

1) Encoding scheme and initial solution: 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.

2) The neighborhood search: The neighborhood search

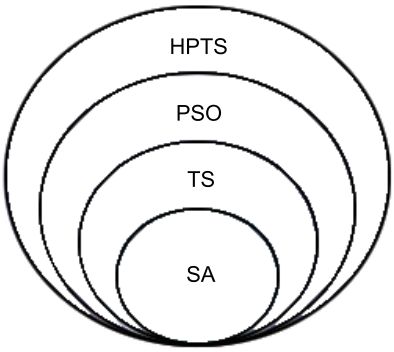

Figure 1: The structure of HPTS.

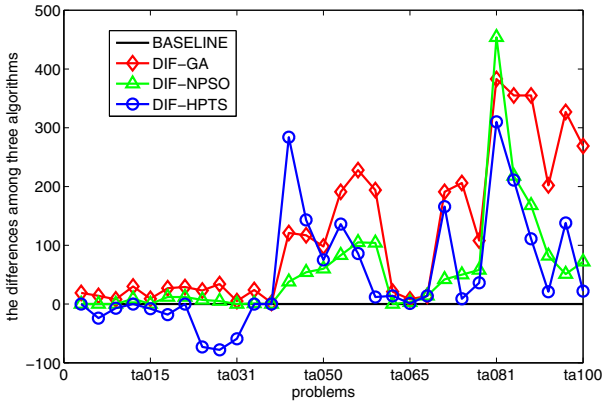

Figure 2: The comparison of three algorithms.

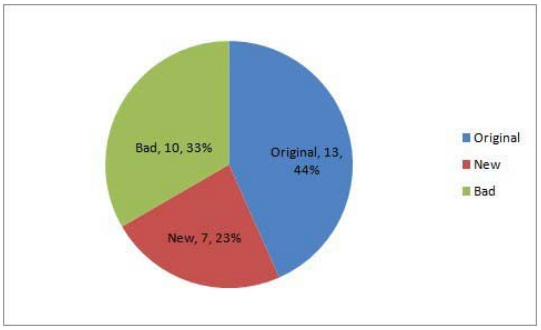

Figure 3: The detailed comparison of experimental results

strategy is directly effective on the efficiency of the local search for the FSSP and JSSP, 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 [2].

3) 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.

4) 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.

5) 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.

6) 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. EXPERIMENTAL RESULTS

3.1 Experimental results 1

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. This benchmark contains some instances that have been proven to be very difficult to solve in the sense that the best solutions found so far are through the use of a very lengthy Tabu-search heuristic.

In HPTS, 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. In order to compare the convergence rate with GA and NPSO [22], we run the HPTS algorithm 10 times for every problem. The proposed two algorithms were implemented in MATLAB and simulated in a platform with 2.27 GHz Intel(R) Core(TM)2 Duo CPU, RAM 4GB.

The simulated experimental results are illustrated in Table 1, Figure 2 and Figure 3. In Table 1, PS denotes the problem's size: the number of jobs (Js) multiplied by the number of machines (Ms), $FSSP$ represents the Flow-shop Scheduling Problem, WK represents the well-known optimal 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 2, these parameters satisfy the 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.

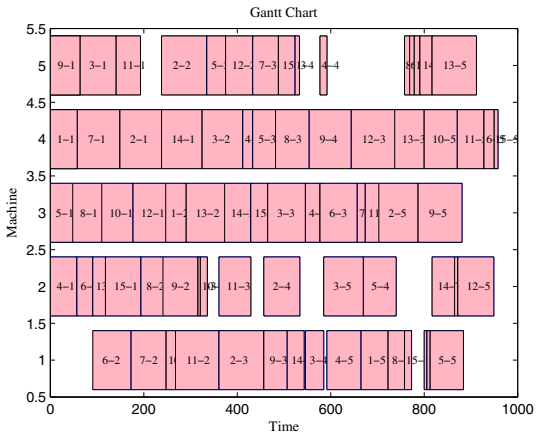

Figure 4: The gantt chart of optimal schedule for LA10.

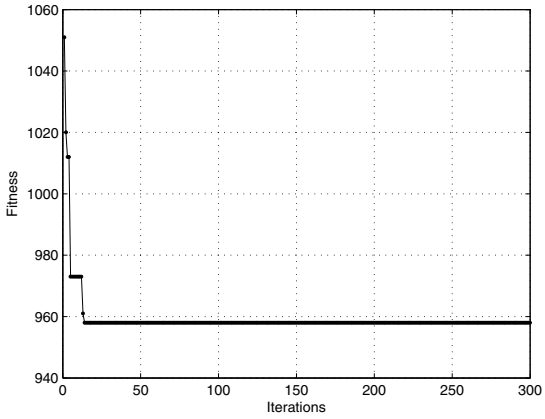

Figure 5: The representative convergence curve for LA10.

The detailed comparison of experimental results is shown in Figure 3.

Table 1 shows the experimental results of different benchmark instances. The comparison of the three different algorithms is shown in Figure 2. 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 2 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 Figure 3, in which 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.

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.

3.2 Experimental results 2

To illustrate the effectiveness of the HPTS algorithm for JSSP to minimize the makespan, 43 instances taken from OR-Library [3] as test benchmarks to test our new proposed algorithm. In the 43 instances, FT06, FT10, and FT20 were designed by Fisher and Thompson [5]. Instances LA01-LA40 that were designed by Lawerence [13].

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. 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 [8], Tsung-Lieh Lin's MPSO [14], Wei-Jun Xia's HPSO [10].

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 fix 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 optimum 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 Tabel 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 instances, 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 bet-

ter solution (LA24, LA36, LA37, LA38, LA39, LA46) than HPSO.

Obviously, the 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.

4. CONCLUSIONS

There is no argument with 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-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 Tailland's and 43 OR-Library benchmarks, 7 new upper bounds and 6 new upper bounds are obtained by the HPTS algorithm, respectively.

For the future work, there are still many problems that are not well solved with the HPTS algorithm. For instance, currently the decision strategy of hybrid algorithm design is still an art instead of a science. It is very hard to explain why a certain decision strategy works better than other approaches. It is still not well understood why some heuristic algorithms work with some classes of instances but not others. There are other potentially useful features in the HPTS algorithms. We plan to apply parallel and distributed computing to solve other NP problems.

5. ACKNOWLEDGMENTS

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

6. ADDITIONAL AUTHORS

7. REFERENCES

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

[2] E. Balas and A. Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44:62–75, 1988.

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

[4] D. Erkan, E. Ferhat, and P. Mehmet. Optimum design of grillage systems under code provisions using particle swarm optimization. *GECCO'10*, 78:135–147, July 2010.

[5] H. Fisher and G. Thompson. Probabilistic learning combinations of local job shop scheduling rules. *Prentice-Hall, NJ: Englewood Cliffs*, pages 225–251, 1963.

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

[7] H. W. Ge, L. Sun, and Y. C. Liang. An effective pso and ais-based hybrid intelligent algorithm for job-shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 38:358–368, 2008.

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

[9] S. He, Q. H.Wu, J. Y. Wen, J. R. Saunders, and R. Paton. A particle swarm optimizer with passive congregation. *BioSystems*, 78:135–147, 2004.

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

[11] J. Kennedy. The particle swarm: social adaptation of knowledge. *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 303–308, 1997.

[12] I. H. Kuo and S. J. Horng. An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. *Expert Systems with Applications*, 36:7027–7032, 2009.

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

[14] L. T. Lieh and H. S. Jinn. An effiecient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37:2629–2636, 2010.

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

[16] F. A. Ogbu and 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.

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

[18] Y. Shi and R. Eberhart. Empirical study of particle swarm optimization. *Proceedings of Congress on Evolutionary Computation*, pages 1945–1950, 1999.

[19] R. K. Sursh, A. Albrecht, and 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.

[20] E. Taillard. Some efficient heuristic methods for the

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

flow shop sequencing problem. *European Joural of Operational Research*, 47:65–74, 1990.

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

[22] C. S. Zhang, J. X. Ning, and D. T. Ouyang. A hybrid alternate two phases particle swarm optimization algorithm for the flow shop scheduling problem. *Computers and Industrial Engineering*, 58:1–11, 2010.

Table 2: The experimental results 2

Problem	Size	BKS	HGA	MPSO	HPSO	HPTS
FT06	6 × 6	55	55	55	55	55
FT10	10 × 10	930	930	930	930	930
FT20	20 × 5	1165	1165	1165	1178	1165
LA01	10 × 5	666	666	666	666	666
LA02	10 × 5	655	655	655	655	655
LA03	10 × 5	597	597	597	597	597
LA04	10 × 5	590	590	590	590	590
LA05	10 × 5	593	593	593	593	593
LA06	15 × 5	926	926	926	926	926
LA07	15 × 5	890	890	890	890	890
LA08	15 × 5	863	863	863	863	863
LA09	15 × 5	951	951	951	951	951
LA10	15 × 5	958	958	958	958	958
LA11	20 × 5	1222	1222	1222	1222	1222
LA12	20 × 5	1039	1039	1039	1039	1039
LA13	20 × 5	1150	1150	1150	1150	1150
LA14	20 × 5	1292	1292	1292	1292	1292
LA15	20 × 5	1207	1207	1207	1207	1207
LA16	10 × 10	945	945	945	945	945
LA17	10 × 10	784	784	784	784	784
LA18	10 × 10	848	848	848	848	848
LA19	10 × 10	842	842	842	842	842
LA20	10 × 10	902	902	902	907	902
LA21	15 × 10	1046	1046	1046	1046	1046
LA22	15 × 10	927	927	927	927	927
LA23	15 × 10	1032	1032	1032	1032	1032
LA24	15 × 10	935	953	941	938	**935**
LA25	15 × 10	977	986	977	977	977
LA26	20 × 10	1218	1218	1218	1218	1218
LA27	20 × 5	1235	1256	1239	1236	**1236**
LA28	20 × 5	1216	1232	1216	1216	1216
LA29	20 × 5	1152	1196	1173	1164	**1166**
LA30	20 × 5	1355	1355	1355	1355	1355
LA31	30 × 10	1784	1784	1784	1784	1784
LA32	30 × 10	1850	1850	1850	1850	1850
LA33	30 × 10	1719	1719	1719	1719	1719
LA34	30 × 10	1721	1721	1721	1721	1721
LA35	30 × 10	1888	1888	1888	1888	1888
LA36	15 × 15	1268	1279	1278	1269	**1268**
LA37	15 × 15	1397	1408	1411	1401	**1399**
LA38	15 × 15	1196	1219	1208	1208	**1201**
LA39	15 × 15	1233	1246	1233	1240	**1233**
LA40	15 × 15	1222	1241	1225	1226	**1222**