Evolutionary Petri Net Approach to Periodic Job-Shop-Scheduling

Hiroki TOHME, Morikazu NAKAMURA, Koji HACHIMAN, Kenji ONAGA
Department of Information Engineering, University of the Ryukyus
1 Senbaru Nishihara, Okinawa, 903-0213 Japan
E-mail: {hiro, morikazu}@ads.ie.u-ryukyu.ac.jp

ABSTRACT

In this paper, Periodic Job-Shop-Scheduling Problem (PJSSP) extended from Job-Shop-Scheduling Problem (JSSP) is considered. Our approach, called as "Evolutionary Petri Net", solves the problem approximately by evolving the net structure of Petri nets. An effective crossover is proposed such that is uses bottle-neck information of Petri net model. The proposed crossover can accelerate the evolution.

1. INTRODUCTION

Job-Shop-Scheduling Problems (JSSP) are well-known combinatorial optimization problems characterized as NP-hard [1].

Here we consider Periodic Job-Shop-Scheduling Problems (PJSSP), an iterative version of the JSSP. From a practical point of view, the PJSSP is important though its research results are less enough than the ordinal (non-periodic) one [2]. In the PJSSP, each job is processed iteratively n times (n is arbitrary). For the traditional JSSP, the schedule length is minimized but, for the PJSSP, the cycle time (the average schedule length) is the criteria of the optimization, that is, $min(\lim_{n\to\infty} \frac{The\ total\ length}{n})$. Since a machine can start the k-th job processing before the $(k-1)$th job processing at all the machines has been completed (the machine does not need to wait until the $(k-1)$th job processing has been completed), "the cycle time" is not longer than "the schedule length" for the same Job-Shop instance.

Timed Petri Nets (TPN) are very useful for the optimal design and control of manufacturing systems. It is known that Timed Marked Graphs (TMG) can model scheduled (i.e., deterministic) Periodic Job-Shop systems [4].

In this paper, we propose an evolutionary computation approach to the PJSSP in which a Petri net model of the Periodic Job-Shop system is generated automatically. Most of the conventional Petri net researches on the manufacturing systems do not treat the automatic net generation. Moreover, a Petri net is constructed *evolutionarily* in this paper, therefore, we call our approach "Evolutionary Petri nets". As far as the authors know, there are no research results related with Evolutionary Petri nets.

The remainder of this paper is as follows. Section 2 introduces notations and basic properties of Petri nets. In Section 3, we propose evolutionary Petri net approach to the PJSSP. Experimental evaluation for the proposed approach is performed in Section 4. Finally, we conclude in Section 5.

2. PRELIMINARIES

We assume the reader is familiarized with the structure, firing rules, and basic properties of Petri nets (See for details [3]).

Let $N = (P, T, PRE, POST)$ be a place / transition net with the place set P and the transition set T. PRE and $POST$ constructing $C(= POST - PRE)$ are $|P| \times |T|$ matrix corresponding to the pre and post incidence functions, respectively.

If an integer vector Y ($|Y| = |P|$) satisfies $Y^t \cdot C = 0$, we call it P-semiflow of N. For an initial marking M_0 and any reachable marking M from M_0, the following invariant holds: $Y^t \cdot M = Y^t \cdot M_0$. The support of P-semiflow is defined by $||Y|| = \{p \in P |\ Y(p) \geq 0\}$. For all the places, if the number of input and output transitions is equal to 1, the Petri net is called a marked graph (MG). It is well-known that i) a MG is live iff all circuits include at least one token and ii) a MG is conservative iff it is strongly connected.

0-7803-5731-0/99/$10.00 ©1999 IEEE

Let $N' = (P, T, PRE, POST, \theta)$ be a TPN in which P, T, PRE and $POST$ are the same as those in N. $\theta: T \rightarrow \mathcal{N}$ is a function representing transition firing time, where N is the set of natural number. In this paper, we assume the firing time is *deterministic* but not *stochastic*.

The objective of the JSSP is to obtain the shortest schedule, while the PJSSP optimizes the cycle time γ (the inverse of γ, $1/\gamma$, represents the throughput, that is, the number of firings per unit time). It is well-known that the cycle time is obtained as follows [5]:

$$\gamma = \max_{C_i} \frac{\sum_{t_j \in C_i} \theta_j}{\sum_{p_k \in C_i} M(p_k)} \quad (1)$$

where the numerator means the sum of firing times of the transitions included in C_i and the denominator does the number of tokens in C_i.

The direct computation of (1) is so difficult since there can be exist exponential number of (elementary) circuits in a graph (Petri net). However, some efficient methods are also known in the literature [5].

For TMGs, the minimum cycle time (to be achieved by the earliest firing policy) can be computed exactly in polynomial time by solving the following LPP [5] [6]:

$$\mathcal{LPP}1: \quad \Gamma = \max Y^t \cdot PRE \cdot \theta \quad (2)$$

$$s.t. \quad Y^t \cdot C = 0 \quad (3)$$

$$Y^t \cdot M_0 = 1 \quad (4)$$

$$Y \geq 0 \quad (5)$$

The inputs of LPP1 are C, θ and M_0, while the outputs are Γ and Y. The initial marking is an input data which corresponds to the resource distribution of the manufacturing system and represents the number of machines, AGVs, and so on. It is obvious from (3) that Y should be a P-semiflow. Note that each elementary circuit in a MG corresponds to a P-semiflow. By using the simplex method, we can solve LPP1 efficiently (in linear time practically).

3. EVOLUTIONARY PETRI NETS

In this section, the evolutionary Petri net approach to the PJSSP is proposed, that is, the proposed method constructs a Petri net minimizing the cycle time for the periodic Job-Shop system by the evolutionary computation.

3.1. OUTLINE

The net of the periodic Job-Shop system is composed of the Task-Sequence-Net and the Machine-Schedule-Net as follows:

Task-Sequence-Net(TSN) is the net representing the task sequence for each job. Each transition corresponds to a task composing a job. Each place in a TSN shows the condition if the material arrives at the machine on which the task corresponding to its output transition is to be processed. Note that tokens in a TSN correspond to AGVs. For periodicity, it composes a cycle by joining the last task to the first task.

Machine-Schedule-Net(MSN) is the net representing the schedule for each machine. This net connects the tasks to be processed at the same machine by a circuit, therefore, there are $|M|$ circuits in this net, where $|M|$ is the number of machines. Note that tokens in an MSN correspond to available machines. For example, if the circuit for machine M_i includes two tokens, there are two available machines of the machine type M_i.

Periodic-Job-Shop-Net(PJSN) is the net representing the whole net by combining the TSNs and the MSNs.

Let us show a "4 × 3"problem instance (i.e., $\#jobs$= 4, $\#machine$= 3). Table 1.(a) is the task sequence for each job (Input data). That is, job J_1 is composed of three tasks in which the first task is processed at M_3, the second task at M_1, and the third one at M_2, and so on. Table 2.(b) is a machine schedule (Output) in which M_1 processes iteratively the tasks based on the order "$J_1 \rightarrow J_4 \rightarrow J_1 \rightarrow J_2$". Figure 1 shows the PJSN (TSN and MSN) corresponding to Table 1.(a) and (b).

Table 1 Example of PJSSP

(a) Task Sequence

JOB	SEQUENCE		
J_1	M_3	M_1	M_2
J_2	M_2	M_3	M_1
J_3	M_1	M_2	M_3
J_4	M_2	M_1	M_3

(b) Machine Schedule

MACH	SCHEDULE			
M_1	J_3	J_4	J_1	J_2
M_2	J_2	J_3	J_4	J_1
M_3	J_1	J_4	J_2	J_3

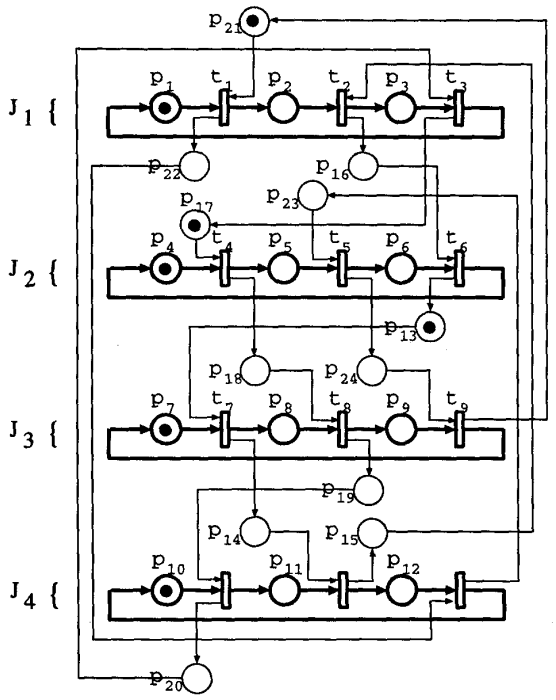

Figure 1 Example of PJSN

In Figure 1, the subnets drawn by bold lines represent TSNs, while the others MSNs. A transition corresponds to a task (a task is a processing element in a job). For example, t_1 corresponds to the task in J_1 to be processed at M_3 from Table 1.(a). The input places to each transition represent the conditions for processing such that the places in the TSN are marked when the material arrives and those in the MSN are marked when the machine is available.

According to the condition stated in Section 2, the initial marking is constructed by putting one token at the starting place of each circuit in the TSN and the MSN. In Figure 1, $p_1, p_4, p_7, p_{10}, p_{13}, p_{17}$, and p_{21} is marked with one token.

The Petri net systems constructed as the above are live and conservative, because the net is a MG, each elementary circuit has at least one token, and strongly connected.

In our approach, a TSN is fixed since it is determined uniquely from given jobs. On the other hand, an MSN is constructed evolutionarily, that is, it is represented by chromosome and applied with genetic operations [8]. At the fitness evaluation stage, each chromosome is converted into an MSN and then a PJSN is constructed. M_0 is also determined by taking the resource condition into account. By solving LPP1 with the simplex method, the cycle time of the PJSN with M_0 is computed. The outline of our approach is described as follows:

```
EvolutionaryPN(){
  input task sequences;
  construct TSN;
  generate initial population (machine schedules);
  convert each chromosome into a MSN and
                              construct the PJSN;
  determine M0 based on the resource condition;
  compute throughput of the PJSN
                              with M0 by LPP1;
  while(the termination condition does not hold){
    select by the elite and the roulette wheel;
    crossover according to the crossover rate;
    mutation according to the mutation rate;
    convert each chromosome into a MSN and
                              construct the PJSN;
    compute throughput of the PJSN
                              with M0 by LPP1;
  }
}
```

3.2. CHROMOSOME REPRESENTATION

Since TSNs can be determined uniquely from task sequences, only MSNs are to be evolved. The chromosome is simply represented as a sequence of jobs. That is, a gene corresponds to a job. For example, the Petri net system shown in Figure 1 is represented by the chromosome "3 – 4 – 1 – 2 – 2 – 3 – 4 – 1 – 1 – 4 – 2 – 3" in which the first four genes "3-4-1-2" correspond to the machine sequence of M_1, i.e., $J_3 - J_4 - J_1 - J_2$, the second four genes "2 – 3 – 4 – 1" do to that of M_2 "$J_2 - J_3 - J_4 - J_1$", and the remains correspond to that of M_3 "$J_1 - J_4 - J_2 - J_3$".

3.3. GENETIC OPERATIONS

In order to improve the quality of solutions, the information on the bottle-neck circuit is used in the crossover operation. The bottle-neck circuit is obtained when the cycle time is computed by LPP1. Here we explain the proposed crossover operation. The proposed idea is described as follows:

Let C_1 and C_2 be chromosomes and Y_1 and Y_2 be the P-semiflows corresponding to the bottle-neck circuits in the nets constructed from C_1 and C_2, respectively. Let M_i be such that $| MSN_i \cap ||Y_1|| | = \max_k | MSN_k \cap ||Y_1|| |$, where MSN_i represents the circuit corresponding to the schedule of M_i. Similarly let M_j be such that $| MSN_j \cap ||Y_2|| | =$

$\max_k \mid MSN_k \cap ||Y_2|| \mid$. The crossover is performed by exchanging the substrings C_1' and C_2' which correspond to the machine schedules of M_i and M_j, respectively.

Let us consider an example shown in Figure 2 in which for easier explanation the net representation is used in stead of chromosome one. On PJSNs in Figure 2.(a) and (b), the crossover is performed: The shaded places in both nets are included in the bottle-neck circuits, i.e., P-semiflow Y_1 and Y_2, respectively. The machine schedule drawn by dotted line includes the maximum number of places in Y_1 (Y_2). By exchanging these machine schedules, we have two children shown in Figure 3.(c) and (d). The shaded places in (c) and (d) also compose the bottle-neck circuit. The idea of the crossover is to destroy the bottle-neck circuit while the rest is remained.

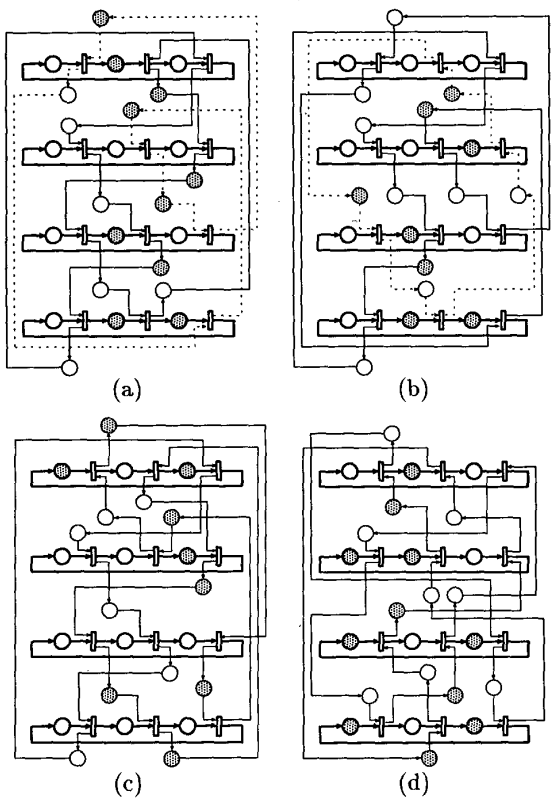

Figure 2 Crossover based on the bottle-neck circuits

A simple mutation is chosen in our approach in which it randomly rearranges jobs among the same machine.

To leave best chromosomes (i.e., shortest machine schedule), we adopt the roulette wheel with the elite selection strategy. So we can keep some excellent chromosomes during the evolving process.

4. EXPERIMENTAL EVALUATION

To evaluate the proposed method, we solve problem instances such as 10×5 (la01.dat and la03.dat), 15×5 (la06.dat), 20 × 5 (la12.dat), 10 × 10 (la18.dat), and so on. They are taken from OR-Library [10].

In the experiment, the following common genetic parameters in this experiment are used:

The generation: 300
The crossover rate: 0.75
The mutation rate: 0.08

All the results in this section show the average value of optimal solution curves among 30 executions for each instance with respect to generations.

We assume in the first (4.1) and second (4.2) experiments that the number of tokens in solutions each circuit of the MSN is equal to 1 (the number of available machines of each type is 1) and the number of tokens in each circuit of the TSN is equal to 1 (the number of AGVs for each job is 1). These restrictions lead to the simplest system structure. If the system has an initial marking M_0' such that $M_0' < M_0$, the system should be dead.

4.1. COMPARISON OF CROSSOVERS

In this experiment, the following genetic parameters are used:

The number of chromosomes: 50

This experiment compares the proposed crossover with the uniform order crossover [9]. The proposed crossover employs the information obtained from the bottle-neck circuit but not in the standard uniform order crossover. Figure 3 depicts a result in which we can observe that the net is evolved as the generation proceeds and the usage of the bottle-neck information contributes to improve the solution quality. The same results are got from other instances.

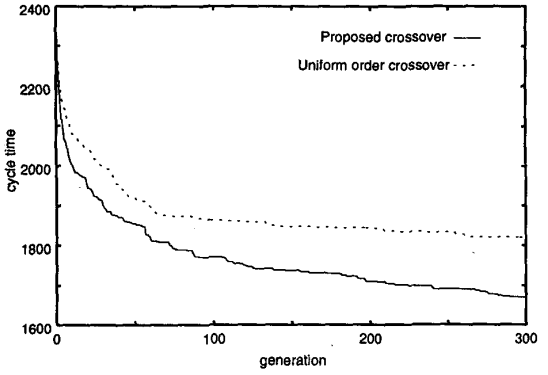

Figure 3 Solution-curves varying crossovers (la18.dat)

4.2. EFFECTS OF THE NUMBER OF CHROMOSOMES

Here we experiment to observe effects of the number of chromosomes. Our proposed method runs with the following parameters:

The number of chromosomes: 10, 30 ,50, 70

Figure 4 shows the experimental result. Even though it is very natural, the result says that larger population size leads to better solution quality.

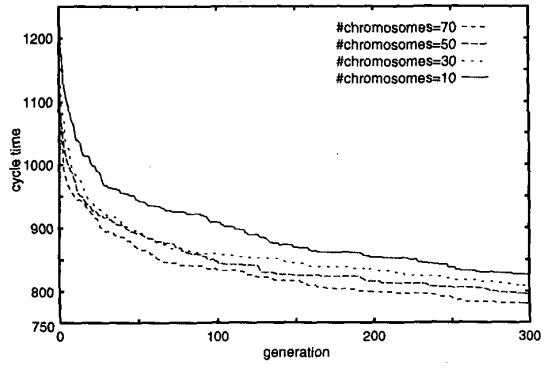

Figure 4 Solution-curves varying pop-size (la03.dat)

4.3. EFFECTS OF THE NUMBER OF AGVs

In Figure 5 effects of the number of AGVs are shown in which the number of chromosomes is fixed as 50 and the number of AGVs is varied from 1 to 4. We can observe that the solution quality is improved more when the number of AGVs is larger. However, the improvement saturates when we increase AGVs since the quality depends also on other resources.

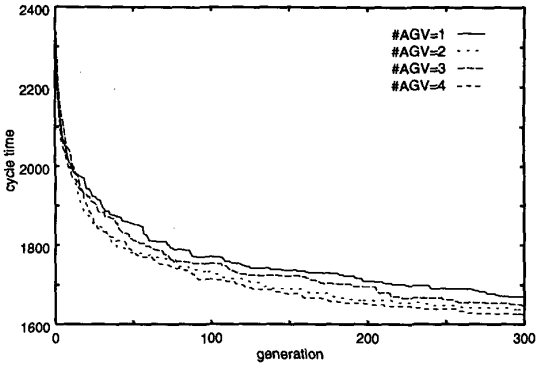

Figure 5 Solution-curves varying #AGVs (la18.dat)

Figure 6 shows the experimental result for la03.dat in which there is no difference on the solution quality when the number of AGVs is 2, 3, and 4. It means that two of four AGVs are not useful to improve the solution quality.

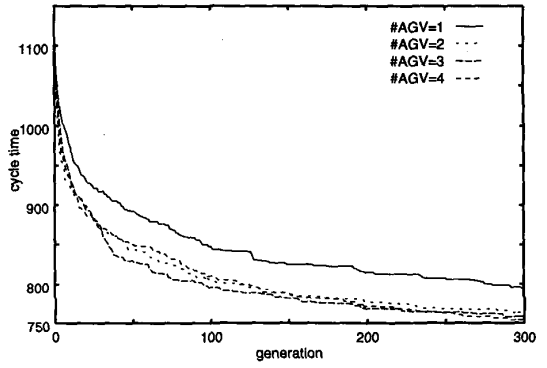

Figure 6 Solution-curves varying #AGVs (la03.dat)

4.4. MULTI-OBJECTIVE OPTIMIZATION

When it comes to real manufacturing systems, the number of AGVs should be small since increasing AGVs makes not only the system cost increased but also the system control so difficult. For example, the collision problem among AGVs become so complicated.

From the above reason, we extend our approach to multiple objective one, that is, the new method minimizes first the cycle time and then the number of AGVs. Figure 7 depicts on execution curve of the new method in which it obtains finely the cycle time = 655 and the number of AGVs = 2 by one GA execution.

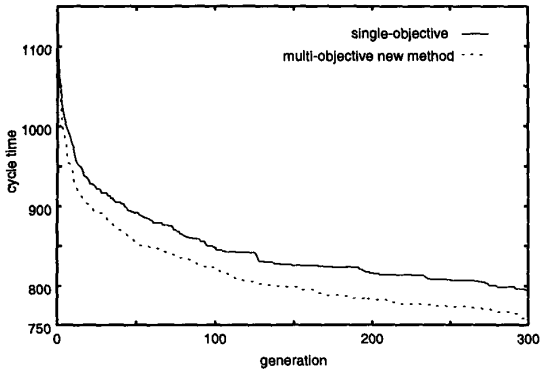

Figure 7 Solution-curves varying efficiently #AGVs (la03.dat)

5. CONCLUSION

In this paper, we proposed an evolutionary computation approach to the PJSSP in which a Petri net model of periodic Job-Shop system is generated automatically and evolutionarily. The crossover operation proposed in this paper uses information of the bottle-neck circuit. Experimental evaluation shows effectiveness of our approach and especially prove usefulness of the proposed crossover. Moreover we extended the proposed one to a multi-objective optimization version. The extension optimizes first the cycle time and then the number of AGVs in one GA execution. We are currently trying to apply our method to more complex resource cases.

References

[1] J. Magott, "New NP-Complete problems in performance evaluation and concurrent systems using Petri nets," IEEE Trans. on Software Engineering, Vol.SE-13, No.5, pp.578-581, 1987.

[2] TE. Lee and JS. Song,"Petri Net Modeling and Scheduling for Periodic Job Shops with Blocking," Proc. 1st International Workshop on Manufacturing and Petri nets, Osaka, 1996.

[3] T. Murata, "Petri Nets: Properties, Analysis and Applications," Proceedings of the IEEE, Vol.77, No.4, pp.541-580, April 1989.

[4] F. Dicesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat, "Practice of Petri Nets in Manufacturing," Chapman and Hall, 1993.

[5] J. Campos, J.M. Colom, and M. Silva :"Performance Evaluation of Repetitive Automated Manufacturing Systems," Proc. Rensselaer Polytechnic Institute 2nd Int. Conf. Computer Integrated Manufacturing, Troy, NY, pp.74-81, 1990.

[6] J. Campos, G. Chiola, J. M. Colom, and M. Silva, "Properties and Performance Bounds for Timed Marked Graphs," IEEE Transactions on Circuits and systems I, Vol.39, No.5, May 1992.

[7] S. Laftit, J.M. Proth, and S.L. Xie :"Optimization of Invariant Criteria for Event Graphs," IEEE Transactions on Automatic control, Vol.37, No.5, pp.547-555, May 1992.

[8] J. H. Holland :"Adaptation in Natural And Artificial Systems," A Bradford Book, MIT Press edition, 1996.

[9] L. Davis, "Handbook of Genetic Algorithms," Van Nostrand Reinhold, 1991.

[10] http://mscmga.ms.ic.ac.uk/info.html