Asian Information–Science–Life
Volume 1, Number 3, pp. 241-248

ISSN 1541-8219

BRANCH AND BOUND ALGORITHM FOR FLEXIBLE FLOWSHOP WITH LIMITED MACHINE AVAILABILITY*

XIJUN WANG AND JINXING XIE†

Abstract. Machines may not be available all the time due to uncertain breakdown or preventive maintenance activities. In this paper we develop a branch and bound algorithm to solve the flexible flowshop scheduling problems with non-resumable availability constraint. Both the machine-based and job-based lower bounds are developed by making use of the lower bounds of the classical flexible fowshop problems. Numerical experiments are conducted and the results show that the branch and bound algorithm is effective when the problem size is not very large.

Key words. flexible flowshop, scheduling, limited availability, branch and bound

1. Introduction. Flexible flowshop consists of a set of machine centers with identical parallel machines. Since the classical flowshop scheduling problem $F_2 \mid \cdot \mid C_{max}$ is known to be solvable in polynomial time [9], but the problem $F_3 \mid \cdot \mid C_{max}$ is proved to be strongly $\mathcal{NP}$-hard [5], the flexible flowshop scheduling problems with makespan objective have attracted a great deal of attention in the last decades. The problem $F_2(P) \mid \cdot \mid C_{max}$ is $\mathcal{NP}$-hard in the strong sense, even in the case where there are only two machines in one stage and only a single machine in the other stage [7]. Various approximation algorithms for the flexible flowshop problems were designed [15], [3], for some of which the corresponding worst case performance analysis was provided and the average case performance was showed. Brah and Hunsucker [2] presented a branch and bound algorithm to solve flexible flowshop scheduling problems. Hall [6], Schuurman and Woeginger [14] discussed the existence of polynomial time approximation schemes for flexible flowshop scheduling problems.

Most literature on scheduling assumes that machines are available all the time. In practice, however, a machine may not always be available in the scheduling horizon, for example, due to machine breakdowns (stochastic) or preventive maintenances (deterministic). Besides, if a machine should continue to process those unfinished jobs that were scheduled on the machine in the preceding planning period, it is unavailable at the beginning of the following period. This paper studies the scheduling problem with limited machine availability under the deterministic case, i.e., all the input data of the problem, including the unavailable time intervals, are known in advance.

Scheduling problems with limited machine availability have been studied to a less extent. Adiri et al. [1] showed that a single machine problem with machine breakdowns is $\mathcal{NP}$-hard if the breakdowns are known in advance. Schmidt [13] discussed scheduling problems with unavailable intervals for a parallel machine problem. Hwang and Chang [8] proved that the makespan of the LPT (Longest Processing Time) schedule is bounded by twice the optimal makespan if no more than half of the machines are allowed to be shutdown simultaneously. Lee [11] proved that the two-machine flowshop problem with makespan objective and one unavailable interval is $\mathcal{NP}$-hard in the ordinary sense, and presented a dynamic programming approach and several heuristics with worst case performance analysis. Kubiak et al. [10] extended the complexity results and proved that the two-machine flowshop problem with arbitrary number of unavailable intervals on one machine is $\mathcal{NP}$-hard in the strong sense.

*This work was supported by the National Natural Science Foundation of China (NSFC Project No. 69904007).

†Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China (jxie@math.tsinghua.edu.cn).

They also proposed a branch and bound algorithm for the problem with arbitrary number of unavailable intervals on both machines. Lee [12] gave the first discussion about the semi-resumable two-machine flow shop model and provided worst case performance analysis for some algorithms. Espinouse, Formanowicz and Penz [4] studied both the resumable and the non-resumable two-machine flow shop models under the no-wait environment. Wang and Xie [16] gave the first study for the two-stage flexible flowshop with limited machine availability. They proved the $\mathcal{APX}$-hardness of the problems with unavailable interval(s) on the whole second stage, or on the whole first stage with more than one machine. They also analyzed the worst case performance of two heuristics for the case where there is only one machine at the first stage and only one unavailable interval on that machine.

In this paper we develop a branch and bound algorithm to solve the flexible flowshop scheduling problems with non-resumable availability constraint. In the next section, we introduce the notations used in the following sections and analyze the characteristics of this problem. Section 3 provides the lower bounds for flexible flowshop with limited machine availability and Section 4 gives the branch and bound algorithm in detail. Numerical experiments are given in Section 5. The results of the computational experience shows that the branch and bound algorithm is effective when the problem size is not very large.

2. Problem characteristics. In the flexible flowshop scheduling problem, all machines at the same machine center are assumed to be identical, i.e. they have the same capability and speed. Thus any two tasks assigned to two machines at the same machine center can be interchanged while keeping the makespan (the maximum completion time) unchanged. However, generally, this property does not hold when the constraint of limited machine availability exists, since the unavailable intervals differ from machine to machine. This property implies that the assumption of identical machines is greatly weaken when the constraint of limited machine availability applies.

According to the theoretical analysis of the complexity and approximability for the flexible flowshop with limited machine availability [16], the most tractable algorithms solving these problems in practice might be heuristics. However, to our surprise, this kind of polynomial-time heuristics will not have any guarantee of finite worst case bounds [16]. In such cases, it's important to provide some benchmarks to the design of heuristics, by making use of optimal algorithms such as branch and bound algorithms.

The flexible flowshop consists of a set of $m \geq 2$ machine centers $[Z_1, Z_2, \ldots, Z_m]$ with center Z_j having $m_j \geq 1$ identical parallel machines $\{M_{j1}, M_{j2}, \ldots, M_{jm_j}\}$. Machine centers are also called stages. There are n jobs $\{J_i \mid 1 \leq i \leq n\}$ to be processed in the flowshop. Function $p(J_i) = [p_{i1}, p_{i2}, \ldots, p_{im}]$ denotes the processing times required by J_i in centers $[Z_1, Z_2, \ldots, Z_m]$ respectively. We assume that unavailable intervals do not overlap on each machine. Denote by $s_{jk,l}$ and $h_{jk,l}$ the starting time and length, respectively, of unavailable interval l on machine M_{jk}, numbered according to the starting times of the unavailable intervals. Each machine can process at most one job at a time and each job can be processed by at most one machine at a time. The objective considered is to minimize the makespan of the schedule in the non-resumable case, although our branch and bound algorithm can also be used to solve the problems with other performance measures. Using the three-field notation for scheduling problems, the problem considered in this paper can be written as $F(P) \mid s_{jk,l}, h_{jk,l} \mid C_{max}$, where $F(P)$ means the flowshop with parallel machines.

Denote by N the set of all jobs, i.e., $N = \{J_i \mid 1 \leq i \leq n\}$. When all the jobs have been assigned to machines in all stages, it's easy to see that only the schedules where all the jobs are processed as early as possible need to be considered. Thus the total amount of

all feasible schedules will be $\prod_{j=1}^{m}(n!m_j^n)$.

Supposing that the partial schedule involving all jobs on all machines from stage 1 through stage $j-1$ has been determined, we continue to make the scheduling decision on stage j for all the jobs. Given a ordered subset A of N, let $S_j(A)$ be the partial schedule through stage $j-1$, along with the sequence of the jobs in subset A which have been assigned to machines for processing at stage j. Denote by $C_k(S_j(A))$ the completion time of the partial schedule $S_j(A)$ on machine M_{jk}, and let $C_k(S_j(A)) = 0$ if no job has been assigned to machine M_{jk} in the partial schedule $S_j(A)$. Denote by $JC_j(J_q)$ the completion time of job J_q at stage j, and let $JC_0(J_q) = 0$ for each job J_q in N.

For a job $J_q \notin A$, let $A' = A \cup \{J_q\}$ and $S_j(A')$ represent the augmented partial schedule formed by appending job J_q to $S_j(A)$ at stage j. It is obviously that the earliest possible time for job J_q to start to process on machine M_{jk} is

$$ET(S_j(A), J_q) = \max\{C_k(S_j(A)), JC_{j-1}(J_q)\}. \tag{2.1}$$

However, due to that there are unavailable intervals on the machine, the job may be delayed to be processed. Following these understandings, we have

$$C_k(S_j(A')) = \begin{cases} ET(S_j(A), J_q) + \Delta_{qj}(S_j(A)) + p_{qj}, \\ \qquad \text{if } J_q \text{ is assigned to } M_{jk}; \\ C_k(S_j(A)), \text{otherwise}, \end{cases} \tag{2.2}$$

where $\Delta_{qj}(S_j(A))$ means the delay time for job J_q on machine M_{jk} due to the unavailability constraint which results in that the job J_q can not be completed between $ET(S_j(A), J_q)$ and the unavailable intervals following $ET(S_j(A), J_q)$ on machine M_{jk}.

If no unavailable interval exists from $ET(S_j(A), J_q)$ to $ET(S_j(A), J_q) + p_{qj}$ on machine M_{jk}, then job J_q need not to be delayed and $\Delta_{qj}(S_j(A)) = 0$. Otherwise, J_q will be delayed for processing. Under this situation, the first unavailable interval following $ET(S_j(A), J_q)$ is

$$l_1 = \min\{l : 0 \le s_{jk,l} - ET(S_j(A), J_q) < p_{qj}\}. \tag{2.3}$$

However, J_q may be further delayed due to the unavailable intervals following the unavailable intervals l_1 on machine M_{jk}. Let l_2 be the first unavailable interval after which J_q can start to process on the machine immediately. Specifically, it can be calculated as

$$l_2 = \min\{l : l \ge l_1, s_{jk,l+1} - (s_{jk,l} + h_{jk,l}) \ge p_{qj}\}. \tag{2.4}$$

Therefore, the delayed time for J_q is

$$\Delta_{qj}(S_j(A)) = (s_{jk,l_2} + h_{jk,l_2}) - ET(S_j(A), J_q). \tag{2.5}$$

Following these notations, the makespan (the maximum completion time for all the jobs on all machine centers) is minimized if and only if $\max_k\{C_k(S_m(N))\}$ is minimized. That is to say, the objective function of the problem is

$$C_{max} = \max_k\{C_k(S_m(N))\}. \tag{2.6}$$

3. Lower bounds. In the following subsections, we propose lower bounds for the completion time of all jobs at all stages.

3.1. Machine-based bound. Given a partial schedule $S_j(A)$, the unprocessed workload of the unprocessed jobs (the jobs in set $N - A$) at stage j can be utilized to get a lower bound of the completion time of all jobs at stage j, and therefore a lower bound of the completion time of all jobs at all stages.

Denote K_1 the set of machines at machine center Z_j to which some tasks has been assigned in partial schedule $S_j(A)$, and K_2 the set of other machines at machine center Z_j. For a machine $k \in K_1$, the earliest possible time for this machine to process the jobs in set $N - A$ is

$$T_{1k} = \max\{\min_{J_q \in N-A} JC_{j-1}(J_q), C_k(S_j(A)\}. \tag{3.1}$$

For a machine $k \in K_2$, the earliest possible time for this machine to process the jobs in set $N - A$ is

$$T_2 = \min_{J_q \in N-A} JC_{j-1}(J_q). \tag{3.2}$$

It is easy to see that the completion time for all the jobs through stage j is at least

$$T_3 = \{\sum_{k \in K_1} T_{1k} + |K_2|T_2 + \sum_{J_q \in N-A} p_{qj}\}/m_j. \tag{3.3}$$

However, due to the unavailability constraint, the completion time may be even longer. Denote $\Delta_j(S_j(A))$ the sum of the times wasted on machines at stage j due to the limited machine availability. Then a lower bound of the completion time of a partial schedule $S_j(N)$ augmented from $S_j(A)$, which just completes the scheduling for all the jobs through stage j, can be expressed as

$$\begin{aligned}
&ACT(S_j(A)) \\
&= T_3 + \Delta_j(S_j(A))/m_j \\
&= \{\sum_{k \in K_1} T_{1k} + |K_2|T_2\}/m_j + \{\sum_{J_q \in N-A} p_{qj}\}/m_j + \Delta_j(S_j(A))/m_j \\
&= \{\sum_{k \in K_1} \max\{C_k(S_j(A)), \min_{J_q \in N-A} JC_{j-1}(J_q)\} \\
&\quad + |K_2| \min_{J_q \in N-A} JC_{j-1}(J_q)\}/m_j \\
&\quad + \{\sum_{J_q \in N-A} p_{qj}\}/m_j + \Delta_j(S_j(A))/m_j.
\end{aligned} \tag{3.4}$$

The first term on the right hand side of the above equation represents the average of the possible earliest time point from which the machines at stage j can process the jobs in $N-A$. The second term represents the remaining average workload at stage j for the unprocessed jobs in $N-A$. The last term represents the average time wasted on machines at stage j due to the limited machine availability. From the previous section, we can see that the wasted time on a machine can be exactly calculated by adding the unprocessed jobs one by one into the partial schedule $S_j(A)$. However, the calculation is time-consuming, and thus we use a lower bound which can be calculated quickly to estimate $\Delta_j(S_j(A))$. Specifically, we think of that $\Delta_j(S_j(A))$ consists of only the following two components: a) The sum of the time lengths of the unavailable intervals for machines in K_1 between the time T_{1k} and the time

T_3; b) The sum of the time lengths of the unavailable intervals for machines in K_2 between the time T_2 and the time point T_3. Therefore, we estimate $\Delta_j(S_j(A))$ as

$$\Delta_j(S_j(A)) = \{\sum_{k\in K_1}\{\sum_{l:T_{1k}\le s_{jk,l}<T_3} s_{jk,l}\} + \sum_{k\in K_2}\{\sum_{l:T_2\le s_{jk,l}<T_3} s_{jk,l}\}\}. \tag{3.5}$$

From these observations, we know that $ACT(S_j(A))$ is a lower bound of the completion time of any partial schedule $S_j(N)$ augmented from $S_j(A)$, i.e.

$$ACT(S_j(A)) \le \max_k C_k(S_j(N)). \tag{3.6}$$

Finally, the machine-based lower bound of complete schedules for all jobs on all stages, augmented from a partial schedule $S_j(A)$, can be calculated as

$$LBM(S_j(A)) = ACT(S_j(A)) + \min_{J_q\in N}\sum_{j'=j+1}^{m} p_{qj'}. \tag{3.7}$$

3.2. Job-based bound. Given a partial schedule $S_j(A)$, the information of jobs in job set A can be utilized to get the earliest starting time of the rest unassigned jobs, and therefore a lower bound of the completion time of all jobs at all stages. For example, Brah and Hunsucker [2] presented a job-based bound

$$LBJ_0(S_j(A)) = \min_k C_k(S_j(A)) + \max_{J_q\in N-A}\sum_{j'=j}^{m} p_{qj'}. \tag{3.8}$$

Using this approach, if some machines have not been assigned any task, the estimate of the earliest possible starting time of the rest unassigned jobs will be zero, and thus the estimate of the completion time of all jobs at all stages will be small. By considering the completion time of the rest unassigned jobs at the remaining stages, the lower-bound estimate of the completion time of the complete schedule augmented by $S_j(A)$ will become larger. Therefore, the job-based bound can be modified to

$$LBJ(S_j(A)) = \begin{cases} \min\{\min_{J_q\in N-A} JC_{j-1}(J_q), \max_{k\in K_1} C_k(S_j(A))\} \\ \quad + \max_{J_q\in N-A}\sum_{j'=j}^{m} p_{qj'}, & \text{if } K_2 \neq \phi; \\ \min_k C_k(S_j(A)) + \max_{J_q\in N-A}\sum_{j'=j}^{m} p_{qj'}, & \text{if } K_2 = \phi. \end{cases} \tag{3.9}$$

3.3. Integrated bound. In any machine center, if the number of jobs to be processed is much larger than that of all the machines, the average completion time of that machine center will generally be easily reached by a schedule, and so the machine-based lower bound will be closer to the optimal completion time of complete schedules augmented from the partial schedule. On the other hand, if the number of jobs to be processed is much smaller than that of all the machines at some machine centers, the estimate of the earliest possible starting time of the remaining unprocessed jobs at that stage will generally be more exact, and so the job-based lower bound will be closer to the optimal completion time of complete schedules augmented from the partial schedule.

Combining the machine-based bound and the job-based bound, we get a integrated lower bound

$$LBC(S_j(A)) = \max\{LBM(S_j(A)), LBJ(S_j(A))\}. \tag{3.10}$$

4. Branch and bound algorithm. In the previous section, we introduced the lower bound of the problem. In this section, we give the detailed procedures of the branch and bound algorithm for the problem.

4.1. Branching scheme. Any partial schedule will be considered to be a sub-branch in our algorithm. According to the analysis in Section 2, the tasks assigned to two machines at the same machine center cannot be interchanged. Otherwise, the makespan of the schedule will be changed. Therefore, each schedule augmented from a $S_j(A)$ will be a possible sub-branches of it. As we know, the effectiveness of the branch and bound algorithm is sensitive to the exactness of the lower bound of each sub-branch. The branching scheme used by [2] can still be used to eliminate the repetitive search of equivalent sub-branches if it's known in advance that there is no unavailability constraint on that machine center. Whenever this is the case, this branching scheme is accepted.

4.2. Procedures of the algorithm. The branch and bound algorithm for solving the flexible flowshop with limited machine availability can be described as follows in detail:

Step 0: Initialize the best current solution.

Step 1: Generate all of the partial schedules in which only one job is assigned to machine M_{11}. Calculate the lower bound for each branch. Push these branches in the lower-bound-decreasing order into the stack in which all branches waiting to be processed are stored.

Step 2: If the stopping criteria is satisfied, then go to Step 6.

Step 3: Pop the branch out of the top of the stack. Generate all the sub-branches of this branch. If these branches are leaf branches, then go to Step 5.

Step 4: Cut off those branches whose lower bounds are no less than the current best makespan. Push the left branches into the stack in the lower-bound-decreasing order. Go to Step 2.

Step 5: Find out the leaf branch with the minimum makespan. Save this branch as the best current solution, and go to Step 2.

Step 6: Output the best current solution and stop.

We can limit the amount of memory needed to $O(n^2m)$ by keeping track of the path from the top level to the current level of the branches, and recording the level numbers of all the branches before they are pushed into the stack.

In practical implementation, we can firstly use other heuristics to get the initial solution in Step 1, because a good initial solution may reduce the possibility for the algorithm visiting to deep-level branches and thus improve the performance of the algorithm. Secondly, the stoping criteria in Step 2 and the cut-off condition in Step 4 could be changed to fit different situations according to the requirements of the practical problems. If the computational time is not critical, the stoping criteria can simply be 'stop when the stack is empty' and the solution obtained in this way is the optimal solution for the problem. In order to end all the computation in a given time, the time limitation could be added to the stoping condition in Step 2. By adding to the cut-off condition in Step 4 a limitation of the difference between the lower bound of each branch and the best current solution, the algorithm will provide a

TABLE 5.1

Computing results of the branch and bound algorithm for the flexible flowshop with limited machine availability. The performance ratio is defined as the ratio between the makespans of the first and the last feasible solutions searched by the algorithm.

n	m	m_j, $j = 1, \cdots, m$	Instances tested	# of feasible solutions	Average run time Hr:Min:Sec	Average # of solutions searched	Average performance ratio
4	2	2,2	10	1.475×10^5	00:00:01	12	1.174
4	4	2,2,2,2	10	2.174×10^{10}	00:00:01	666	1.500
6	2	2,2	10	2.123×10^9	00:00:01	3.117×10^3	2.286
6	3	2,3,2	10	1.115×10^{15}	00:04:02	2.612×10^5	1.623
6	4	2,2,3,2	10	5.136×10^{19}	08:30:02	3.540×10^8	1.627
8	2	3,2	10	2.731×10^{15}	02:43:13	1.279×10^6	1.383

solution with a guarantee of the given precision. Finally, if the algorithm stops once a leaf branch is encountered at the first time, it becomes a kind of heuristic algorithm finding a feasible solution.

5. Numerical experiments. As we have pointed out in the previous sections, the performance of the branch and bound algorithm for flexible flowshop scheduling problems with limited machine availability depends on the exactness of the lower bound. In order to investigate the effectiveness of the algorithm, we have conducted a set of numerical experiments. In these numerical experiments, only the branches with lower bound smaller than 99% of the best current makespan are kept. All of the other branches are cut off. We also make the following reasonable assumptions when generating the testing instances of the problem:

1. The processing times are generated from a normal distribution with a constant mean value.

2. The starting time of the unavailable interval on each machine is generated from a normal distribution. The mean value of the starting times of the unavailable intervals on the first machine center are generated from a uniform distribution between 0 and the average workload of this machine center. The difference of the mean value of any two adjacent machine centers is equal to the mean value of the average workload of the latter machine center.

3. The length of any unavailable interval is generated from a normal distribution.

4. The mean value of the length of any unavailable interval for each machine center is positively proportional to the average work load of that machine center.

5. There is only one unavailable interval on each machine in the planning horizon.

Assumption 1 corresponds to the similarity of jobs to be processed. By modifying the standard deviation of the normal distribution, the difference of processing times can also be implemented. If the unavailable intervals are regarded as machine maintenance times, assumption 2 can be explained by the rationality of the maintenance plan. Assumption 3, similar to the assumption 1, can be explained by the similarity of the maintenance times of those machines at the same machine center. Assumptions 4 and 5 can be explained by the periodicity and volatility of the planning horizon.

The computing results are summarized in Table 5.1. The computation is carried out on an IBM PC of 400MHz. Comparing the average number of end nodes (completed schedules for all the jobs on all the machine centers as feasible solutions) searched with the number of all possible end nodes, we can see that the lower bound developed here significantly reduces the visit to the end nodes. The average ratio of the makespan of the first end node to

the final makespan obtained by the algorithm, with relative error bound of 1%, is about 1.6 according to the last column of Table 5.1. Such an average performance is relatively satisfying for such an $\mathcal{APX}$-hard problem. Thus the branch and bound algorithm proposed here might be useful for the flexible flowshop scheduling problem with limited machine availability when the problem size is not very large. It provides a benchmark for other heuristics solving the problem.

REFERENCES

[1] I. Adiri, J. Bruno, E. Frostig and A. H. G. Rinnooy Kan, *Single machine flow-time scheduling with a single breakdown*, Acta Informatica, 26 (1989), pp. 679–696.

[2] S. A. Brah and J. L. Hunsucker, *Branch and bound algorithm for the flow shop with multiple processors*, European Journal of Operational Research, 51 (1991), pp. 88–99.

[3] B. Chen, *Analisis of classes of heuristics for scheduling a tow-stage flow shop with parallel machines at one stage*, Journal of the Operational Research Society, 46 (1995), pp. 234–244.

[4] M. L. Espinouse, P. Formanowicz and B. Penz, *Complexity results and approximation algorithms for the two machine no-wait flow-shop with limited machine availability*, Journal of the Operational Research Society, 52 (2001), pp. 116–121.

[5] M. R. Garey, D. S. Johnson and R. Sethi, *The complexity of flow shop and job shop scheduling*, Mathematics of Operations Research, 1 (1976), pp. 117–129.

[6] L. A. Hall, *Approximability of flow shop scheduling*, Mathematical Programming, 82 (1998), pp. 175–190.

[7] Hoogeveen, J. A., Lenstra, J. K. and B. Veltman, *Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard*, European Journal of Operational Research, 89 (1996), pp. 172–175.

[8] H.-C. Hwang, and S. Y. Chang, *Parallel machines scheduling with machine shutdowns*, Computers and Mathematics with Applications, 36 (1998), pp. 21–31.

[9] S. M. Johnson, *Optimal two- and three-stage production schedules with setup times included*, Naval Research Logistics Quarterly, 1 (1954), pp. 61–68.

[10] W. Kubiak, J. Błazewicz, P. Formanowicz and G. Schmidt, *A branch and bound algorithm for the two machine flow shops with limited machine availability*, Research Report RA–001/97, Institute of Computing Science, Poznan University of Technology, 1997.

[11] C.-Y. Lee, *Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint*, Operations Research Letters, 20 (1997), pp. 129–139.

[12] ———, *Two-machine flowshop scheduling with availability constraints*, European Journal of Operational Research, 114 (1997), pp. 420–429.

[13] G. Schmidt, *Scheduling on semi-identical processors*, Zeitschrift für Operations Research, 28 (1984), pp. 153–162.

[14] P. Schuurman and G. J. Woeginger, *A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem*, Theoretical Computer Science, 237 (2000), pp. 105–122.

[15] C. Sriskandarajah and S. P. Sethi, *Scheduling algorithms for flexible flowshops: Worst and average case performance*, European Journal of Operational Research, 43 (1989), pp. 143–160.

[16] X. Wang and J. Xie, *Two-stage flexible flowshop scheduling with limited machine availability*, Working paper, Department of Mathematical Sciences, Tsinghua University, Beijing, China, 2001.