Timetable Scheduling Using Particle Swarm Optimization

Shu-Chuan Chu, Yi-Tin Chen
Department of Information Management
Cheng-Shiu University
840 ChengCing Rd., NiaoSong Township,
Kaohsiung County 833, Taiwan
scchu@csu.edu.tw

Jiun-Huei Ho
Department of Electronic Engineering
Cheng-Shiu University
840 ChengCing Rd., NiaoSong Township,
Kaohsiung County 833, Taiwan
jhho@csu.edu.tw

Abstract

In timetable scheduling problems, examination subjects must be slotted to certain times that satisfy several of constraints. They are NP-completeness problems, which usually lead to satisfactory but suboptimal solutions. As PSO has many successful applications in continuous optimization problems, the main contribution of this paper is to utilize PSO to solve the discrete problem of timetable scheduling. Experimental results confirm that PSO can be utilized to solve discrete problem as well.

1. Introduction

Timetable scheduling problems [1, 2] are problems of time-based planning and combinatorial optimization that tend to be solved with a cooperation of search and heuristics, which usually lead to satisfactory but suboptimal solutions. There are many kinds of timetable scheduling problems in the daily life, such as examination, lecture, and transportation timetables. In timetable scheduling problems, events must be slotted to certain times which satisfy several of constraints. It is difficult to design knowledge-based and OR-based algorithms to solve such problems. On the other hand, constraint satisfaction techniques have been used to solve hard constraints; however, it is more difficult to handle soft constraints such as preferences. PSO algorithm is used to schedule exam timetables and the main issues we are going to address are as follows:

- The quality of the examination timetable.
- The times spent in producing the timetable.

Particle swarm optimization (PSO) [3, 4, 5] is an alternative population-based evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [6, 7, 8]. Unlike GA, PSO has no evolution operators such as crossover and mutation. Therefore, PSO attempts to simulate social behavior, which differs from the natural selection schemes of genetic algorithms. In PSO, the population consists of potential solutions, called particles, similar to birds in a flock. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. While flying, every particle updates its velocity and position based on its own best experience and that of the entire population. The updating policy will cause the particle swarm to move toward a region with a higher object value.

2. Problem description

In this specific timetable scheduling problem, the events are exams, and the times are the separate time slots. There are three days for the allocation of the exams. Each day contains four time slots, two are in the morning and another two are in the afternoon. Table 1 illustrates the exam timetable framework used for this paper, where $1, 2, 3, \cdots, 12$ are time slots. A student may take different number of exams because they may take different number of courses. We consider this problem in terms of maximum number of exams per student in the following three cases:

- a student takes 11 exams,
- a student takes 13 exams, and
- a student takes 15 exams.

The minimum number of exams and the number of students are 9 and 50, respectively. The most important constraint of the exam timetable is that the same student cannot take two different exams at the same time. In other words, the exams cannot clash for any student. This is the only hard constraint. However, we also consider the several soft constrains:

- Very strongly prefer no more than two exams per day for any student.
- Strongly prefer not to have exams consecutive on the same half day for a student.
- Prefer not to have exams consecutive on different half days of the same day for the same student.

Table 1. Time slots allocation for examination

time	AM		PM	
	8:30 10:00	10:30 12:00	13:30 15:00	15:30 17:00
1st day	1	2	3	4
2nd day	5	6	7	8
3rd day	9	10	11	12

3. Particle swarm optimization (PSO)

PSO is a methodology that is based on social behavior of evolution, which means it is naturally not alike those methodologies that use natural evolution as the weeding-out process [6, 7, 8]. PSO processes the search system using populations of particles, which corresponds to the use of individuals in genetic algorithms. Each particle is updated by following two best values in every iteration. The first one is the best previous position of the the kth particle at the ith iteration P_k^i. Another is tracked by the particle swarm optimizer and obtained so far by any particle in the population. This best position is a global best amongst all the particles G^i from the first iteration to the ith iteration. Each particle is equivalent to a candidate solution of a problem. The particle moves according to the velocity adjusted, which is based on that particle's experience and the experience of its companion. As a result, PSO usually can achieve the nearly best solution in much lesser evolution than the others [9, 10]. The original particle swarm optimization (PSO) can be described as follows.

In step 1 , how many particles are used to solve the problem is decided. Every particle has its own position, velocity and best solution. Then,

$$f(P_k^i) \leq f(P_k^{i-1}) \leq \cdots \leq f(P_k^1).$$

In step 2 , the process of velocity update is shown as follows:

$$V_k^{i+1} = V_k^i + C_1 \cdot r_1 . (P_k^i - X_k^i) + C_2 \cdot r_2 \cdot (G^i - X_k^i).$$

In step 3 , movement of the particles is processed by the following equation:

$$X_k^{i+1} = X_k^i + V_k^{i+1}, i = 0, 1, \cdots, M-1,$$

where M is the particle size, $-V_{max} \leq V_k^{i+1} \leq V_{max}$ (V_{max} is the maximum velocity), and r_1 and r_2 are random variables such that $0 \leq r_1, r_2 \leq 1$. If a solution is better than G^i, G^i will be replaced by this solution to represent G^{i+1} in step 4. Otherwise, there will be no change for the global best solution, i.e. $G^i = G^{i+1}$. These recursive steps will go on unless we reach the termination condition in step 5. In this paper, we propose a method based on self-mutation concept for PSO to solve the discrete problem of timetable scheduling which will be presented in next section.

4. PSO for timetable scheduling

The proposed method attempts every particle would be self-changed two slots of particle and gives the chance to have the global best and the local best solutions. It increases the chance of particles to find a better solution and to leap the local optimal solutions. The self-mutated PSO method can be shown in Figure 1 and described as follows:

First of all, perform with PSO operation similar, the proposed self-mutated PSO method also random generation of an initial population that we want to use to solve the problem. For example, there are M particles to be used; the particles can be represented as:

$$X = \{x_1, x_2, \cdots, x_M\}$$

In beginning, we randomly produce a group of 20 candidate solutions as particles. Each particle is equivalent to a candidate solution of a problem. The performance of 20 candidate solutions is first evaluated and ordered. The best previous position of the kth particle should be put in P_k^i at the ith iteration. The best position amongst all the particles from the first iteration to the ith iteration should be put in G^i. Then new timetables of next generation are produced by following several steps:

- Movement of the particles is processed by the following procedure:
  1. Each particle (X_k^i) must be changed two slots at random by itself.

$$S_k^{i+1} = rand - mutate(X_k^i)$$

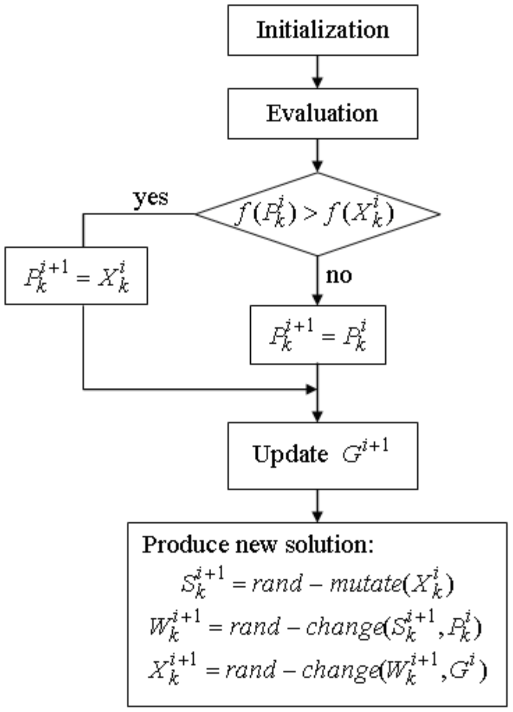

Figure 1. Outline of PSO for Timetable Scheduling.

2. Copy a slot for a subject randomly from the local best (P_k^i) to particle (S_k^i).

$$W_k^{i+1} = rand - change(S_k^{i+1}, P_k^i)$$

3. Copy a slot for a subject randomly from the global best (G^i) to W_k^{i+1} at random.

$$X_k^{i+1} = rand - change(W_k^{i+1}, G^i)$$

The evolutionary cycle will repeat over and over again until an optimal timetable is found or a certain maximum number of iteration is reached. Therefore, it is easier to escape from local optimum and approach the global or near global optimum in short time.

5. Experimental result

The overall quality of the timetable is evaluated by the evaluation function that adds up all violations of all constraints by testing it with each student's exams. Each constraint has an associated weight or penalty defined in an intuitive way as follows:

1. A single clash has a penalty of 10.
2. An instance of four and three exams in on day has a penalty 4 and 3 respectively.
3. An instance of two exams consecutive in the same half day or in different half day has a penalty 2 and 1 respectively. The results averaged from 10 seeds are shown in Table 2 and Figure 2, 3, 4.

Table 2. Performance evaluation using PSO

Performance evaluation using PSO						
	11 exams		**13 exams**		**15 exams**	
seed	Penalty	Time (sec.)	Penalty	Time (sec.)	Penalty	Time (sec.)
1	219	3.97	409	4.06	463	4.27
2	216	3.88	452	4.08	463	4.22
3	218	3.86	390	4.08	464	4.23
4	217	3.86	420	4.06	467	4.22
5	217	3.86	444	4.16	464	4.25
6	219	3.88	433	4.06	470	4.20
7	217	3.84	407	4.05	464	4.22
8	218	3.86	408	4.08	458	4.23
9	219	3.86	389	4.05	474	4.20
10	216	3.88	408	4.05	476	4.22
Average	217.60	3.88	416	4.07	466.30	4.23

Experimental results demonstrate that PSO can be an efficient method for solving the discrete problem of the timetable scheduling and there is no clash happened for the timetable scheduling. We will also apply our former work, parallel particle swarm optimization (PPSO) [11] or combining the PPSO with the tabu search approach to further improving the efficiency of the timetable scheduling problem.

References

[1] H. L. Fang, "Investigating genetic algorithms in scheduling," in *Technical Paper 12, Department of Artificial Intelligence, University of Edinburgh*, 1992.

[2] S. C. Chu and H. L. Fang, "Genetic algorithms vs. tabu search in timetable scheduling," in *Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, 1999, pp. 492–495.

[3] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Sixth International Sym-*

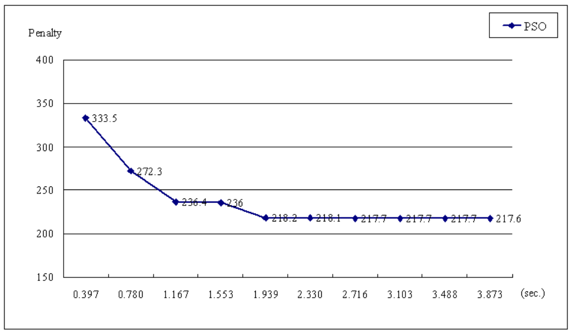

Figure 2. A student takes 11 exams at most.

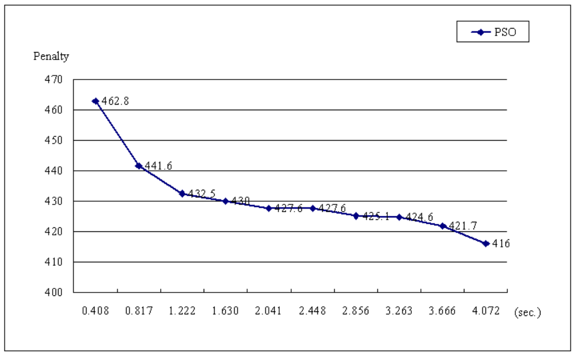

Figure 3. A student takes 13 exams at most.

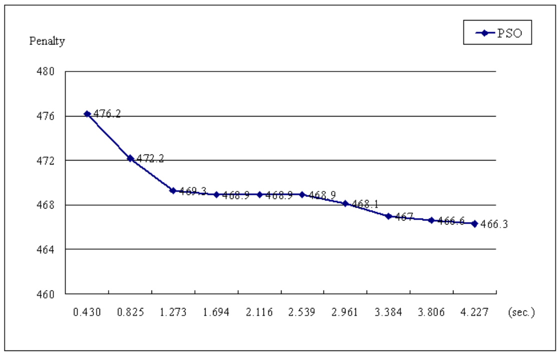

Figure 4. A student takes 15 exams at most.

posium on Micro Machine and Human Science, 1995, pp. 39–43.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[5] Peter Tarasewich and Patrick R. McMullen, "Swarm intelligence," *Communications of the ACM*, vol. 45, no. 8, pp. 63–67, 2002.

[6] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, 1989.

[7] L. Davis, *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, 1991.

[8] M. Gen and R. Cheng, *Genetic algorithm and engineering design*, John Wiley and Sons, New York, 1997.

[9] P. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Proc. Seventh Annual Conference on Evolutionary Programming*, 1998, pp. 601–611.

[10] R. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proc. Seventh Annual Conference on Evolutionary Programming*, 1998, pp. 611–619.

[11] J.-F. Chang, S. C. Chu, John F. Roddick, and J. S. Pan, "A parallel particle swarm optimization algorithm with communication strategies," *Journal of Information Science and Engineering*, vol. 21, no. 4, pp. 809–818, 2005.