

Online Algorithms for Single Machine Schedulers to Support Advance Reservations from Grid Jobs

Bo Li and Dongfeng Zhao

School of Information Science and Engineering, Yunnan University,
Kunming, Yunnan, 650091, China
{libo, dfzhao}@ynu.edu.cn

Abstract. Advance Reservations(AR) make it possible to guarantee the QoS of Grid applications by reserving a particular resource capability over a defined time interval on local resources. However, advance reservations will also cause the processing time horizon discontinuous and therefore reduce the utilization of local resources and lengthen the makespan (i.e., maximum completion time) of non-resumable normal jobs. Single machine scheduling is the basis of more complicated parallel machine scheduling. This study proposed a theoretic model, as well as four online scheduling algorithms, for local single machine schedulers to reduce the negative impact on the utilization of local resources and to shorten the makespan of non-AR jobs resulting from advance reservations for Grid jobs. The performances of the algorithms were investigated from both of the worst case and the average case viewpoints. Analytical results show that the worst case performance ratios of the algorithms against that of possible optimal algorithms are not less than 2. Experimental results for average cases suggest that the First Fit and the First Fit Decreasing algorithm are better choices for the local scheduler to allocate precedence-constrained and independent non-AR jobs respectively.

1 Introduction

Grid computing affiliates the sharing of all kinds of resources. Advance Reservations(AR) make it possible to guarantee the QoS of Grid applications by reserving a particular resource capability over a defined time interval on local resources^[1]. In general, advance reservations will also cause the processing timeframe discontinuous and therefore reduce the utilization of local resources and lengthen the completion time of non-AR jobs. Up till now, many studies and experiments have been conducted on reservation-based Grid QoS technologies. Most of them have been focused on the benefits for AR jobs coming from advance reservations^[2-5]. Few of them have been devoted to evaluate the impact from advance reservations on the performances of local resources and/or normal jobs^[6-9]. However, none of them has been published to try to reduce the disadvantages on local resources and/or normal jobs resulting from advance reservations for Grid jobs.

Single machine scheduling is the basis of more complicated parallel machine scheduling. In this study, it was aimed to find a theoretic model, as well as local

scheduling algorithms, for single machine schedulers to reduce the disadvantages on utilization of local resources and to shorten the makespan(i.e., maximum completion time) of normal jobs while supporting advance reservations for Grid jobs. The local scheduler can fulfill both targets by allocating non-AR jobs into available intervals with the goal to minimize the makespan of the jobs.

For single machines, the impact of AR on the allocating of local jobs is depicted in Fig.1. Jobs are classified into two types: AR jobs and non-AR jobs. AR jobs are Grid jobs with advance reservations. Once their reservation requirements are accepted, they will be put into the AR-job queue and will be processed within reserved processing intervals. Non-AR jobs are put into the non-AR-job queue and will be processed only within those available intervals left by AR jobs. All reserved intervals for AR jobs are unavailable for non-AR jobs.

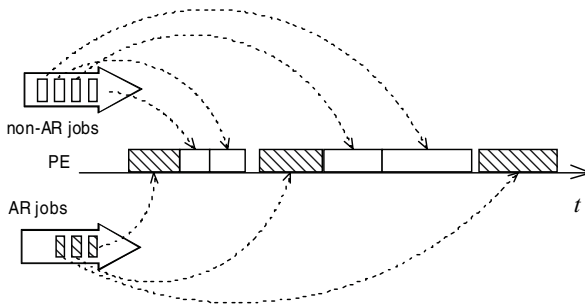


Fig. 1. The allocating of AR and non-AR jobs on single processing element

Because non-AR jobs can be processed only in discontinuous available intervals, the allocating of non-AR jobs into available intervals to minimize the makespan is a kind of availability constraint scheduling problem. With availability constraints, a machine can process jobs only in discontinuous available intervals. This restriction arises in many areas such as manufacturing industry, production scheduling and computer science [10, 11]. Suppose all non-AR jobs are non-resumable, it is NP-hard to allocate them into available intervals to minimize the makespan. When a non-resumable job cannot be finished before the ending of the available interval in which it is processed, it has to restart from the beginning later in another available interval, rather than to continue.

Assume the information (e.g., arrival time, required processing time) on queued AR and non-AR jobs is known before scheduling. If there are more than one AR jobs in queue, in order to meet the reservation requirements of possible arriving AR jobs, it is necessary to allocate non-AR jobs only into existing available intervals made by accepted reservations. Suppose the number of queued AR jobs is one and the scheduler knows the time point when a new available or unavailable interval will end as it arrives, this setting can be referred to as online.

This paper will propose a new theoretic model for the local single machine scheduler to minimize the makespan of non-AR jobs while supporting advance reservations for

Grid jobs, will present four online scheduling algorithms and evaluate their performances from both of the average case and the worst case viewpoints.

2 Problem Definition and Online Algorithms

For the online scheduling problem to allocate non-AR jobs into available intervals, the completion time of the non-AR jobs equals the length of the duration from the starting time point to the end time of the last job. Considering that the makespan comes from both of the available intervals and unavailable intervals, we can divide it into two parts: the one coming from unavailable intervals and the other coming from available intervals. For any deterministic problem instance of which the arrival time and the duration of any unavailable or available interval are fixed, the makespan is dominated by how to allocate the jobs to the available intervals. In other words, the goal to minimize the makespan is equivalent to allocate the jobs into as small number of earlier available intervals as possible. By regarding the jobs and their processing time requirements as items and item sizes respectively, and by regarding the available intervals of the machine and their durations as bins and bin sizes respectively, the deterministic online single machine scheduling problem for nonresumable non-AR jobs can be transformed into an online version of a variant of the standard Variable-Sized Bin Packing(VSBP) problem firstly proposed in [12]: given a list $L = (a_1, a_2, \dots, a_n)$ of items, each with size $s(a_j), 1 \leq j \leq n$, and a list $B = (b_1, b_2, \dots)$ of bins, each with size $s(b_i), 1 \leq i$, the goal is to pack the n items into the bins with a minimum total size of bins from b_1 to the last one being used (say $b_t, t \geq 1$) in list B . To distinguish the new problem from the standard VSBP[13], in this paper we call it the Variant of the Variable-Sized Bin Packing problem, VVSBP for short.

According to the sizes of the bins and the items, we can divide VVSBP into two subcases with feasible solutions:

1. Subcase A: $\min\{s(b_i)\} \geq \max\{s(a_j)\}$;
2. Subcase B: $\min\{s(b_i)\} \geq \min\{s(a_j)\}$ and $\max\{s(b_i)\} \geq \max\{s(a_j)\}$.

where $\min\{s()\}$ and $\max\{s()\}$ denote the minimum and the maximum in $s()$ respectively.

In the online scheduling problem, if the duration of a new available interval is not greater or equal to the maximum processing time requirement of the non-AR jobs in queue, this interval will not be considered for processing jobs. This is equivalent to subcase A. If only those available intervals the durations of which are less than the minimum processing time requirement of the non-AR jobs are ignored, we get subcase B. To guarantee feasible solutions for any problem instance of subcase A, we assume the number of bins is not less than the number of items. Because subcase A can be viewed as a subset of subcase B, to guarantee feasible solutions for any problem instance of subcase B, we can exclude those intervals with sizes less than the maximum item size in the problem instance of subcase B to construct a problem instance of

subcase A, and require the number of the bins in the derived problem instance is not less than the number of items.

In the online version of the classic one-dimensional bin packing(BP) problem or the standard VSBP problem, we have all information on the bins but we cannot preview the information of items before they arrive. An online algorithm assigns items to bins in the order of (a_1, a_2, \dots, a_n) with item $a_j, 1 \leq j \leq n$ solely on the basis of the sizes of the preceding items and the bins to which they were assigned. Without loss of generality, we can assume that no new item will arrive until its preceding items are packed into bins and no new bin will open to hold items until all the used bins cannot accept the current item. In the online version of the VVSBP problem, we have all information on items but we cannot preview the type of the bins before packing. We must decide which items should be packed into the bin as it arrives. A new bin will arrive after the current bin is closed and a closed bin will never be reopened to accept items even if it is empty.

The differences between the two kinds of online issues make those online algorithms for the BP or the VSBP problem not applicable to the VVSBP problem. It is reasonable to extend the four algorithms in [12] for both of the subcases of the online version of the VVSBP problem. Except NF, FF, NFD and FFD, we can also adapt other bin packing algorithms for the VVSBP problem, such as the Worst Fit(WF) algorithm, the Almost Worst Fit(AWF) algorithm, the Worst Fit Decreasing(WFD) algorithm and the Almost Worst Fit Decreasing(AWFD) algorithm. However, because these four algorithms are not as typical as NF, FF, NFD and FFD in practice, they will not be considered in the following. Without causing confusion, we also use the names of the algorithms in the BP problem to denote their analogues adapted for two subcases of the VVSBP problem.

1. Next Fit(NF): Pack items in the order of (a_1, a_2, \dots, a_n) , if the current bin b_i has not sufficient space to hold the current item a_j , b_i will be closed and the next bin b_{i+1} will be opened to try to hold a_j
2. First Fit(FF): All items are given in the VVSBP problem and they will be packed in the order of a_1, a_2, \dots, a_n . When packing $a_j, 1 \leq j \leq n$, put it in the lowest indexed open bin into which it will fit. If such an open bin does not exist, open a new bin until we find the least i such that $b_i, i \geq 1$ is capable of holding a_j .

Presort the items in non-increasing order by size, and then apply FF, BF to the re-ordered items, we will get the First Fit Decreasing(FFD) algorithm and the Best Fit Decreasing(BFD) algorithm respectively.

In the scheduling problem, if the jobs must be processed in index order, we say they are precedence-constrained. On the other hand, if the jobs can be processed in random order, we say they are independent. Of course, only these algorithms without presorting as mentioned above, i.e., FF, BF, WF and AWF, can be used for precedence-constrained jobs.

3 Worst Case Analysis

Competitive ratio is usually used to evaluate the performance of online algorithms, which describes the maximum deviation from optimality that can occur when a

specified algorithm is applied within a given problem class. For a list L of items, a list B of bins and an approximation algorithm H for the VVSBP problem, let $H(L, B)$ denote the total size of bins by algorithm H , and let $OPT(L, B)$ denote the total size in optimal packing, the competitive ratio of algorithm H is defined as $R_H^\infty = \limsup_{k \rightarrow \infty} \{H(L, B) / OPT(L, B) \mid OPT(L, B) \geq k\}$. For instances of subcase B in section 2, if feasible solutions do not exist, we say $R_H^\infty = \emptyset$ for any algorithms; if feasible solutions exist but algorithm H can not get a feasible solution, we say $H(L, B) = \infty$ and thus $R_H^\infty = \infty$.

Let H denote any of the online algorithms defined above, we get their competitive ratios in subcase A and subcase B as follows.

Theorem 1. For worst case instances of subcase A, $R_H^\infty = 2$.

This theorem can be proved by applying the results in [12] in that subcase A is equivalent to the essential assumption in [12] and that the four algorithms here are identical to those in [12].

Theorem 2. For worst case instances of subcase B, $R_H^\infty = \infty$.

Proof. Without lose of generality, let the list L of items with sizes $s(a_1) > s(a_2) > \dots > s(a_n), s(a_1) < s(a_2) + s(a_3)$, two instances are considered:

Instance 1: If $H \in \{NF, NFD\}$, suppose the sizes of bins are
$$\begin{cases} s(b_i) = s(a_{n-i+1}), 1 \leq i \leq n \\ s(b_i) = s(a_n), i > n \end{cases}$$
, an optimal and feasible solution is to pack a_i into $b_{n-i+1}, 1 \leq i \leq n$.

Instance 2: If $H \in \{FF, FFD\}$, suppose the sizes of bins
$$\begin{cases} s(a_2) + s(a_3) \leq s(b_1) < s(a_1) + s(a_3) \\ < s(a_1) + s(a_2) \\ s(a_1) \leq s(b_2) < s(a_2) + s(a_3) \\ s(b_i) = s(a_{i+1}), 3 \leq i \leq n - 1 \\ s(b_i) = s(a_n), i \geq n \end{cases}$$
, an optimal and feasible solution is to

pack a_1 into b_2 , a_2 and a_3 into b_1 , a_i into b_{i-1} for $4 \leq i \leq n$.

In the two instances, H cannot get feasible solutions, we have $H(L, B) = \infty$ and $OPT(L, B) = \sum_{i=1}^n s(b_i)$, and thus $R_H^\infty = \infty$.

4 Average Case Experiments

Usually, simulations with real workload trace records(e.g. Parallel Workload Archive[14]) are used to investigate the performances of parallel or Grid scheduling algorithms[8, 15-17]. However, up till now, there are not any real workload trace records for single machine schedulers. In this section, we will investigate the average-case performances of the online algorithms only from the bin packing viewpoint.

Theoretical analysis and experimental simulation are two methods to determine the average-case performance of bin packing algorithms. The usual approach for average-case analysis is to specify a density function for the problem data, including the sizes of items and bins, and then to establish probabilistic properties of an algorithm, such as the asymptotic expected ratio, the wasted space expected ratio etc.[18, 19] Among the distributions for average-case analysis, continuous uniform or discrete uniform distributions are widely used. For uniform distributions, the asymptotic expected ratio is defined as $ER_H^\infty = \lim_{n \rightarrow \infty} ER_H^n = \lim_{n \rightarrow \infty} E[H(L_{n,1})/OPT(L_{n,1})]$ and the wasted space expected ratio is defined as $EW_H^\infty = \lim_{n \rightarrow \infty} EW_H^n = \lim_{n \rightarrow \infty} E[H(L_{n,1}) - s(L_{n,1})]$, $s(L_{n,u})$ denoting the total size of n items drawn independently from the uniform distribution on the interval $[0, u]$. It is suggested in [18] that it is often easier to analysis the average case results about EW_H^n than about ER_H^n directly and the former typically imply the latter in that for most distributions that have been studied $\lim_{n \rightarrow \infty} E[OPT(L_{n,1})/s(L_{n,1})] = 1$ and $ER_{BF}^\infty = ER_{BFD}^\infty = 1$ for the classic one-dimensional bin packing problem. Moreover, when item sizes are drawn independently from the uniform distribution on the interval $[0, u]$, it is proved in [20] that $ER_u^-(H) = \lim_{n \rightarrow \infty} E[H(L_{n,u})/s(L_{n,u})]$ and $ER_u^-(FF) = 1, u \in [0, 1]$. When only consider the limitation of $u \rightarrow 1$, it is suggested in [18] that $\lim_{b \rightarrow 1} R_{FF}^\infty(U[0, b]) = 1$.

For instances of the VVSBP problem with given lists of bins and items, the goal to minimize the total size of bins is not only related to the numbers and sizes of items and bins, but also related to the sequence of bins. This makes it more difficult to analyze the average-case performance of algorithms mathematically. Instead of mathematical analysis, a numerical experiment solution is used in this section to simulate the behaviors of different algorithms for typical problem instances with sizes of items and bins drawn independently from uniform distributions.

In the following, we adapt the definition of $ER_u^\infty(H)$ for algorithms in average-case experiments as $ER_u^\infty(H) = \lim_{n \rightarrow \infty} E[H(L_{n,u}, B_{wn,u})/s(L_{n,u})]$, in which $B_{wn,u}$ denotes the list B of wn bins with sizes drawn from the uniform distribution in the interval $[u^*, 1]$. In experiments, for problem instances in subcase A, $u^* = u$ and $w = 1$; For subcase B $u^* = \min\{L_{n,u}\}$ and $w = 10$.

Let us first examine the situation when n is small. Fig. 2(a) and Fig. 2(b) depict the average values of $H(L_{n,u}, B_{n,u})/s(L_{n,u})$ of algorithms when applying them to problem instances in subcase A and subcase B respectively with $n = 20$ and 100 . u ranges from 0.05 to 1 in steps of 0.05 and each data point represents the average of 100 runs. It can be seen from Fig.2(a) and Fig.2(b) that FF and FFD are always better than NF and NFD. As n increases, the improvement of NF and NFD are not as remarkable as that of FF and FFD. Surprisingly, in Fig.2(a), when $u > 0.7$, the performance curves of NF and NFD with $n = 100$ become even worse than that with $n = 20$.

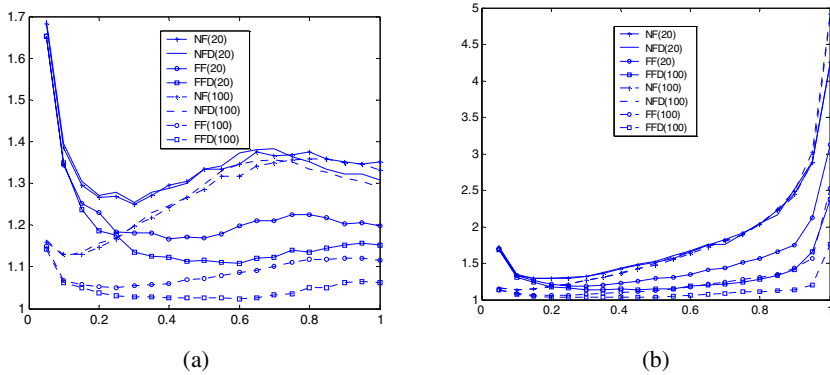


Fig. 2. $ER_u^{\infty}(H)$ for subcase A(a) and B(b) as a function of u with $n=20$ and 100

To be able to obtain the asymptotical average performances of algorithms, long lists of items are preferred. Experiments with $n = 500, 1000, 2000, 5000, 10000$ and 20000 are carried out to examine the asymptotical performances. Fig.3(a) and Fig.3(b) depict the experimental results with $n = 500$ and 2000 for subcase A and B respectively.

In experiments, with the same value of u , the improvements of the average ratios of NF and NFD coming from the increasing of n are very small: as n increases from 500 to 2000, in subcase B, the improvements of the average ratios of NF for $u = 0.05$ through 1 are within $[-0.8405, 0.0202]$, and the improvements of NFD are within $[-0.9870, 0.0184]$; In subcase A, the improvements of NF are within $[-0.0005, 0.0187]$, and the improvements of NFD are within $[-0.0022, 0.0168]$.

Moreover, by $n = 10000$, both the NF and the NFD algorithm appear to have converged, and both of them seem to be very near their asymptotes: in subcase A, as n increases from 10000 to 20000, the improvements of NF for $u = 0.05$ through 1 are within $[-0.0013, 0.0017]$, and the improvements of NFD are within $[-0.0005, 0.0019]$. Similar convergence happens in subcase B. Due to the convergence properties of NF and NFD in subcase A, it is shown that the results of NF or NFD with different n are overlapped in most cases. Thus in Fig.3, for simplicity, only the results with $n = 2000$ were given out.

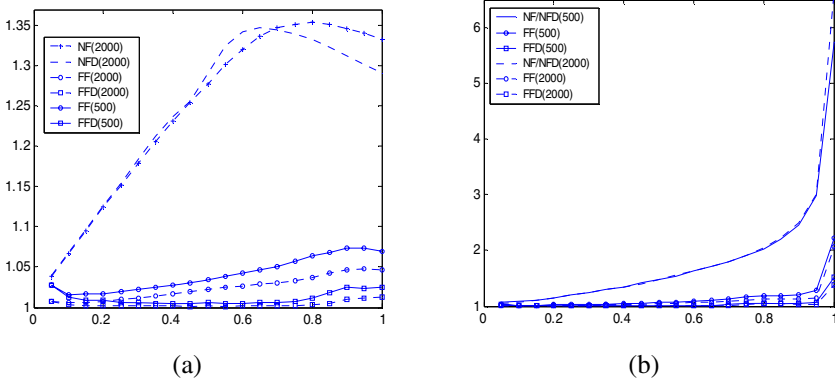


Fig. 3. $ER_u^m(H)$ for subcase A(a) and B(b) as a function of u with $n=500$ and 2000

5 Conclusions

Although many researches have been reported to investigate the impact on the utilization of local resources, on AR jobs or on non-AR jobs resulting from advance reservations for Grid jobs, none of them has been devoted to try to reduce the disadvantages on local resources and/or normal jobs. This study was aimed to find a theoretic model, as well as local scheduling algorithms, for single machine schedulers to reduce the disadvantages on utilization of local resources and to shorten the makespan(i.e., maximum completion time) of normal jobs while supporting advance reservations for Grid jobs. The performances of the algorithms were investigated from both of the worst case and the average case viewpoints. Analytical results show that the worst case performance ratios of the algorithms against that of possible optimal algorithms are not less than 2. Experimental results for average cases show FF and FFD are better than NF and NFD. Considering that when applying NFD to the non-AR jobs, it's required that the processing order of the jobs can be changed without interfering the execution of the jobs, NFD can be applied only for independent non-AR jobs. Because it is not necessary to change the processing order of the jobs when applying FF, this algorithm can be applied for both of independent of precedence-constrained non-AR jobs. For independent non-AR jobs, the average performance of FFD is better than that of FF.

Acknowledgements

This research was supported by the Natural Science Foundation of China with grant number 60663009, the Application Science and Technology Foundation of Yunnan Province with grant number 2006F0011Q, the Research Foundation of the Education Department of Yunnan Province with grant number 6Y0013D and the Research Foundation of Yunnan University with grant number 2005Q106C.

References

1. MacLaren, J.: Advance Reservations: State of the Art. Global Grid Forum (2003)
2. Noro, M., Baba, K., Shimojo, S.: QoS control method to reduce resource reservation failure in datagrid applications, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM) (IEEE Cat. No. 05CH37690) (2005) pp. 478–481 (2005)
3. Mohamed, H.H., Epema, D.H.J.: The design and implementation of the KOALA co-allocating grid scheduler. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 640–650. Springer, Heidelberg (2005)
4. Sulistio, A., Buyya, R.: A Grid Simulation Infrastructure Supporting Advance Reservation. In: Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems. Anaheim, pp. 1–7. ACTA Press, Cambridge, Boston (2004)
5. Mateescu, G.: Quality of service on the grid via metascheduling with resource co-scheduling and co-reservation. International Journal of High Performance Computing Applications 17, 209–218 (2003)
6. McGough, A.S., Afzal, A., Darlington, J., Furmento, N., Mayer, A., Young, L.: Making the grid predictable through reservations and performance modelling. Computer Journal 48, 358–368 (2005)
7. Smith, W., Foster, I., Taylor, V.: Scheduling with advanced reservations. In: Proceedings of 14th International Parallel and Distributed Processing Symposium, pp. 127–132. IEEE Computer Society Press, Los Alamitos (2000)
8. Heine, F., Hovestadt, M., Kao, O., Streit, A.: On the impact of reservations from the Grid on planning-based resource management. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J. (eds.) ICCS 2005. LNCS, vol. 3514, pp. 155–162. Springer, Heidelberg (2005)
9. Snell, Q., Clement, M., Jackson, D., Gregory, C.: The Performance Impact of Advance Reservation Meta-Scheduling. In: Feitelson, D.G., Rudolph, L. (eds.) IPDPS-WS 2000 and JSSPP 2000. LNCS, vol. 1911, pp. 137–153. Springer, Heidelberg (2000)
10. Sanlaville, E., Schmidt, G.: Machine scheduling with availability constraints. Acta Informatica 35, 795–811 (1998)
11. Schmidt, G.: Scheduling with limited machine availability. European Journal of Operational Research 121, 1–15 (2000)
12. Zhang, G.: A new version of on-line variable-sized bin packing. Discrete Applied Mathematics 72, 193–197 (1997)
13. Friesen, D.K., Langston, M.A.: Variable sized bin packing. SIAM journal on computing 15, 222–230 (1986)
14. Parallel Workloads Archive, <http://www.cs.huji.ac.il/labs/parallel/workload/>
15. Edi, S., G, F.D.: Backfilling with lookahead to optimize the packing of parallel jobs. Journal of Parallel and Distributed Computing 65, 1090–1107 (2005)
16. Chiang, S.-H., Fu, C.: Re-evaluating Reservation Policies for Backfill Scheduling on Parallel Systems. In: Proceedings of the 16th IASTED International Conference on Parallel and Distributed Computing and Systems, Cambridge, MA, pp. 455–460 (2004)
17. Weng, C., Li, M., Lu, X.: An Online Scheduling Algorithm for Assigning Jobs in the Computational Grid. IEICE Trans Inf Syst E89-D, 597–604 (2006)
18. Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Approximation Algorithms for Bin Packing: A Survey. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-Hard Problems, pp. 46–93. PWS Publishing, Boston (1996)

19. Coffman Jr., E.G., Galambos, G., Martello, S., Vigo, D.: Bin Packing Approximation Algorithms: Combinatorial Analysis. In: Du, D.-Z., Pardalos, P.M. (eds.) *Handbook of Combinatorial Optimization*, pp. 1–47. Kluwer Academic Publishers, Dordrecht (1998)
20. Csirik, J., Johnson, D.S.: Bounded Space On-Line Bin Packing: Best is Better than First. *Algorithmica* 31, 115–138 (2001)