Computers & Operations Research 40 (2013) 1034–1037

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Off-line scheduling with forbidden zones

Amir Abdekhodaee [a], Andrew Wirth [b,*]

[a] Faculty of Engineering and Industrial Sciences, Swinburne University of Technology, VIC 3122, Australia
[b] Department of Mechanical Engineering, The University of Melbourne, VIC 3010, Australia

ARTICLE INFO

Available online 6 November 2012

Keywords:
Scheduling
Bin packing
Tidal constraints

ABSTRACT

In various manufacturing and computing environments there may be certain time intervals, during which processing may continue but may not be initiated. We examine the problem of off-line scheduling in the presence of such forbidden zones. The problem is closely related to a one-dimensional open-end bin packing problem. We prove that the decision version of the problem is strongly NP-complete and then establish bounds on the asymptotic performance ratio of an $O(n \log n)$ approximation algorithm for a special case, and test it numerically. We present a further heuristic for a non-regular case and test it empirically.

1. Introduction

The problem we address in this paper is a variant of the one-dimensional bin packing problem. Its formulation was motivated by our earlier research [1] on the management of a coal supply chain. That work involved considering subproblems such as mine management, rail scheduling, stockyard management, and ship loading scheduling. In particular, the scheduling of ship berthing provides a challenge because of tidal constraints. That is, during certain time periods, it is unsafe for ships to move in or out of a berth, as the depth of water is not within required ranges that have been set by marine regulations. We call these time periods *forbidden zones*. The time periods between two successive forbidden zones are called the *allowed zones*. The loading (or processing) of ships, however, is not restricted by tidal periods and can be started once the ship has berthed. Moreover, ship loading times are different depending on the size of the ship and the cargo to be loaded. The aim is to sequence the ships so that a high level of berth utilization can be achieved. This type of problem arises in other contexts as well. For example, suppose a particular resource, such as a supervisor, is required to start and to complete a job without being involved in the job's processing. If supervision is only available in certain periods, then the allocation of jobs becomes similar to scheduling with forbidden zones.

In the following discussion we restrict ourselves to the consideration of a single resource, for example we assume that a berth can accommodate only one ship at a time. Nevertheless, our solution methods can be extended to handle multiple resources.

2. Problem definition and notation

We follow the notation of [1,2], which contain discussions of the off- and on-line versions of this problem, respectively. Let n be the number of non-preemptable jobs to be processed on a single machine and $\{p_1,p_2,\ldots,p_n\}$ be their processing times. We partition time into a set of equal size *intervals* $\{I_1,I_2,\ldots,I_m\}$ which also include nonzero portions with durations F_j, $j=1,\ldots,m$ where $0<|F_j|<|I_j|\ \forall j$. The intervals defined by the F_j's are called *forbidden zones* and without loss of generality they are always at the end of the corresponding I_j's. Forbidden zones represent time intervals during which jobs cannot be started or released if completed; however, they can be processed and if completed prior to an interval's end, they will be released at the start of the next interval. We call a job *delayed* if it finishes within a forbidden zone. We say that $I_j \backslash F_j$ (that is, the set of elements in I_j but not in F_j) is the jth *allowed zone*. It seems reasonable to assume that $|F_j|<|I_j|$, $|F_j|>0$ and $|I_j|>0\ \forall j$, since empty forbidden and empty allowed zones can be ignored (Fig. 1).

The objective is to sequence the jobs so that the number of intervals used is minimized. We call this the *forbidden zone* (*FZ*) problem. (If the last forbidden zone contains a job, then this is equivalent to minimizing the maximum completion time or makespan.)

In the next section we study the *regular* case where all allowed and forbidden zones are the same length, this length being also an upper bound to the job length. We show that even this restricted problem is strongly NP-complete, construct an $O(n \log n)$ approximation algorithm for the regular case and prove a result about its asymptotic worst case performance. We then consider the general case and present some empirical results for various heuristics. We conclude with some possible future research problems.

* Corresponding author. Tel.: +61 3 8344 4852; fax: +61 3 9347 8784.
E-mail address: wirtha@unimelb.edu.au (A. Wirth).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.10.020

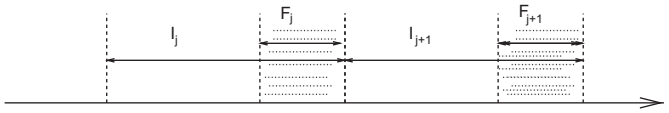

Fig. 1. Intervals and forbidden zones.

3. The regular case

We begin by studying the special case where all allowed and forbidden zones are of equal length, say 1, and where $p_i \le 1\ \forall i$. We call this the *regular* case.

Thus $I_1=[0,2], I_2=[2,4], \ldots, I_m=[2m-2,2m]$. Each I_j includes a corresponding *forbidden zone* $F_j \subset I_j\ \forall j$, where $F_1=(1,2], F_2=(3,4], \ldots, F_m=(2m-1,2m]$. Below, when we refer to *intervals* we shall mean the sets I_j. Thus, the latest time a job may commence in I_j is at $t=2j-1$.

Here, as in [1,2], we assume that we may place a job in the forbidden zone even if there is room for it in an allowed zone. By way of contrast, the assumptions of the model in [3] are equivalent to only allowing the placement of a job in a forbidden zone if it cannot be accommodated in its corresponding allowed zone, and the corresponding allowed zone is not full. We also note that [2,3] deal with the regular case only. The difference in initial assumptions leads to substantially different results for the two scenarios in an on-line setting. For example, for the on-line version of the problem Leung et al. [3] show that no algorithm has an asymptotic performance ratio less than 2 and that *next fit* (or *list scheduling*) is optimal. On the other hand, for the Khammuang et al. [2] version of the problem the situation is more involved. For example, there exist on-line heuristics with asymptotic performance ratios of less than 2. An integer programming formulation of the regular problem appeared earlier in [2].

3.1. Complexity

Since our problem is a variant of the bin packing problem we may expect it likewise to be strongly NP-complete. We adapt slightly the complexity proof of [3] to show that this is, indeed, the case.

Proposition 1. *The decision version of the regular FZ problem is strongly NP-complete.*

Proof. It is clear that *FZ* is in NP. We now reduce the 3-partition problem to the regular FZ problem, to complete the proof. We first recall the partition problem. Given a list $A=(a_1,a_2,\ldots,a_{3m})$ of $3m$ integers such that $\sum_{i=1}^{3m} a_i = mB$ and for each i, $1 \le i \le 3m$, $\frac{1}{4}B < a_i < \frac{1}{2}B$, can A be partitioned into $A_1, A_2, \ldots, A_m$ such that for each j, $1 \le j \le m$, $\sum_{a_i \epsilon A_j} a_i = B$?

We let $p_i = a_i/B$ for $1 \le i \le 3m$, be the *short* jobs and $p_i=1$ for $3m < i \le 4m$, be the *long* jobs. If a 3-partition exists then it is clear we can schedule the jobs in m or fewer intervals, placing job p_{3m+j} in the jth forbidden zone.

Conversely if the jobs in P can be scheduled in m or fewer intervals then each interval can contain at most 3 short and 1 long job, or 2 long jobs, or 4 short jobs. Let the numbers of such intervals be α, β and γ, respectively. Then $3\alpha+4\gamma \ge 3m$ and $\alpha+2\beta \ge m \ge \alpha+\beta+\gamma$. Hence $\alpha=m$, $\beta=\gamma=0$. So A can be partitioned into $A_1, A_2, \ldots, A_m$ such that for each j, $1 \le j \le m$, $\sum_{a_i \epsilon A_j} a_i \le B$. Hence a 3-partition exists. □

3.2. Heuristics

We first recall the definition of a c-competitive algorithm. A minimization algorithm A is said to be *c-competitive* if $A(I)/OPT(I) \le c$ for all job instances I, where $A(I)$ and $OPT(I)$ are the objective function values for algorithm A and the optimal solution, respectively. The *performance ratio* $R_A(\alpha)$ is defined as follows. Suppose that $p_{max}=\alpha$, where $0<\alpha\le 1$, then $R_A(\alpha) = \inf\{r \ge 1 : A(I)/OPT(I) \le r,\ \text{for all } I\}$. We abbreviate $R_A(1)$ to R_A. The corresponding *asymptotic performance ratio* $R_A^\infty(\alpha)$ is defined as follows: $R_A^\infty(\alpha) = \inf\{r \ge 1$: for some $N>0, A(I)/OPT(I) \le r$, for all I with $OPT(I) \ge N\}$.

We also recall the simplest heuristic of all, namely *next fit* (*NF*) or *list scheduling* (*LS*). This heuristic places the next job in the list in the current interval if it fits, otherwise it closes that interval and places the item at the beginning of the available zone in the next interval.

Proposition 2. $R_{NF}^\infty = 2$.

Proof. The proof is similar to that, for *NF* for bin packing and, in Theorem 2 of [3] and Proposition 4 of [2], for the online version of our problem. Consider a sequence of $2n$ jobs of lengths: $1, \epsilon, 1, \epsilon, \ldots$. *NF* requires n intervals whereas the optimal solution requires only $n/2+1$ for sufficiently small ϵ. Thus $R_{NF}^\infty \ge 2$. On the other hand, for any *NF* schedule, each allowed zone, except possibly the last one, is full, so $R_{NF}^\infty \le 2$. □

Leung et al. [3] construct an adaptation of the Karmarkar and Karp algorithm for bin packing that results in a fully polynomial approximation scheme (with respect to the asymptotic worst-case ratio) for their problem. However, as Coffman et al. [4] state, this comes at the price of running time issues. Leung et al. also comment that their adaption is suitable for any other approximation algorithm such the standard and fast *first fit decreasing* (*FFD*) heuristic, but do not study its performance.

We adapt their algorithm to construct a version of *FFD* appropriate to our problem. Our heuristic is based on the observation [2, Proposition 1] that there exists an optimal solution to *FZ* for which the longest jobs are all in the forbidden zones. A simple interchange proof suffices to demonstrate this result.

We recall that for the bin packing problem *FFD* first sorts the items in order of non-increasing size and then allocates each item to the first bin able to accommodate it.

Algorithm zone first fit decreasing (ZFFD) sort the jobs in order of non-increasing processing times. Relabel the jobs so that we now have, $p_1 \ge p_2 \ge \cdots \ge p_n$. Place job i at the beginning of F_i for $i=1,\ldots,\lceil n/2 \rceil$. Use *FFD* to bin pack all the remaining jobs into the first a, say, allowed zones. If $a < \lceil n/2 \rceil$ then remove the $\lceil (\lceil n/2 \rceil - a)/2 \rceil$ smallest jobs in the forbidden zones from their forbidden zones and add them to the jobs currently in the allowed zones. Repack *all* the jobs allocated to the allowed zones, using *FFD*. Repeat this process until the number of used allowed zones equals or exceeds the number of used forbidden zones. Let γ (or δ) denote the final number of used allowed (or forbidden) zones. Then $\gamma \ge \delta \ge \gamma - 1$.

Proposition 3. $\frac{284}{251} \le R_{ZFFD}^\infty \le \frac{11}{9}$.

Proof. If we consider the jobs that are allocated to the allowed zones at the end of the *ZFFD* process it is clear that, asymptotically, they cannot be packed into fewer than $9\gamma/11$ allowed zones since, for bin packing, $R_{FFD}^\infty = \frac{11}{9}$, Coffman et al. [4]. And we recall that we may assume that in the optimal solution the longest jobs are in the forbidden zones. So, in fact, the optimal *FZ* solution would also need to accommodate the $2\gamma/11$ shortest jobs allocated to the forbidden zones by the *ZFFD* solution.