Two-level Rolling Procedure Based on Dummy Schedule for Dynamic Scheduling Problem with Incomplete Global Information[1)]

WANG Bing[1,2] XI Yu-Geng[2]
[1](*School of Information Engineering, Shandong University at Weihai, Weihai* 264209)
[2](*Institute of Automation, Shanghai Jiaotong University, Shanghai* 200030)
(E-mail: wangbing@sdu.edu.cn)

Abstract This paper addresses the single-machine scheduling problem with release times minimizing the total completion time. Under the circumstance of incomplete global information at each decision time, a two-level rolling scheduling strategy (TRSS) is presented to create the global schedule step by step. The estimated global schedules are established based on a dummy schedule of unknown jobs. The first level is the preliminary scheduling based on the predictive window and the second level is the local scheduling for sub-problems based on the rolling window. Performance analysis demonstrates that TRSS can improve the global schedules. Computational results show that the solution quality of TRSS outperforms that of the existing rolling procedure in most cases.

Key words Two-level rolling scheduling, dummy schedule, preliminary scheduling, local scheduling

1 Introduction

Scheduling problems are sorted as deterministic and dynamic problems according to whether the global information about jobs and machines is known a priori or not. The optimal solution of dynamic or large-size scheduling problem is impossibly obtained because of incomplete global information and computational complexity. When the predictive control principle[1] was extended to scheduling problems, the rolling horizon scheduling procedures[2~6], which can just deal with computational complexity and uncertain information, were developed. In [2], a generic rolling scheduling mechanism based on initial schedule was established and the global performances were analyzed, however, the discussed scheduling circumstance was deterministic. In fact, for dynamic scheduling problems with release times, jobs arriving in further future are usually unknown at decision moments due to the incomplete global information. Rolling horizon procedures (RHPs) addressed such dynamic circumstance based on local optimal sub-problems in [3] and the computational results show effective. However, the solutions are sometimes bad due to no global performance analysis.

It is difficult to analyze the global performances for dynamic scheduling in finite horizon. In this paper, we will extend the rolling mechanism of [2] to address dynamic scheduling problems with incomplete global information, present a kind of two-level rolling scheduling strategy, and analyze the global performances.

2 TRSS for dynamic $1/r_i/\Sigma C_i$

The scheduling problem model $1/r_i/\Sigma C_i$ indicates that n jobs are to be scheduled for a single machine to minimize the total completion time and each job has a release time r_i and a processing time p_i. This problem is strongly NP-hard[7]. We assume that n is a finite integer.

2.1 Dummy initial schedule

The involved definitions and detail descriptions of rolling scheduling based on initial schedule were presented in [2]. There are two stages, *i.e.*, firstly an initial schedule is established for all jobs and secondly a rolling scheduling procedure is performed based on the initial schedule. At each decision moment, the global schedule is transformed by the local scheduling. A schedule before the local scheduling is called a previous schedule and the one after the local scheduling is called the current schedule. Here the initial schedule provides not only a sequence of jobs entering rolling windows but also a base of formulating global schedules. It helps to analyze global performances.

1) Supported by National Natural Science Foundation of P.R. China (60274013, 60474002), Shanghai Development Foundation for Science and Technology (04DZ11008), Science Research Foundation of Shandong University at Weihai (XZ2005001)
Received November 3, 2004; in revised form October 14, 2005

A global schedule will be unlikely specified in a dynamic scheduling problem until global information is completely known. At the beginning, when all jobs are unknown, we can create a dummy initial schedule, denoted as $\tilde{D}$. $\tilde{D}$ is made by the dispatching rule FIFO (First in first operating), where jobs are sequenced from small to large in job release times as well as job processing times (whenever the release times are the same). In a dummy schedule, a job is just a symbol and not specified until the job is predicted. Though any dispatching rules can be used to create a dummy schedule, only FIFO can make the job sequence consistent with the sequence of jobs to be predicted. This is important to the following analysis. We can provide an estimated schedule for unknown jobs based on dummy schedule so that global performances can be estimated with incomplete information.

2.2 Preliminary scheduling for predictive window

Jobs are predicted step by step with the decision moment going forward. At each moment, a predictive window, which consists of all unfixed jobs that have been predicted, is identified. At t, if we let the size of predictive horizon be T and the physical decision moment be u_t, the predictive horizon is specified to be $[u_t, u_t+T]$. The predictive window can be identified by predictive horizon based on the previous schedule at each moment.

At $t=1$, the global previous schedule is just the dummy initial schedule $\tilde{D}$, where the jobs arriving during the predictive horizon $[0,T]$ are predicted. All known jobs constitute the predictive window $F(1)$ which forms the previous schedule, denoted as $D_Y(F(1))$. Though $D_Y(F(1))$ comes from the beginning of $\tilde{D}$, it is not a partial dummy schedule any more and becomes a partial known schedule. The unknown job set following $F(1)$, denoted as $\tilde{F}(1)$, forms the corresponding previous schedule $D_Y(\tilde{F}(1))$ which is still a partial dummy schedule. We identify previous schedules by subscript Y. Generally, at t, the global previous schedule $D_Y(t)$ comes from the global current schedule of $t-1$, which consists of two partial schedules. The front is the fixed partial schedule $D(S(t-1))$ and the following is a partial schedule to be rescheduled. The predictive window $F(t)$ consists of all known jobs from the latter with $r_i \leqslant u_t+T$, for which the corresponding partial schedule is denoted as $D_Y(F(t))$. The unknown jobs following $F(t)$ constitute the job set $\tilde{F}(t)$ and the corresponding schedule $D_Y(\tilde{F}(t))$ is still a dummy schedule. At t, the global schedule is estimated to be the global previous schedule $D_Y(t)=D(S(t-1))+D_Y(F(t))+D_Y(\tilde{F}(t))$.

The preliminary scheduling is performed aiming at the previous schedule $D_Y(F(t))$, where $D_Y(F(t)) = D_R(K\bar{L}(t-1))+D_Y(\bar{K}(t-1))$. $D_R(K\bar{L}(t-1))$ is the remaining partial schedule unfixed in the local solution at $t-1$ and $D_Y(\bar{K}(t-1))$ represents the partial schedule of $F(t)$ outside the rolling window $K(t-1)$, shown as Fig. 1 (The interval lengths represent the processing times of the partial schedules and the dashed lines represent the dummy partial schedules.)

The previous schedule $D_Y(F(t))$ is transformed into the preliminary schedule $D_P(F(t))$ by the following preliminary scheduling algorithm:

1) At $t=1$, compute the completion time $C^Y_{F(1)}$ and the performance $J^P_{F(1)}$ of $D_Y(F(1))$; generate the preliminary schedule $D_P(F(1))$ through rescheduling $F(1)$ by use of SPT[8], compute the completion time $C^P_{F(1)}$ and the performance $J^P_{F(t)}$ of $D_P(F(1))$;

2) When $t>1$, compute the completion time $C^Y_{F(t)}$ and the performance $J^P_{F(t)}$ of $D_Y(F(t))$; transform $D_Y(\bar{K}(t-1))$ into $D_P(\bar{K}(t-1))$ by use of SPT; then the preliminary schedule for $F(t)$ is $D_P(F(t))=D_R(K\bar{L}(t-1))+D_P(\bar{K}(t-1))$; compute the completion time $C^P_{F(t)}$ and the performance $J^P_{F(t)}$ of $D_P(F(t))$;

3) Keep $D_Y(\tilde{F}(t))$ invariant.

After preliminary scheduling, the global schedule is estimated to be the preliminary schedule $D_P(t)=D(S(t-1))+D_P(F(t))+D_Y(\tilde{F}(t))$.

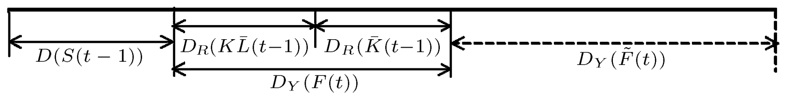
2.3 Local scheduling for rolling window

At t, the rolling window $K(t)$ is specified to be a set of the first k jobs from $D_P(F(t))$, $k=\min\{\kappa, |F(t)|\}$, where κ is the parameter of the rolling window and used to control the size of the rolling window, $|F(t)|$ represents the number of jobs in $F(t)$. The global schedule can be thought to be $D_P(t)=D(S(t-1))+D_P(K(t))+D_P(\tilde{K}(t))$, where $D_P(K(t))$ is the preliminary schedule for $K(t)$ and $D_P(\tilde{K}(t))$ is the preliminary schedule for the set $\tilde{K}(t)$, which is formed by jobs following $K(t)$.

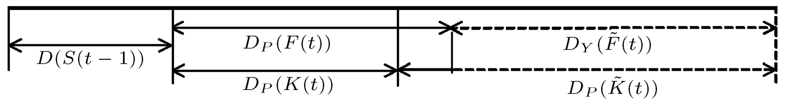
Local scheduling aiming at $D_P(K(t))$ is performed based on terminal penalty (TP) sub-problem in [2], where we just need to know the jobs of $K(t)$ and the number of jobs in $\tilde{K}(t)$. There is no need for job information in $\tilde{K}(t)$. That just adapts to the circumstance of incomplete global information.

Let the solution for TP sub-problem be $D_R(K(t))$. $D_P(\tilde{K}(t))$ is shifted by $\Delta C^R_{K(t)}$ just like that in [2], where $\Delta C^R_{K(t)} = \max\{0, C^R_{K(t)} - C^P_{K(t)}\}$ and $C^R_{K(t)}$ represents the completion time of $D_R(K(t))$. After local scheduling, the global schedule is estimated to be the current schedule $D_R(t) = D_R(S(t-1)) + D_R(K(t)) + D_R(\tilde{K}(t))$.

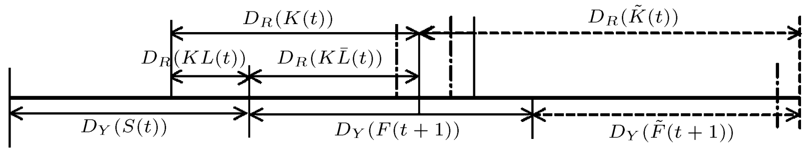
The first $\lambda(1 \leqslant \lambda < k)$ jobs in $D_R(K(t))$ form the set $KL(t)$, which constitute the partial schedule $D_R(KL(t))$. $D_R(KL(t))$ is fixed and merged into $D(S(t-1))$ at t. The remaining partial schedule of $D_R(K(t))$ is denoted as $D_R(K\bar{L}(t))$. The completion time of $D_R(KL(t))$ is just the physical moment u_{t+1} of $t+1$. $D_R(K\bar{L}(t))$ is left in the predictive window $F(t+1)$, shown in Fig. 1. $D_R(t)$ is exactly the global previous schedule at $t+1$, *i.e.*, $D_Y(t+1) = D_R(t)$.



a. Previous schedule at t



b. Predictive schedule at t



c. Windows$'$ rolling from t to $t+1$

Fig. 1 Two-level rolling scheduling strategy

Therefore, in TRSS, the rolling procedure is performed based on the first level of preliminary scheduling and the second level of local scheduling. Preliminary scheduling is implemented for some jobs by dispatching rules and only requires a low computational cost. Though the local scheduling requires to optimally solve sub-problems, the sizes of sub-problems are strictly limited. Therefore, the computational complexity of TRSS is essentially no higher than that of RHP [3].

3 Performance analysis of TRSS

In TRSS, we cannot obtain the specific value of global performance until the last decision moment because we cannot know job information of $\tilde{F}(t)$. However, global performances are just intermediate results during the rolling procedure. Actually, we pay attention to the relative varying trend rather than the specific value of global performance. Since job sequence for $\tilde{F}(t)$ comes from the dummy initial schedule that does not vary during the rolling procedure, the estimated global schedule is consistent and the estimated global performance is comparable at different moments. For $1/r_i/\Sigma C_i$, due to the separable criteria we can decompose the global performance according to different schedule partitions.

3.1 Performance analysis of preliminary scheduling

Lemma 1. For a scheduling problem formulated as $1/r_i/\Sigma C_i$, the SPT solution and the FIFO solution have the same least completion time among all feasible solutions and the performance of SPT solution is no worse than that of FIFO.

Proof. Assume N is the set of all jobs in a scheduling problem formulated as $1/r_i/\Sigma C_i$. The SPT solution and FIFO solution are denoted as D^S and D^F, respectively. Let the beginning times of D^S and D^F be B_S and B_F, the completion times of D^S and D^F be C_S and C_F, respectively.

If for any job $i \in N$ let $r_m = \min\limits_{i \in N} r_i$, then $B_S = B_F = \max(u, r_m)$, where u is the machine idle time before process beginning. In D^S or D^F, because the machine will be idle only if no job arrives, they have the same total idle time $\varOmega$ for the same problem. Therefore $C_S = B_S + \sum_{i \in N} P_i + \varOmega = C_F$. Because the machine will not abandon any jobs that have arrived while waiting for a future job according to

SPT or FIFO, no extra idle time is inserted in D^S and D^F. Therefore the completion time of D^S and D^F are the same, which is the least completion time among all feasible schedules for N.

In the following, let a FIFO solution be $D^F = (1, 2, \cdots, i, \cdots, n)$, where i is the index number of position in D^F and represents the job in this position. Assume job i is the first job following job j (*i.e.*, $r_i \geqslant r_j$) that has $p_i \leqslant p_j$ and $r_i \leqslant b_j$, where b_j is the begining time of job j. Then for any job $x(j < x < i)$, we have $r_x \leqslant b_j$ and $p_x \geqslant p_j$. Therefore, we have $p_i \leqslant p_x$. Because no idle time exists between jobs j and i, exchanging the positions of job i and its previous job $i-1$ will not increase the completion times of the two jobs as well as the following jobs, and furthermore will not increase the performance of D^F. Go on exchanging i with its previous job in the same manner until exchanging i with j. After exchanging all pair of jobs like i and j in D^F, we will obtain the SPT solution D^S for the same problem. Because all such exchanges do not increase the performance, the performance of D^S will not be worse than that of D^F. In addition, because the idle time in the solutions dose not vary during exchange, D^S and D^F have the same completion time. □

We define this kind of consecutive exchange for jobs between i and j in the aforementioned proof non-increasing exchange. Then we can have the following lemma.

Lemma 2. In TRSS, through non-increasing exchanges for a previous schedule, the completion time of schedule will not vary and the performance of schedule will not increase.

Theorem 1. In TRSS, under preliminary scheduling algorithm for predictive window, the completion time $C^P_{F(t)}$ of $D_P(F(t))$ is equal to the completion time $C^Y_{F(t)}$ of $D_Y(F(t))$ and the performance $J^P_{F(t)}$ of $D_P(F(t))$ is not worse than that $J^F_{F(t)}$ of $D_Y(F(t))$.

Proof. According to Lemma 1, the theorem is true at $t = 1$. When $t > 1$, the previous schedule $D_Y(F(t))$ consists of the remaining partial schedule $D_R(K\bar{L}(t-1))$ and the following partial schedule $D_R(\bar{K}(t-1))$ of $K(t-1)$. Local scheduling cannot change $D_R(\bar{K}(t-1))$, which actually is a sequence by SPT and FIFO at $t-1$. Therefore, no inserted idle time exists in $D_R(\bar{K}(t-1))$. We can perform non-increasing exchanges for $D_R(\bar{K}(t-1))$ and obtain the SPT solution for $\bar{K}(t-1)$, which is just the preliminary schedule $D_P(F(t))$. Due to Lemma 2, $C^P_{F(t)} = C^Y_{F(t)}$, and $J^P_{F(t)} \leqslant J^F_{F(t)}$. □

Theorem 1 can ensure the following conclusion (proof is omitted).

Theorem 2. In TRSS, at each t, if the previous schedule $D_Y(t)$ is feasible, the preliminary schedule $D_P(t)$ is feasible and the performance $J^P(t)$ of $D_P(t)$ is no worse than the performance $J^Y(t)$ of $D_Y(t)$.

3.2 Performance analysis of local scheduling

In the second level of TRSS, because the previous schedule $D_Y(1)$ is just the dummy initial schedule $\tilde{D}$, which is a feasible solution by FIFO, Theorem 3 can be concluded, where proof is similar to that of Theorem 1 in [2]. We omit the proof due to the limitation of paper length. Further more Theorem 4 is concluded due to Theorems 2 and 3.

Theorem 3. In TRSS, if the dummy initial schedule is feasible, at each t, the estimated global schedule is feasible and the performance $J^R(t)$ of the current schedule $D_R(t)$ is no worse than the performance $J^P(t)$ of the predictive schedule $D_P(t)$.

Theorem 4. In TRSS for dynamic $1/r_i/\Sigma C_i$, as job information is predicted from one decision moment to another, feasible dummy partial schedule is becoming a known schedule, the estimated global schedule is always feasible at each decision moment, and the performance of estimated schedule is reaching the ultimately realized value which is the best one among all estimated global performances.

4 Computational experiments and results analysis

In this section, the experiment consists of two parts. TP sub-problem in local scheduling was solved by Branch and Bound developed in [4]. All procedures were coded by C language in Visual c++ 6.0 and all tests ran on a computer with Pentium 4-M CPU 1.80 GHz and Windows XP operating environment. Problems were randomly generated using a format similar to that used by [3].

In the first part of experiment, the influence of initial schedule quality on TRSS was investigated to show the effectiveness of TRSS. Let the size of predictive horizon $T = \rho * TT$, where ρ is the parameter that is used to control how rapidly jobs are expected to arrive. A total of ten different ρ values, varying from 0.20 to 3.00, were considered. The rolling parameters (κ, λ) were specified to be $(17, 2)$. Let $TT = 400$. The experiment was conducted for 50-job and 250-job problems, respectively. Twenty problems were respectively tested for the same size problems with ten ρ values and a total of four hundreds problems were tested. Since the optimal solution is unlikely to be obtained for a large size problem, FIFO and TRSS solutions were compared with the lower bound (LB) of the optimal solution, where LB was computed according to the LB algorithm in [9]. The percentage improvement of FIFO and SPT over LB were calculated as (FIFO-LB)/LB and (TRSS-LB)/LB, respectively. Table 1 presents the computational results, where each entry is the average coming from 20 instances of one case.

Table 1 Average percentage improvement of FIFO and TRSS over LB

ρ		0.20	0.40	0.60	0.80	1.00	1.25	1.50	1.75	2.00	3.00
50-job problems	FIFO	38.1	36.5	22.6	16.9	6.23	3.75	3.34	1.24	0.48	0.22
	TRSS	1.13	2.66	4.58	5.17	3.24	2.49	1.83	0.42	0.36	0.16
250-job problems	FIFO	43.7	36.4	25.8	17.8	12.7	1.81	1.28	0.76	0.38	0.07
	TRSS	5.62	4.09	3.73	4.82	6.07	1.12	0.89	0.51	0.26	0.04

If the percentage improvement is large, it is indicated that the solution is bad. Table 1 shows that FIFO and TRSS are getting better as ρ increases. When ρ is little, FIFO is very bad. However, the ultimate solution after TRSS is not so bad. This phenomenon demonstrates that the worse the initial schedule is, the larger the improvement of TRSS is. Therefore, we needn′t worry about the solution quality of TRSS due to poor initial schedule when we just care about the ultimate solution. Whatever the initial schedule is, the ultimate solution after TRSS is not so bad.

In the second part of the experiment, TRSS was compared with RHP to show the advantage of TRSS. The rolling parameters (x, y, z) of RHP were specified to be (12,5,2). TRSS and RHP were used to solve each problem respectively. If TRSS was better than RHP, the percentage improvement was calculated as (RHP-TRSS)/TRSS. If RHP was better than TRSS, the percentage improvement was calculated as (TRSS-RHP)/RHP. The computational results are shown in Table 2 and Table 3, where "num." represents the number of times of corresponding cases out of 20 problems, "ave." represents the average percentage improvement of the corresponding cases, and "max." represents the maximum percentage improvement of corresponding cases.

Table 2 Comparison of TRSS with RHP: 50-job problems

ρ	TRSS better than RHP			RHP better than TRSS		
	Num.	Ave. (%)	Max. (%)	Num.	Ave. (%)	Max. (%)
0.20	20	1.98	3.61	0	0	0
0.40	20	1.32	1.23	0	0	0
0.60	18	0.32	0.89	0	0	0
0.80	20	0.15	0.24	0	0	0
1.00	20	0.12	0.19	0	0	0
1.25	12	0.11	0.23	0	0	0
1.50	9	0.03	0.28	0	0	0
1.75	0	0	0	0	0	0
2.00	0	0	0	0	0	0
3.00	0	0	0	0	0	0

Table 3 Comparison of TRSS with RHP: 250-job problems

ρ	TRSS better than RHP			RHP better than TRSS		
	Num.	Ave. (%)	Max. (%)	Num.	Ave. (%)	Max. (%)
0.20	20	0.83	1.28	0	0	0
0.40	20	0.65	1.36	0	0	0
0.60	20	0.44	0.81	0	0	0
0.80	20	0.37	0.76	0	0	0
1.00	20	0.22	0.58	0	0	0
1.25	18	0.02	0.07	1	0.01	0.01
1.50	16	0.01	0.03	0	0	0
1.75	19	0.01	0.02	0	0	0
2.00	17	0.01	0.02	0	0	0
3.00	0	0	0	0	0	0

The results show that TRSS are consistently better than RHP for almost all ρ-value problems. In particular, it is obviously observed that TRSS outperforms RHP for $\rho \leqslant 1.250$, which is just the case that RHP performs poorly in [3]. It demonstrates that TRSS gets a definite advantage over RHP.

5 Conclusions

For dynamic scheduling problems formulated as $1/r_i/\Sigma C_i$ with incomplete global information, a kind of two-level rolling scheduling strategy is developed. Dummy initial schedule is established to estimate the schedule for unknown jobs. During the rolling procedure, global schedules are estimated based on dummy initial schedule. In the first level, preliminary scheduling is performed sufficiently utilizing known job information in a predictive window and does not increase the completion time or the performance of predictive window. The second level is the local scheduling for the rolling window, where jobs are selected into sub-problems based on the preliminary schedule. TP sub-problem ensures that the current schedule is no worse than the preliminary schedule at each decision moment. The global performance analysis demonstrates that TRSS is performed with a trend of improving the global performance and it theoretically limits the worst cases of solution. Experiments testified that whatever the initial schedule by FIFO was, the ultimate solution of TRSS would not be too bad. TRSS obviously improved the solutions in problems where RHP were bad, and TRSS was consistently better than RHP in most cases.

References

1 Xi Y G. Predictive control of general control problems under dynamic uncertain environment. *Control Theory and Applications*, 2000, **17**(5): 665∼670

2 Wang B, Xi Y G, Gu H Y. Terminal penalty rolling scheduling based on an initial schedule for single-machine scheduling problem. *Computers and Operations Research*, 2005, **32**(11): 3059∼3072

3 Chand S, Traub R, Uzsoy R. Rolling horizon procedures for the single machine deterministic total completion time scheduling problem with release dates. *Annals of Operations Research*, 1997, **70**: 115∼125

4 Wang B, Xi Y G, Gu H U Y. An improved rolling horizon procedure for single-machine scheduling with release times. *Control and Theory*, 2005, **20**(3): 257∼260

5 Ovacik I M, Uzsoy R. Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 1994, **32**(6): 1243∼1263

6 Fang J, Xi Y G. Rolling horizon job shop rescheduling strategy in the dynamic environment. *International Journal of Advanced Manufacturing Technology*, 1997, **13**: 227∼232

7 Lawler E L, Lenstra J K, Rinnooy Kan A H G, Shmoys D B. Sequencing and scheduling: Algorithm and complexity. In: S.C. Graves, A.H.G. Rinnooy Kan, and P. Zipkin (Eds), Handbooks in Operations Research and Management Science Vol. 4: Logistics of Production and Inventory. North-Holland, Amsterdam: Elsevier, 1993

8 Chandra R. On $n/1/\bar{F}$ dynamic deterministic problems. *Naval Research Logistics Quarterly*, 1979, **26**: 537∼544

9 Dessouky M I, Deogun J S. Sequencing jobs with unequal ready times to minimize mean flow time. *SIAM Journal on Computing*, 1981, **10**(1): 192∼202

WANG Bing Associate professor of Shandong University at Weihai. Received her Ph. D. degree from Shanghai Jiaotong University in 2005. Her research interests include production scheduling and combinatorial optimization

XI Yu-Geng Professor of Shanghai Jiaotong University. Received his Ph. D. degree from Technical University of Munich, Germany in 1984. His research interests include predictive control, large-scale system, and intelligent robotics.