行政院國家科學委員會補助專題研究計畫成果報告

結合自指引基因演算法與排程優勢性質求解單機與平行機排程問題

計畫類別：☑個別型計畫　　□整合型計畫

計畫編號：NSC 98－2218－E－343－001－MY2

執行期間：98 年 01 月 01 日 至 99 年 07 月 31 日

計畫主持人：陳世興

共同主持人：

計畫參與人員：陳萌智、陳木中、白健志、宋央琳

執行單位：南華大學電子商務學系

中　　華　　民　　國　　99　年　10　月　17　日

行政院國家科學委員會專題研究計畫成果報告

•結合自指引基因演算法與排程優勢性質求解單機與平行機排程問題

Self-Guided GA with Dominance Properties for Single Machine and Parallel Machine Scheduling Problems

計 畫 編 號：NSC 98-2218-E-343-001-MY2
執 行 期 限：98 年 01 月 01 日至 99 年 07 月 31 日
主 持 人：陳世興　南華大學電子商務學系
共 同 主 持 人：
計畫參與人員：陳萌智　吳鳳科技大學資訊管理學系
陳木中　吳鳳科技大學資訊工程學系
白健志　南華大學資管所
宋央琳　南華大學資管所

一、中文摘要

本計劃研究排程優勢性質(Dominance Properties)與具機率模式的自指引基因演算法(Self-Guided Genetic Algorithm)之結合，用來有效解決單機與平行機排程問題。在本計劃的前半段研究時間裡，本研究先完成推導單機與平行機排程問題的優勢性質之後，發現排程優勢性質的運作效率高且能得到較好的解，且可以進一步與本主持人所提出的自指引遺傳演算法 (Self-Guided Genetic Algorithm)結合，改善排程的結果。

第二年加入整備時間(Setup time)於單機與平行機排程問題之中，因為整備時間使得排程問題更為複雜，因此過去的方法較難得到好的結果，因此本研究也推導在考慮整備時間的單機與平行機排程優勢性質，協助演算法得到較好的結果，之後也與自指引基因演算法結合，也得到比過去有顯著性改善的結果。

有鑑於研究結果能有效的解決這些複雜的排程問題，目前已有兩篇論文已被 SCI 期刊接受，另外還有審查中與國際研討會的論文發表各一篇。

最後，由本研究的結果可以了解到排程的優勢性質的優點，因此在研究排程問題的學者，可以朝此方向導出其他問題的優勢性質，除了可與不同的演算法來做結合，也能透過自指引基因演算法帶來更好的求解品質。

關鍵詞：具機率模式之演化式演算法，排程，

2 緣由與目的：

More and more sophisticated Estimation of Distribution Algorithms (EDAs) have been proposed and developed to solve combinatorial problems in recent years. Some of them were quite successful; however, it is not always clear why and how an EDAs works. Recently, a number of evolutionary algorithms that guide the exploration of the search space by building probabilistic models of promising solutions found so far have been proposed. For sample drawn from an unknown probability distribution, the selected set of promising solutions allow the optimization algorithm to generate new solutions that are somehow similar to the ones contained in the original selected set of solutions. There are algorithms that are able to estimate that probability distribution by using the selected set of solutions itself and use this estimate to generate new solutions. These algorithms are called Evolutionary algorithms based on probabilistic models (EAPMs) [1, 2, 3, 4, 5, 6, 7].

EAPMs rely on probabilistic model which is built by parental distribution and sample from the model. There are some algorithms of EAPMs that employ probabilistic model to building genetic algorithms, such as EA/G [8], ant-miner [1], ACGA [9] and Self Guided GA [10]. In these solutions are selected from an initially randomly generated population of solutions like in the simple GA. Then, the true probability distribution of the selected set of solutions is estimated and new solutions are generated according to this estimate. The new solutions are then added into the original population, replacing some of the old ones. The process is repeated until the termination criteria are met.

In [11], they stated when an appropriate heuristic approach is incorporated into an EAPM, the performance of the EAPM is improved. [12] discussed some properties of the optimal solution, and these properties are used to develop both optimal and heuristic algorithms. In [13], they analyzed the performance of various heuristic procedures, including dispatch rules, a greedy procedure and a decision theory search heuristic. Among the heuristics, [14] developed several dispatch rules and a filtered beam search procedure. [13] presented an additional dispatch rule and a greedy procedure, and also considered the use of dominance rules to further improve the schedule obtained by the heuristics. In our previous study [15], dominance properties of (the conditions on) the optimal schedule are developed based on the switching of two adjacent jobs i and j. These dominance properties are only necessary conditions and not sufficient conditions for any given schedule to be optimal. The DPs are good at generating efficient solutions which might be helpful for genetic algorithm to obtain better parental distribution in the beginning.

Consequently, the two-year project considered a framework which integrates the dominance properties and EAPM to enhance the solution quality when we solve scheduling problems. This research studied two kinds of scheduling problems in the first year; one is the single machine scheduling problems, and the other is the parallel scheduling problems. When we studied the single machine scheduling strongly NP-hard problems to minimize the total earliness and tardiness costs. Then, the parallel machine scheduling problems are to minimize the makespan of the scheduling jobs.

Based on the results of the first year project, the second year project studied the scheduling problem considering the setup cost. When the setup cost is considered, the problems are even harder than the ones in the first year. As a result, this research also derived the dominance properties for the single machine and parallel machine scheduling problems with setup cost.

When we conducted the experiments, we compare with the influence of dominance properties to further improve the solution quality of EA/G, ACGA and Self-guided GA and take a close look at the evolutionary diversity for a single machine scheduling problem. We find out dominance properties are good for solution quality of Self-guided GA in the proposed framework. In addition, because DPs are excellent to yield good solution quality no matter the single machine scheduling problems as well as the parallel machine scheduling problem, the research results have been accepted by some SCI journals and an international conference. There is a SCI paper which is still under review. The detail list is shown below:

1. Chang, P. C., and S. H. Chen (2010), "Integrating Dominance Properties with Genetic Algorithms for Parallel Machine Scheduling Problems with Setup Times," accepted by Applied Soft Computing Journal.

2. Chang, P. C., S. H. Chen, T. Lie, and Y. C. Liu (2011), "A Genetic Algorithm Enhanced

by Dominance Properties for Single Machine Scheduling Problems with Setup Costs," accepted by International Journal of Innovative Computing, Information and Control, 7(4).

3. Shih-Shin Chen, Pei-Chann Chang, Min Chih Chen, and Yuh Min Chen (2009), A Self-guided Genetic Algorithm with Dominance Properties for Single Machine Scheduling problems, 2009 IEEE Symposium on Computational Intelligence in Scheduling (CISched 2009), Nashville, TN, U.S.A.

4. Shih-Shin Chen, Pei-Chann Chang, Min Chih Chen, Estimation of Distribution Algorithms with DominanceProperties for Machine Scheduling problems, submitted to Applied Mathematical Modelling.

Tthrough these fruitful research results, researchers could understand the DPs are beneficial to solve the scheduling problems. It is a good direction for the academic researchers in doing scheduling problems. Moreover, DPs could be further integrated with other meta-heuristics to further improve the solution quality while the Self-Guided GA might be a potential algorithm could be applied. Finally, because there are many research result carried out by this research while the number of pages is limitted, the final report illutrates the DPs with Self-Guided GA which solves single machine scheduling problems first. And then to show the derivation of DPs for single machine and unrelated parallel scheduling problem which consider the setup cost.

3 研究報告應含的內容

3.1 DPs with Self-Guided GA for the Single Machine Scheduling problems

In this paper, a deterministic single machine scheduling problem without release date is investigated and the objective is to minimize the total sum of earliness and tardiness penalties. A detailed formulation of the problem is described as follows: A set of n independent jobs $\{J_1, J_2, ..., J_n\}$ has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and unforced idle time is not allowed. Job J_j, $(j = 1, 2, ..., n)$ becomes available for processing at the beginning, requires a processing time p_j and should be completed on its due date d_j. For any given schedule, the objective is to find a schedule that minimizes the sum of of the earliness and tardiness penalties of all jobs $\sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j)$ where α_j and β_j are the earliness and tardiness penalties of job J_j. The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods when they are needed.

We derive the dominance properties for two adjacent jobs (i and j), which has distinct due dates (d_i and d_j), earliness penalties (α_i and α_j), and tardiness penalties (β_i and β_j). The processing time of these jobs are p_i and p_j. The dominance properties give the precedence relationship between any two adjacent jobs in a schedule. In the optimal schedule, all the adjacent jobs will satisfy the dominance properties.

There is a schedule Π, in which two adjacent jobs i and j are in positions k and $k+1$ respectively. We consider the objective function $Z(\Pi)$ for this schedule Π. we rewrite the objective function $Z(\Pi)$, in such a way that only terms corresponding to jobs (i and j) in positions k and $k+1$ are present explicitly in the objective function $Z(\Pi)$. The other terms are absorbed in constants defined below.

$$Z(\Pi) = G_1 + G_2 + \gamma_i|d_i - f_i| + \gamma_j|d_j - f_j| \quad (1)$$

where

$$G_1 = \sum_{l=1}^{k-1} \gamma_l |d_l - f_l|$$

$$G_2 = \sum_{l=k+2}^{n} \gamma_l |d_l - f_l|$$

In the above expressions, the value of γ_p is defined as follows:

- $\gamma_p = \alpha_p$, if $d_p > f_p$; this means that job p is an early job.
- $\gamma_p = \beta_p$, if $d_p < f_p$; this means that job p is a tardy job.
- $\gamma_p = 0$, if $d_p = f_p$; this means that job p is an on time job.

Consider the schedule Π_x given as

$$\Pi_x = \{* \ * \ *...i\ j\ *...*\ *\} \quad (2)$$

The schedule Π_x, the jobs i and j are in positions k and $k+1$ respectively. In Π_x '*' denotes some other jobs (other than i and j) are in that positions 1 to n (other than positions k and $k+1$). In the schedule Π_x, the finish time (f_i) of job i is $(A+p_i)$ and finish time (f_j) of job j is $(A+p_i+p_j)$. The value of A is the finish time of the job in position $(k-1)$ and is

$$A = \sum_{l=1}^{k-1} p_l \quad (3)$$

When the jobs i and j are interchanged the schedule Π_x, the resulting schedule is Π_y and is

$$\Pi_y = \{* \ * \ *...j\ i\ *...*\ *\}$$

Note that in schedule Π_y only the jobs i and j are interchanged and all other jobs are in the same positions as in schedule Π_x. In Π_y, the finish time (f_j) of job j is $(A+p_j)$ and finish time (f_i) of job i is $(A+p_j+p_i)$. We will compare the schedules Π_x and Π_y and find the conditions under which Π_x is better than Π_y. These conditions are the dominance properties.

In this schedule Π_x, the jobs i and j are either early, tardy or on time, respectively. Thus there are nine status of the job status combination when we consider the jobs i and j.

Let P be the sum of processing time of all the jobs. ($P = \sum_{j=1}^{n} p_j$). We assume that $d_j < P$, for all jobs ($j = 1, 2, ..., n$). We discuss the case when this assumption is not true later. With this assumption, we consider the above mentioned nine status and to illustrate first status in detail and derive the dominance properties.

Status.1: Consider two adjacent early jobs i (in position k) and j (in position $k+1$) in the schedule Π_x. This two adjacent jobs are early means that $d_i > (A+p_i)$ and $d_j > (A+p_i+p_j)$. This $d_j > (A+p_i+p_j)$ implies that $d_j > (A+p_j)$. Hence, there are two possibilities on d_i as given below.

- Possibility.(i). $d_i > (A+p_i+p_j)$
- Possibility.(ii). $d_i < (A+p_i+p_j)$.

Possibility.(i). Here in the schedule Π_x jobs i and j (in positions k and $k+1$) are early jobs. After interchange, in the schedule Π_y, the jobs j and i (in positions k and $k+1$) are also early jobs. This means that $d_i > (A+p_i)$, $d_j > (A+p_j)$, $d_i > (A+p_i+p_j)$ and $d_j > (A+p_i+p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for the schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \alpha_j(d_j - A - p_i - p_j) \quad (4)$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \alpha_i(d_i - A - p_j - p_i) \quad (5)$$

We now derive the condition under which $Z(\Pi_x) \leq Z(\Pi_y)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = -\alpha_i p_j + \alpha_j p_i \quad (6)$$

From the above expression, we see that $X \geq 0$ when the following condition is satisfied.

$$\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j} \quad (7)$$

We see that if $X > 0$, then the schedule Π_x is better than the schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}$. Based on this analysis, we state the following property.

Property.1: In the schedule Π_x, for two adjacent early jobs i (in position k) and j (in position $k+1$), and if $d_i > (A+p_i+p_j)$, then the schedule Π_x is better than the schedule Π_y, only when $\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}$.

The other possibility of Status 1 and other dominance properties can be proved in the same manner. For the further explanations, please refer to our work in [15].

3.1.1 DP-Self-Guided Genetic Algorithm

After we introduced the DPs, this section describes the detail procedures of the Self-guided GA. Because Self-guided GA originally uses random solutions as an initial population, DPs will replace it and generate good initial solutions for Self-guided GA. In other words, the whole procedures are identical to the original Self-guided GA except the initialization of the population.

EAPMs extract the gene variable structure from the population distribution and express it in a probabilistic model [3]. This research embeds the probabilistic model in the crossover and mutation operators to guide the evolution progress towards a more

promising solution space. The probabilistic model serves as a fitness surrogate and the probabilistic model evaluates the figure of merit of a new solution beforehand. This means that the probabilistic model will calculate the probability difference between two selected genes located in different positions to guide the movement of genes decided by the crossover and mutation operators. As a result, the proposed EAPM algorithm will guide the searching direction instead of blindly searching the solution space since it does not rely on proportional selection to generate solutions.

The benefits of the proposed method are preserving the salient genes of the chromosomes, and exploring and exploiting good searching directions for genetic operators. In addition, since the probabilistic difference provides good neighborhood information, it can serve as a fitness function surrogate. The detailed procedure of the DP-Self-guided GA is described as follows:

Population: A set of solutions
Generations: The maximum number of generations
$P(t)$: Probabilistic model
t : Generation index

1: Initialize *Population* by Dominance Properties
2: $t \leftarrow 0$
3: Initialize $P(t)$
4: **while** $t < generations$ **do**
5: EvaluateFitness (*Population*)
6: Selection/Elitism(*Population*)
7: $P(t+1) \leftarrow$ BuildingProbability-Model(*Selected Chromosomes*)
8: Self-Guided Crossover()
9: Self-Guided Mutation()
10: $t \leftarrow t+1$
11: **end while**

Figure 1: Algorithm1: MainProcedure of DP-Self-guided GA()

Step 1 is the initialization of a population by dominance properties. DPs generate good solutions and Self-guided GA does further evolution based on this good basis. Step 2 initializes the probability matrix $P(t)$ and the matrix size is $n - by - n$, where n is the problem size. Step 7 builds the probabilistic model $P(t)$ after the selection procedure.In Step 8 and Step 9, $P(t)$ is employed in the self-guided crossover operator and the self-guided mutation operator. The probabilistic model will guide the evolution direction, which is shown in Section 3.1.2 and Section 3.1.3. In this research, the two-point central crossover and swap mutation are applied in the crossover and mutation procedures for solving the scheduling problem under study.

We explain the proposed algorithm in detail in the following sections. We explain how the probabilistic model guides the crossover and mutation operators.

3.1.2 Mutation Operator with Probabilistic Model

Suppose two jobs i and j are randomly selected and they are located in position a and position b, respectively. p_{ia} and p_{jb} denote job i in position a and job j in position b. After these two jobs are swapped, the new probabilities of the two jobs become p_{ib} and p_{ja}. The probability difference Δ_{ij} is calculated as Eq. 8, which is a partial evaluation of the probability difference because the probability sum of the other jobs remains the same.

$$\begin{aligned}\Delta_{ij} &= P(X') - P(X) \\ &\approx \prod_{p \notin (aorb), g=[p]}^{n} P_{t+1}(X_{gp})[(p_{ib}p_{ja}) - (p_{ia}p_{jb})]. \end{aligned} \quad (8)$$

Now that the part of $\prod_{p \notin (aorb), g=[p]}^{n} P_{t+1}(X_{gp})$ is always ≥ 0, it can be subtracted and Eq. 8 is simplified as follows:

$$\Delta_{ij} = (p_{ib}p_{ja}) - (p_{ia}p_{jb}). \quad (9)$$

$$\Delta_{ij} = (p_{ib} + p_{ja}) - (p_{ia} + p_{jb}). \quad (10)$$

If Δ_{ij} is positive, it implies that one gene or both genes might move to a promising area. On the other hand, when Δ_{ij} is negative, the implication is that at least one gene moves to an inferior position.

On the basis of the probabilistic differences, it is natural to consider different choices of swapping points during the mutation procedure. A parameter TM is introduced for the self-guided mutation operator, which denotes the number of tournaments in comparing the probability differences among the TM choices in swap mutation. Basically, $TM \geq 2$ while $TM = 1$ implies that the mutation operator mutates the genes directly without comparing

the probability differences among the different TM choices.

When $TM = 2$, suppose the other alternative is that two jobs m and n are located in position c and position d, respectively. The probability difference of exchanging jobs m and n is:

$$\Delta_{mn} = (p_{md} + p_{nc}) - (p_{mc} + p_{nd}). \quad (11)$$

After Δ_{ij} and Δ_{mn} are obtained, the difference between the two alternatives is as follows:

$$\Delta = \Delta_{ij} - \Delta_{mn}. \quad (12)$$

If $\Delta < 0$, the contribution of swapping job m and n is better, so we swap job m and n. Otherwise, jobs i and j are swapped. Consequently, the option of a larger probability difference is selected and the corresponding two jobs are swapped. By observing the probability difference Δ, the self-guided mutation operator exploits the solution space to enhance the solution quality and prevent destroying some dominant genes in a chromosome. Moreover, the main procedure of the self-guided mutation is Eq. 12, where the time-complexity is only a constant after the probabilistic model is employed. This approach proves to work efficiently.

3.1.3 Crossover Operator with Probabilistic Model

The idea of Self-Guided Crossover is the same with Self-Guided Mutation, which employs the probability differences of the mating chromosomes by using the Eq. 13. By doing so, we could evaluate which chromosome is mated with a parent solution. For the detail description, please refer in [16].

$$\Delta = \Delta_1 - \Delta_2 = \prod_{p\in(CP1\ to\ CP2),g=[p]}^{n} P(Candidate1_{gp}) - \prod_{p\in(CP1\ to\ CP2),g=[p]}^{n} P(Candidate2_{gp}). \quad (13)$$

To conclude, the DP-Self-guided GA is obviously different from the previous EAPM algorithms. Firstly, the algorithm utilizes dominance properties to generate good initial solutions. Secondly, the proposed algorithm explicitly samples new solutions without using the crossover and mutation operators. The Self-guided GA embeds the probabilistic model in the crossover and mutation operators to explore and exploit the solution space. Most important of all, the algorithm works more efficiently than previous EAPM in solving the scheduling problem because the time-complexity is $O(n)$ whereas EAPM needs $O(n^2)$ time.

3.1.4 Experimental Results of DP-Self Guided GA

We conducted extensive computational experiments to evaluate the performance of the combination of dominance properties and Self-guided GA in solving the single-machine scheduling problem to minimize the total weighted earliness and tardiness costs [17]. We compared it with some algorithms in the literature. We implemented the algorithms in Java 2 (With JBuilder JIT compiler) on a Windows 2003 server (Intel Xeon 3.2 GHZ). In all the experiments, we replicated each instance 30 times.

The proposed algorithm was compared with SGA and some other algorithms from the literature, including GADP [15], ACGA [9], ACGADP [9], ACGA with evaporation method [18], and EA/G [8]. For more detailed results, please refer to our website[1].

Since there were significant differences in the results produced by the different algorithms under test (see Table 1), we conducted Duncan pairwise comparisons of the performance of the algorithms and show the results in Table 2.

Table 1: ANOVA results on the objective values of the single-machine scheduling problem produced by different algorithms

Source	DF	SS	Mean Square	F Value	P Value
instances	213	7.98E12	37457591062	1.94E7	<.0001
method	5	1214063.08	242812.62	125.66	<.0001
instances*method	1065	34492090.87	32386.94	16.76	<.0001
Error	37236	71952837.87	1932.35		
Corrected Total	38519	7.9785746E12			

When Duncan grouping is used, it shows that there

[1] http://peterchenweb.appspot.com/publications/sourceCodes/InjectionArtificialChromosomes/Results.htm

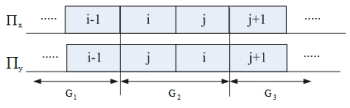
are significant differences between/among subjects if they are given different alphabets. Otherwise, there are no differences between/among the subjects. In Table 2, the results of Duncan grouping indicate that the DP-Self-guided GA outperforms any other algorithms. In addition, Self-guided GA performs as well as ACGA and EA/G. Finally, GADP is the worst in solving the single-machine scheduling problem in this Duncan comparison.

Table 2: Duncan grouping of the objective values of the single-machine scheduling problem produced by different algorithms

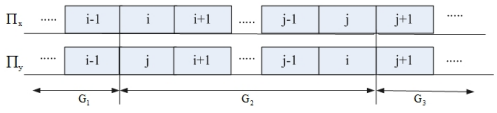
Duncan Grouping	Mean	N	Method
A	12827.07	6420	GADP
B	12816.47	6420	ACGADP
C	12813.66	6420	EA/G
C			
C	12813.28	6420	ACGA
C			
C	12813.25	6420	Self-Guided GA
D	12809.15	6420	DP-Self-Guided GA

3.2 DPs of Single Machine Scheduling problem with Setup Cost

We consider the problem of scheduling n jobs in a single machine and derive the dominance properties (necessary conditions) of the optimal schedule. In this section, we use the objective function ($Z(\prod)$) for total absolute deviation for the schedule $\prod$. To develop these dominance properties, we will consider interchanging two adjacent jobs and nonadjacent jobs in the schedule, and prove some intermediate results. The adjacent interchange and nonadjacent interchange of job i and job j are depicted at figure 2 respectively.



(a) Adjacent interchange



(b) Nonadjacent interchange

Figure 2: Two different types of interchanging methods

Thus, there are two schedules, i.e., $\prod_X$ for schedule X and $\prod_Y$ for the modified schedule Y. The corresponding objective functions of $\prod_X$ and $\prod_Y$, i.e., $Z(\prod_X)$ and $Z(\prod_Y)$, are listed as follows:

$$Z(\prod_x) = G_1 + G_2 + G_3 \tag{14}$$

$$Z(\prod_y) = G_1' + G_2' + G_3' \tag{15}$$

where

1. G_1: the objective of job(s) before job i
2. G_2: the objective between job i and job j
3. G_3: the objective of job(s) after job j
4. G_1': the objective job(s) before job j
5. G_2': the objective between job j and job i
6. G_3': the objective of job(s) after job i

We compare schedules $\prod_X$ and $\prod_Y$ by finding the conditions under which $\prod_X$ is better than $\prod_Y$. For a pair of jobs, i.e., job i and job j in a schedule, no matter for adjacent interchange or nonadjacent interchange, they are in one of the following status:

1. Job i is early and job j is early
2. Job i is early and job j is on-time
3. Job i is on-time and job j is tardy
4. t: Job i is tardy and job j is tardy

Because the objective values of a schedule with adjacent or nonadjacent interchange are different, there are totally 8 conditions corresponding to these two types of exchanges. Other than the cases discussed above, there is one extra case to be discussed in nonadjacent interchange which is the following:

1. Job i is early and job j is tardy

Because of the length of this report, please refer to the following paper for the detail procedures of the DPs for the single machine scheduling problem with setup considerations.

1. Chang, P. C., S. H. Chen, T. Lie, and Y. C. Liu (2011), "A Genetic Algorithm Enhanced by Dominance Properties for Single Machine Scheduling Problems with Setup Costs," accepted by International Journal of Innovative Computing, Information and Control, 7(4).

3.3 DPs of Unrelated Parallel Machine Scheduling Problems

3.3.1 Problem Definition

A Mixed Integer Program (MIP) is formulated to find optimal solutions for the Unrelated Parallel Machine Scheduling Problems with Sequence-Dependent Times.

$$Minimize \quad C_{\max} \tag{16}$$

subject to

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{k=1}^{m} x_{i,j,k} = 1 \ \ \forall j = 1, ..., n \tag{17}$$

$$\sum_{\substack{i=0 \\ j \neq h}}^{n} x_{i,h,k} - \sum_{\substack{j=0 \\ j \neq h}}^{n} x_{h,j,k} = 0 \ \ \forall h = 1, ..., n, \quad \forall k = 1, ..., m \tag{18}$$

$$C_j \geq C_i + \sum_{k=1}^{m} x_{i,j,k}(S_{i,j,k} + p_{j,k}) + M\left(\sum_{k=1}^{m} x_{i,j,k} - 1\right) \quad \forall i = 0, ..., n \ \ \forall j = 1, ..., n \tag{19}$$

$$\sum_{j=0}^{n} x_{0,j,k} = 1 \quad \forall k = 1, ..., m \tag{20}$$

$$x_{i,j,k} \in \{0,\ 1\} \quad \forall i = 0, ..., n, \quad \forall j = 0, ..., n, \quad \forall k = 1, ..., m \tag{21}$$

$$C_0 = 0 \tag{22}$$

$$C_j \geq 0 \ \forall j = 1, ..., n \tag{23}$$

where, C_j: Completion time of job *j* p_{jk}: Processing time of job *j* on machine *k* S_{ijk}: Sequence-dependent setup time to process job *j* after job *i* on machine *k* S_{0jk}: Setup time to process job *j* first on machine *k* x_{ijk}: 1 if job *j* is processed directly after job *I* on machine *k* and 0 otherwise x_{0jk}: 1 if job *j* is the first job to be processed on machine *k* and 0 otherwise x_{j0k}: 1 if job *j* is the last job to be processed on machine *k* and 0 otherwise *M*: a large positive number.

The objective (16) is to minimize the makespan. Constraints (17) ensure that each job is scheduled only once and processed by one machine. Constraints (18) make sure that each job must neither be preceded nor succeeded by more than one job. Constraints (19) are used to calculate completion times and to ensure that no job can precede and succeed the same job. Constraints (20) ensure that no more than one job can be scheduled first at each machine. Note that there is no need for another set of constraints to guarantee that only one is scheduled last on each machine because this is guaranteed by constrains (20) in conjunction with (18). Constraints (21) specify that the decision variable *x* is binary over all domains. Constraints (22) state that the completion time for the dummy job 0 is zero and constraints (23) ensure that completion times are non-negative. Optimal solutions for the problem then can be obtained by solving the MIP software solver.

3.3.2 Derivations of Dominance Properties

We consider the problem of scheduling *n* jobs into unrelated parallel machines and to derive the dominance properties (necessary conditions) of the optimal schedule. In this section, we use the objective function ($Z(\prod)$) for the makespan of schedule $\prod$. In order to derive the dominance properties for schedule$\prod$, we consider interchanging two jobs on the same machine or on different machines to prove some intermediate results.

[j]: The job is at position [j]

$P_{[j][k]}$: The processing time of the job at position [j] on machine [*k*]

$S_{[i][j][k]}$: The setup time of the job at position [j] is after the job [i] on machine [*k*]

$AP_{[i][j][k]}$: The adjusted processing time of the job at position [j] is after the job [i] on machine [*k*]. Thus, $AP_{[i][j][k]}$ is actually equal to $P_{[j][k]}$ plus$S_{[i][j][k]}$.

$\mathrm{C_{k1}}$:The completion time on $\mathrm{k_1}$

$\mathrm{G_{1[k]}}$: The job set before job [i] on machine *k*

$\mathrm{G_{2[k]}}$: The job set between job [i] and job [j+1] on machine *k*

$\mathrm{G_{3[k]}}$: The job set after job [i] on machine *k*

Due to the page limit, please refer to the following paper:

1. Chang, P. C., and S. H. Chen (2010), "Integrating Dominance Properties with Genetic Algorithms for Parallel Machine Scheduling Problems with Setup Times," accepted by Applied Soft Computing Journal.

References

[1] U. Aickelin, E. K. Burke, and J. Li, "An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering," *Journal of the Operational Research Society*, vol. 58, no. 12, pp. 1574–1585, 2007.

[2] P. A. N. Bosnian and D. Thierens, "Permutation Optimization by Iterated Estimation of Random Keys Marginal Product Factorizations," *Parallel Problem Solving from Nature-Ppsn VII: 7th International Conference, Granada, Spain, September 7-11, 2002: Proceedings*, 2002.

[3] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[4] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A Survey of Optimization by Building and Using Probabilistic Models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[5] R. Baraglia, J. Hidalgo, R. Perego, I. CNUCE, and P. CNR, "A hybrid heuristic for the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 6, pp. 613–622, 2001.

[6] G. Harik, F. Lobo, and D. Goldberg, "The compact genetic algorithm," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 287–297, 1999.

[7] H. Muhlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," *Lecture Notes in Computer Science*, vol. 1141, pp. 178–187, 1996.

[8] Q. Zhang, J. Sun, and E. Tsang, "An Evolutionary Algorithm With Guided Mutation for the Maximum Clique Problem," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 2, pp. 192–200, 2005.

[9] P. C. Chang, S. H. Chen, and C. Y. Fan, "Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 767–777, 2008.

[10] S. H. Chen, P. C. Chang, and Q. Zhang, "Self-Guided genetic algorithm," *Lecture Notes in Artificial Intelligence*, vol. 5227, pp. 292–299, 2008.

[11] Q. Zhang, J. Sun, and E. Tsang, "Combinations of estimation of distribution algorithms and other techniques," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 273–280, 2007.

[12] C. Y. Y.D. Kim, "Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates," *Naval Research Logistics*, vol. 41, pp. 913–933, 1994.

[13] J. M. S. Valente and R. A. F. S. Alves, "Heuristics for the early/tardy scheduling problem with release dates," *International Journal of Production Economics*, vol. 106, no. 1, pp. 261–274, 2007.

[14] P. S. Ow and T. E. Morton, "The Single Machine Early/Tardy Problem," *Management Science*, vol. 35, no. 2, pp. 177–191, 1989.

[15] P. C. Chang, S. H. Chen, and V. Mani, "A Hybrid Genetic Algorithm with Dominance Properties for Single Machine Scheduling with Dependent Penalties," *Applied Mathematical Modeling*, vol. 33, no. 1, pp. 579–596, 2009.

[16] S. Chen, P. Chang, and Q. Zhang, "A self-guided genetic algorithm for flowshop scheduling problems," in *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*. Institute of Electrical and Electronics Engineers Inc., The, 2009, pp. 471–478.

[17] F. Sourd and S. Kedad-Sidhoum, "The One-Machine Problem with Earliness and Tardiness Penalties," *Journal of Scheduling*, vol. 6, no. 6, pp. 533–549, 2003.

[18] P. Chang, S. H. Chen, and C. Y. Fan, "Generating Artificial Chromosomes by Mining Gene Structures with Diversity Preservation for Scheduling Problems," *accepted by Annals of Operations Research*, 2009.