Algorithmica (2012) 63:781–794
DOI 10.1007/s00453-011-9502-9

Caching Is Hard—Even in the Fault Model

Marek Chrobak · Gerhard J. Woeginger · Kazuhisa Makino · Haifeng Xu

Received: 24 September 2010 / Accepted: 17 February 2011 / Published online: 10 March 2011

Abstract We prove strong $\mathbb{NP}$-completeness for the four variants of caching with multi-size pages. These four variants are obtained by choosing either the fault cost or the bit cost model, and by combining it with either a forced or an optional caching policy. This resolves two questions in the area of paging and caching that were open since the 1990s.

1 Introduction

The *Caching Problem* deals with page replacement policies in two-level memory systems consisting of a small, fast cache and a large but slow main memory. This is a classical and well-studied problem in the area of on-line algorithms (see, for example, [7]), but in this paper we will be solely interested in its off-line version.

Formally, a caching instance specifies a sequence R of requests for memory pages. The pages in R are requested one by one, and for each page p we are given its size $\text{SIZE}(p)$ and its faulting cost $\text{COST}(p)$. The cache, whose size C is also specified in the instance, can store a subset of memory pages whose total size does not exceed C. When the requested page p is in the cache, the request is served at no cost. When the

M. Chrobak (✉)
Department of Computer Science, University of California, Riverside, Riverside, USA
e-mail: marek@cs.ucr.edu

G.J. Woeginger
Department of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands

K. Makino
Department of Mathematical Informatics, Graduate School of Information and Technology, University of Tokyo, Tokyo, Japan

H. Xu
Department of Mathematics, Zhejiang University, Hangzhou, China

requested page p is not in the cache, a page fault of cost $\text{COST}(p)$ occurs. In response to a fault, p may be fetched into the cache. (Without loss of generality, we assume that pages are fetched only in response to faults.) In order to make room for p, other pages may have to be evicted from the cache. The objective is to decide which pages one should retain in the cache at each step so as to minimize the overall page fault cost.

There are two basic policies that determine how a page fault is resolved:

Forced: The faulted page p must be loaded and stored in the cache, where it occupies $\text{SIZE}(p)$ bits.

Optional: The faulted page p can either be loaded for later use into the cache (where it occupies $\text{SIZE}(p)$ bits), or it can be left outside the cache. In the latter case, the next request to p will necessarily cause a fault.

We stress that the forced policy is the standard in the literature, and all results mentioned later-on in this section assume the forced policy. The optional policy was introduced by Irani [13] in the context of web caching.

The literature contains four fundamental models of caching, defined by imposing different assumptions on page sizes and fault costs (this classification can be found in the work of Albers, Arora and Khanna [1]).

Bit model: For each page p we have $\text{COST}(p) = \text{SIZE}(p)$. The fault cost is proportional to the time it takes to bring the page into the cache. This model goes back to Irani [13].

Cost model: For each page p we have $\text{SIZE}(p) = 1$. (This model is also known as the *weighted caching* problem.) All pages have more or less the same size, but they may have varying fault costs. This model goes (at least) back to Chrobak, Karloff, Payne and Vishwanathan [10].

Fault model: For each page p we have $\text{COST}(p) = 1$. The setup cost for a fault is huge, and hence the exact page sizes have no real influence on the fault cost. This model was introduced by Irani [13].

General model: For each page cost and size can be arbitrary. This model was introduced by Young [14].

What positive results are known about off-line caching? The simplest variant combines the properties of bit, cost, and fault model, and only considers pages of unit size and unit cost; it can be solved to optimality by Belady's rule [6]: "Always evict the cached page whose next request is furthest in the future". Caching in the cost model can be solved in polynomial time using network flow methods; see Chrobak et al. [10]. Albers et al. [1] derived the first off-line approximation results for the bit, fault, and general model. The strongest currently known approximation result for the general model is a polynomial time 4-approximation algorithm by Bar-Noy et al. [5].

What negative results are known about off-line caching? In 1997, Fiat [11] constructed a reduction from the PARTITION problem to caching in the bit model, which implies weak $\mathbb{NP}$-completeness for the bit model as well as for the general model. In 1999 Albers et al. [1] wrote in their concluding remark: *"The hardness results for caching problems are very inconclusive. The $\mathbb{NP}$-hardness result for the bit model uses a reduction from* PARTITION*, which has pseudo-polynomial algorithms. Thus a*

similar algorithm may well exist for the bit model. We do not know whether computing the optimum in the fault model is $\mathbb{NP}$*-hard."* In fact, this quote provides an exact summary of the current state of knowledge (just before our paper), and the open questions about the complexity of these problems have been formulated repeatedly in the caching literature since 1999. The only other relevant work we are aware of is that of Brehob et al. [8], who proved $\mathbb{NP}$-hardness of caching in non-standard cache architectures.

Contribution of this Paper. We establish that caching in the fault model and caching in the bit model are strongly $\mathbb{NP}$-complete, under the forced as well as under the optional policy. Note that for the bit model, our result excludes the possibility of a pseudo-polynomial algorithm (unless $\mathbb{P} = \mathbb{NP}$). These results finally settle the complexity status of all the caching variants discussed above.

As an intermediate step in our construction, we also show $\mathbb{NP}$-hardness of the problem that we call *interval packing*, where the objective is to choose a maximum-weight subset of a collection of weighted intervals, without exceeding a given bound on its cuts (see the next section for a formal definition). The interval packing problem is a special case of the unsplittable flow problem on line graphs that has been well studied in the literature [2, 4], due to its close connections to a variety of optimization problems arising in scheduling and resource allocation. Thus our work contributes to better understanding of the complexity of these other problems as well.

The paper is organized as follows. In Sect. 2 we introduce two interval packing problems that play the central role in the paper and we show that strong $\mathbb{NP}$-completeness of interval packing implies strong $\mathbb{NP}$-completeness of caching. The rest of the paper contains our main technical contribution: Sect. 3 discusses the used gadgets and how they interact, Sect. 4 proves intractability of an intermediate auxiliary problem, and Sects. 5 and 6 finally contain the hardness proofs for interval packing.

Other Related Work. In the literature, the caching problem for multi-size pages is often called *file caching* or *web caching*. Most of the work on file caching is concerned with its online version, where requests to pages arrive over time and the algorithm needs to respond to each request before the next one arrives. In this online version, the focus is on designing algorithms with low competitive ratios. Young [14] and, independently, Cao and Irani [9] gave a C-competitive deterministic algorithm for file caching with an arbitrary cost function (the general model), matching the lower bound. Recently, an $O(\log^2 C)$-competitive randomized algorithm for this problem was given by Bansal, Buchbinder and Naor [3], with the ratio improved to the asymptotically optimal bound $O(\log n)$ for the bit and fault models. More information about online file caching and references to other results on this problem can be found in the above-mentioned papers.

2 Caching Versus Interval Packing

This section introduces an auxiliary weighted interval packing problem where we wish to choose a maximum number of given weighted intervals, subject to the con-

straint that for any point the total weight of the covering intervals does not exceed a given threshold.

We now give a more formal definition. Suppose we are given a set of N intervals (s_i, t_i), $i = 0, \ldots, N-1$. We will identify these intervals by their indices, that is "interval i" will refer to (s_i, t_i). For a subset $S \subseteq \{0, 1, \ldots, N-1\}$ of intervals, we define its weight in the natural way as $w(S) = \sum_{i \in S} w_i$. Also, for any real number γ, we define $cut_\gamma(S) = \{i : s_i < \gamma < t_i\}$ to be the so-called *cut* of S at γ, that is, the set of intervals which contain γ.

The weighted interval packing problem is to choose a maximum-cardinality subset of intervals that satisfies $w(cut_\gamma(S)) \leq W$ for all γ. The decision version of this problem, denoted IntvPack-Card, is formulated as follows.

Problem: IntvPack-Card

Instance: A set of N open intervals (s_i, t_i) for $i = 0, \ldots, N-1$, where each interval i has weight $w_i \geq 0$. Positive integers W and ℓ.

Question: Is there a subset S of ℓ intervals that satisfies $w(cut_\gamma(S)) \leq W$ for all real numbers γ?

In the following variation of IntvPack-Card, the objective changes from finding a subset of large cardinality to finding a subset of large weight.

Problem: IntvPack-Weight

Instance: A set of N open intervals (s_i, t_i) for $i = 0, \ldots, N-1$, where each interval i has weight $w_i \geq 0$. Positive integers W and L.

Question: Is there a subset S of the intervals with $w(S) \geq L$ that satisfies $w(cut_\gamma(S)) \leq W$ for all real numbers γ?

We will prove in Theorems 4 and 5 that both decision problems IntvPack-Card and IntvPack-Weight are strongly $\mathbb{NP}$-complete.

Now let us draw the connection between interval packing and caching problems. Here is a generic decision version of the caching variants that will be proved to be intractable:

Problem: Caching

Instance: A set of pages $p_1, \ldots, p_k$ with sizes $\text{size}(p_1), \ldots, \text{size}(p_k)$. A request sequence $r_1, \ldots, r_m \in \{p_1, \ldots, p_k\}$. A cache size C, and a cost bound F.

Question: Is there a replacement policy that serves $r_1, \ldots, r_m$ with a cache of size C and incurs a total fault cost at most F?

The four caching variants that arise from combining the generic decision problem with the fault/bit model under an optional/forced caching policy are respectively denoted as Caching(fault,optional), Caching(fault,forced), Caching(bit, optional), and Caching(bit,forced).

2.1 Hardness for Optional Policies

Our first reduction is from IntvPack-Card to Caching(fault,optional). In a preprocessing step we perturb the IntvPack-Card instance such that the endpoints of the N intervals become pairwise distinct and coincide with the integer

points $1, 2, \ldots, 2N$. This can be done while preserving the intersection patterns of the intervals.

Now let us construct an instance of CACHING(FAULT,OPTIONAL). For every interval i we introduce a corresponding page p_i with $\text{SIZE}(p_i) = w_i$. The request sequence R consists of $2N$ requests. Every page p_i is requested exactly twice, once at position s_i and once at position t_i of the request sequence. The cache size is $C = W$, and the bound on the number of page faults is $F = 2N - \ell$. We need to show that the original instance of INTVPACK-CARD has a solution if and only if the instance of CACHING(FAULT,OPTIONAL) that we just constructed has a solution.

($\Rightarrow$) Suppose the INTVPACK-CARD instance has a solution set S. Then while serving the page requests, we only load pages p_i with $i \in S$ into the cache, and we evict them right away after they have been requested for the second time. The cut condition guarantees that at every point in time the cache can accommodate all loaded pages. Since every page p_i with $i \in S$ faults once and every page p_i with $i \notin S$ faults twice, this yields a total of at most $2N - \ell$ page faults.

($\Leftarrow$) Next, suppose that the caching instance has a solution with at most $F = 2N - \ell$ page faults. Every page p_i must fault when it is requested the first time at s_i. Let S contain all intervals i for which p_i does not fault when it is requested the second time; this implies $|S| \geq \ell$. Since the pages p_i with $i \in S$ occupy space in the cache from request s_i till request t_i, the cache size W ensures that all cuts have weight bounded by W.

All in all, this yields that the INTVPACK-CARD instance has a solution if and only if the CACHING(FAULT,OPTIONAL) instance has a solution. In an almost identical fashion, we can reduce INTVPACK-WEIGHT to CACHING(BIT,OPTIONAL). The only difference is that this time we define the bound on the total fault cost as $F = 2\sum_{i=0}^{N-1} w_i - L$. All remaining arguments go through as before. With Theorems 4 and 5, this yields the following.

Theorem 1 *Decision problems* CACHING(FAULT,OPTIONAL) *and* CACHING(BIT, OPTIONAL) *are strongly* $\mathbb{NP}$*-complete.*

2.2 Hardness for Forced Policies

Our next reduction will be from CACHING(FAULT,OPTIONAL) to CACHING(FAULT, FORCED), and it is very simple. Take an instance of CACHING(FAULT,OPTIONAL), keep all the old pages, and create two new pages p^* and p^{**} with $\text{SIZE}(p^*) = \text{SIZE}(p^{**}) = C + 1$. The new cache size is $C^f = 2C + 1$. The new request sequence R^f has length $3m$, and it results from the old request sequence R by replacing every request r_j by the three consecutive requests r_j, p^*, p^{**}. The new bound on the number of page faults is set to $F^f = F + 2m$. Then R^f, C^f, F^f specify an instance of CACHING(FAULT,FORCED).

We need to show that one instance has a solution if and only if the other one has one. The idea of the proof is to use the extra space of size $C + 1$ for the requested pages that were not loaded in the optional service for R, while the requests to p^* and p^{**} are added so that the forced service cannot take advantage of having these pages in the cache. A formal argument follows.

($\Rightarrow$) Suppose the CACHING(FAULT,OPTIONAL) instance has a solution with cost F. In the cache of size $C^f = 2C + 1$, we reserve a segment of length C for handling the old pages. We serve request sequence R^f by mimicking the serving of sequence R: Whenever the policy for R loads an old page into the cache, we load the same old page into the reserved segment of the cache. The unreserved segment of length $C + 1$ is used for loading the other old pages (which the policy for R does not load) and the new pages p^* and p^{**}. Then we only incur $2m$ additional faults for the $2m$ requests to p^* and p^{**}, and sequence R^f is served at a cost of $F + 2m$.

($\Leftarrow$) Next suppose that the CACHING(FAULT,FORCED) instance has a solution with cost $F + 2m$. Since the pages p^* and p^{**} do not fit simultaneously into the cache, this solution must fault on every request to p^* and p^{**}. The old pages are served at a total fault cost of at most F, and this induces a solution under the optional policy of cost at most F.

That completes the proof for the fault model. In the bit model, our reduction from CACHING(BIT,OPTIONAL) to CACHING(BIT,FORCED) is similar: Create a new instance with the same page set, R^f and C^f as before. However, the new bound on the cost of page faults this time is set to $F + 2m\,(C + 1)$. Other than this, the proof remains essentially the same.

Theorem 2 *The decision problems* CACHING(FAULT,FORCED) *and* CACHING(BIT, FORCED) *are strongly* $\mathbb{NP}$*-complete.*

3 Setting up the $\mathbb{NP}$-completeness Proof

The hardness proof for INTVPACK-CARD is by reduction from the well-known $\mathbb{NP}$-complete VERTEXCOVER problem; see Garey and Johnson [12]. An instance of VERTEXCOVER consists of an undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, and an integer k, $0 \le k \le n$. The objective is to determine if G has a vertex cover of cardinality k.

We will present a reduction that maps an instance G, k of VERTEXCOVER into a corresponding instance of INTVPACK-CARD. Our construction can be viewed as consisting of two somewhat independent gadgets: one gadget is responsible for choosing a k-element vertex set—a candidate cover of G, while the other one verifies whether this chosen set is indeed a correct cover.

We describe the reduction in several stages. In this section we introduce the main ideas behind the cover-choosing gadget. Sect. 4 gives a construction for a variant of interval packing with more complicated constraints on cut weights. Finally, in Sects. 5 and 6 we will show how to wrap-up this construction and derive hardness of INTVPACK-CARD and INTVPACK-WEIGHT.

The Set Dominance Relation. For two sets $X, Y \subseteq \{0, \ldots, n-1\}$ such that $|X| = |Y| = k$, we write $X \preceq Y$ if there is a 1-1 mapping (matching) $f : X \to Y$ such that $f(x) \ge x$ for all $x \in X$. We will also say that Y *dominates* X. We will write $X \prec Y$ if $X \preceq Y$ and $X \ne Y$. It is easy to show (and well-known) that Y dominates X if and only if $|X|_{\le x} \ge |Y|_{\le x}$ for all x, where $|Z|_{\le x} = |\{z \in Z : z \le x\}|$. The dominance relation is clearly transitive. Further, it satisfies the following important property:

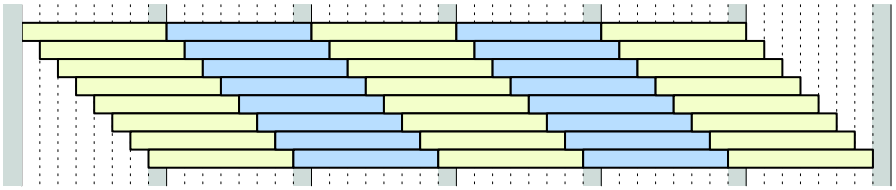

Fig. 1 A cover chooser for $n = 8$. The picture shows only a portion of the instance, with bundles shaded alternately light and dark. Central slots are *shaded*

Lemma 1 *Suppose that* $Z_0 \prec Z_1 \prec \cdots \prec Z_r$. *Then* $r \le k(n-k)$.

Proof Let $\phi_i = \sum_{z \in Z_i} z$. We have $\phi_0 \ge \binom{k}{2}$, $\phi_r \le \binom{n}{2} - \binom{n-k}{2}$, so $\phi_r - \phi_0 \le k(n-k)$. Since $\phi_{i+1} > \phi_i$ for all i, the lemma follows. □

Cover Chooser. Let $P = k(n-k)+1$ and $B = mP+1$. We now consider the instance of IntvPack-Card with bounds $W = k$ and $\ell = kB$, and with $N = nB$ intervals, each of length n: $(s_{b,z}, t_{b,z}) = (bn+z, bn+z+n)$, for $b = 0, \ldots, B-1$ and $z = 0, \ldots, n-1$. All intervals have weight 1. (See Fig. 1 for illustration.) The intervals are grouped into B *bundles* and all bundles, except for the last one, are grouped into P *phases*, as follows:

- Bundle b, $0 \le b \le B-1$, consists of the intervals $(s_{b,z}, t_{b,z})$, $z = 0, \ldots, n-1$.
- Phase p, $0 \le p \le P-1$, consists of the m bundles numbered $pm, pm+1, \ldots, pm+m-1$.

The last bundle $B-1$ does not belong to any phase.

For an integer σ, $0 \le \sigma \le nB+n-1$, the unit interval $(\sigma, \sigma+1)$ is called a *slot*. For each bundle b, the slot $(\lambda_b, \lambda'_b) = (s_{b,n-1}, t_{b,0}) = (bn+n-1, bn+n)$ is called the *central slot* of this bundle.

Consider some solution S of IntvPack-Card. Since all intervals in bundle b overlap its central slot, S contains at most k intervals from each bundle. On the other hand, S contains at least $\ell = kB$ intervals, so it must contain *exactly* k intervals from each bundle. Denote by S_b the set of k intervals from bundle b that are in S. We will identify the intervals in S_b by their index with respect to the bundle, that is S_b contains those z for which $(s_{b,z}, t_{b,z})$ is in S.

Lemma 2 *There is a phase* p, $0 \le p \le P-1$ *for which* $S_{pm} = S_{pm+1} = \cdots = S_{(p+1)m}$.

Proof We start with the following claim: $S_b \preceq S_{b+1}$ for $b = 0, \ldots, B-2$. (See Fig. 2 for illustration.) We prove this claim by contradiction. Suppose that $S_b \not\preceq S_{b+1}$ for some b. Then choose any x for which $|S_b|_{\le x} < |S_{b+1}|_{\le x}$. Let $X = \{z \in S_b : z > x\}$ and $Y = \{z \in S_{b+1} : z \le x\}$. Then $|X \cup Y| > k$ and each interval in $X \cup Y$ contains the slot $I = (bn+x+n, bn+x+n+1)$. (More precisely, if $z \in X$ then I is contained in $(s_{b,z}, t_{b,z})$, and if $z \in Y$ then I is contained in $(s_{b+1,z}, t_{b+1,z})$.) This contradicts the feasibility of S.

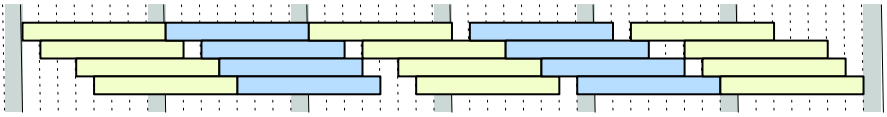

Fig. 2 An illustration of bundle dominance in the proof of Lemma 2. The instance is for $n = 8$ and $k = 4$. Here we have five consecutive sets $S_b = \{0, 1, 3, 4\}$, $S_{b+1} = \{0, 2, 3, 4\}$, $S_{b+2} = \{0, 3, 5, 6\}$, $S_{b+3} = \{1, 3, 5, 7\}$, $S_{b+4} = \{2, 5, 6, 7\}$

Call a phase *good* if it satisfies the lemma and *bad* otherwise. Each bad phase must contain a bundle b for which $S_b \prec S_{b+1}$. By the claim above and Lemma 1, the number of bad phases is at most $P - 1$, so, by the pigeon-hole principle, there must exist a good phase, and the lemma follows. □

The intuition is this: The sets S_b will correspond to a vertex cover and transitions between consecutive bundles will be used to verify the correctness of this cover. Each phase has m such transitions, each one corresponding to one edge and verifying if this edge is covered. In order for this to work, all edges must be verified against the same set S_b. Lemma 2 above guarantees that there will be some phase in which all the sets S_b (and including one set right after phase b) will indeed be the same.

4 An Extension of Interval Packing

We now extend IntvPack-Card as follows: in addition to W, ℓ and the set of intervals (s_i, t_i), $i = 0, \ldots, N - 1$, we are also given a set Γ of pairs (α, β) of numbers. We want to decide whether there is a subset S of at least ℓ intervals that satisfies the following two conditions:

(i) $w(cut_\gamma(S)) \leq W$ for all γ (as before), and
(ii) $\min\{w(cut_\alpha(S)), w(cut_\beta(S))\} \leq W - 1$ for each $(\alpha, \beta) \in \Gamma$.

Intuitively, each pair $(\alpha, \beta) \in \Gamma$ represents a "bottleneck pair", where the weight bound is lower by 1, but only one of these tighter bounds needs to be met, not necessarily both. We will refer to this version as ExtIntvPack.

Our goal in this section is to establish $\mathbb{NP}$-completeness of ExtIntvPack. We transform the given instance $G = (V, E)$, k of VertexCover into an instance of ExtIntvPack. As in the previous section, let $P = k(n - k) + 1$ and $B = mP + 1$. The instance of IntvPack-Card will have bounds $W = k$, $\ell = kB$, and will contain $N = nB$ unit-weight intervals

$$(s_{b,z}, t_{b,z}) = (bn + z, bn + z + n - \delta_{b,z}),$$

for $b = 0, \ldots, B - 1$ and $z = 0, \ldots, n - 1$, where each $\delta_{b,z} \in \{0, \frac{1}{2}\}$ is determined as follows. For the last bundle $B - 1$, we let all $\delta_{B-1,z} = 0$. Let $b \leq B - 2$. Each such b is associated with one edge, with all bundles in each phase associated with different edges. Assume that the edges of G are numbered, say $E = \{e_0, \ldots, e_{m-1}\}$. If $b = pm + a$, for some phase p, then we say that b is *associated* with edge e_a. If $e_a = (u, v)$, then we set $\delta_{b,z} = \frac{1}{2}$ for $z \in \{u, v\}$ and $\delta_{b,z} = 0$ for $z \notin \{u, v\}$.

Next we need to define Γ. We let $\Gamma = \{(\alpha_b, \beta_b)\}_{b=0,\ldots,B-2}$, where each pair (α_b, β_b) is defined as follows: If $e_a = (u, v)$ is the edge associated with b, then $\alpha_b = t_{b,u} + \frac{1}{4} = bn + u + n - \frac{1}{4}$ and $\beta_b = t_{b,v} + \frac{1}{4} = bn + v + n - \frac{1}{4}$.

Theorem 3 *Problem* ExtIntvPack *is strongly* $\mathbb{NP}$*-complete.*

Proof Let $\mathcal{I}$ be the instance of ExtIntvPack constructed above. It is sufficient to prove that G has a vertex cover of size k if and only if $\mathcal{I}$ has a solution.

($\Rightarrow$) Suppose that U is a vertex cover of G of cardinality k. We will specify the solution S by the sets S_b of intervals selected from each bundle b. We simply let $S_b = U$ for each b. Since we choose the same k intervals from each bundle, we have $w(cut_\gamma(S)) \leq k$ for all reals γ; thus condition (i) is satisfied. To verify (ii), consider any $(\alpha_b, \beta_b) \in \Gamma$, for a bundle b associated with an edge $e_a = (u, v)$. Since U is a vertex cover, we either have $u \in U$ or $v \in U$. Without loss of generality, assume $u \in U$ (the other case is symmetric). Then, by the construction of the intervals in $\mathcal{I}$, we have $t_{b,u} < \alpha_b < s_{b+1,u}$, which means that the intervals in S corresponding to u do not intersect α_b; thus $w(cut_{\alpha_b}(S)) \leq k - 1$, proving that condition (ii) holds.

($\Leftarrow$) Now, suppose that $\mathcal{I}$ has a solution S. As before, letting each S_b be the set of intervals from bundle b that are in S, we must have $|S_b| = k$ for all b. By Lemma 2, there is a phase p for which $S_{pm} = S_{pm+1} = \cdots = S_{(p+1)m}$. (Note that, even though we adjusted end-points of some intervals, Lemma 2 still holds, since we decreased these endpoints only by $\frac{1}{2}$, without changing their intersection pattern.) We take $U = S_{pm}$, and we claim that U is a vertex cover. Indeed, let $e_a = (u, v) \in E$ be any edge and take the bundle $b = pm + a$ in phase p associated with edge e_a. By condition (ii) in the definition of ExtIntvPack, we have $w(cut_{\alpha_b}(S)) \leq k - 1$ or $w(cut_{\beta_b}(S)) \leq k - 1$. Without loss of generality, assume that $w(cut_{\alpha_b}(S)) \leq k - 1$ (the other case is symmetric). All intervals $\{z \in S_b : z > u\}$ from bundle b and all intervals $\{z \in S_{b+1} : z < u\}$ from bundle $b + 1$ intersect α_b. Since $S_b = S_{b+1} = U$, this means that $u \in U$, for otherwise this would give us k intervals intersecting α_b, violating the bottleneck bound at α_b. This holds for all edges e_a; therefore we can conclude that U is indeed a vertex cover of G of size k. □

5 Strong $\mathbb{NP}$-completeness of IntvPack-Card

We now show how to "implement" the construction from the previous section using IntvPack-Card. Again, let $G = (V, E)$, k be an instance of VertexCover. To streamline the argument, we assume that $n \geq 4$, $k \leq n - 2$ and that vertices 0 and $n - 1$ of G are isolated, so that, without loss of generality, they will not belong to any vertex cover of size k. This assumption does not affect the $\mathbb{NP}$-completeness of VertexCover. We transform G, k into an instance $\mathcal{J}$ of IntvPack-Card. $\mathcal{J}$ will contain the same intervals as $\mathcal{I}$ from the previous section, plus some additional ones. However, we change the bounds W and ℓ, and we add many more small intervals that will be used to enforce the bound of k on the number of intervals chosen from each bundle and to simulate the bottleneck pairs.

The general idea of the proof is this. The bound on the cut weight in $\mathcal{J}$ will be $W = 2k + 1$. It is useful to visualize a solution of $\mathcal{J}$ as a packing of a horizontal

strip of height $2k+1$ where intervals are represented by rectangles whose heights are equal to their weights. This strip is divided into two tracks: the "upper" track of height k used to simulate the packing of $\mathcal{I}$ from Sect. 4, and the "lower" track used for additional intervals that we call *obstacles*. These obstacles are used to force the packing of the top track to behave exactly as in the reduction in Sect. 4. For example, we will have a large number of obstacles of weight $k+1$ in the central slots. There will be so many of such obstacles that any solution will have to have at least one such obstacle in each central slot, thus forcing the top track to have height at most k. Each bottleneck pair (α, β) will be simulated by a gadget that has a number of other obstacles, grouped into two appropriately overlapping chains, one corresponding to α and the other to β. Any solution will be forced to include exactly one of these chains.

We now proceed with the formal proof. Choose first some large even constant D, say $D = 2n^6$. The bounds in the instance of IntvPack-Card will be $W = 2k+1$ and $\ell = kB + D^2B + D(n-1)(B-1)$. We include in $\mathcal{J}$ the same nB unit-weight intervals as in $\mathcal{I}$: namely all $(s_{b,z}, t_{b,z}) = (bn+z, bn+z+n-\delta_{b,z})$, for $b = 0, \ldots, B-1$ and $z = 0, \ldots, n-1$, where each $\delta_{b,z} \in \{0, \frac{1}{2}\}$ is determined as before, that is, $\delta_{b,z} = \frac{1}{2}$ for intervals z that correspond to the endpoints of the edge associated with b (see the previous section). We will refer to these intervals as *bundle intervals*.

Now we add to $\mathcal{J}$ new intervals called *obstacles*. Most of the obstacles will have length either $\varepsilon = 1/D$ or ε^2 and weight $k+1$, but a few of them (two per bundle) will have length $\varepsilon/2$ and weight $k+2$.

The first category of obstacle intervals is called *plain obstacles*. For each bundle b, we introduce D^2 disjoint obstacles of length ε^2 that fill its central slot (λ_b, λ'_b), namely intervals $(\lambda_b + g\varepsilon^2, \lambda_b + (g+1)\varepsilon^2)$, for $g = 0, \ldots, D^2-1$. All these intervals have weight $k+1$. Thus we have D^2B plain obstacles in the central slots.

More plain obstacles are introduced between central slots of any two consecutive bundles. For each bundle $b \le B-2$ we proceed as follows. Let $e_a = (u, v)$ be the edge associated with b, where $u < v$. Recall that, according to our assumption about the instance of VertexCover, we have $u \ge 1$ and $v \le n-2$, so $t_{b,u} > \lambda'_b$ and $s_{b+1,v} < \lambda_{b+1}$. We fill intervals $(\lambda'_b, t_{b,u})$ and $(s_{b+1,v}, \lambda_{b+1})$ with plain obstacles of length ε. These obstacles are $(\lambda'_b + g\varepsilon, \lambda'_b + (g+1)\varepsilon)$, for $g = 0, \ldots, D(u - \frac{1}{2}) - 1$, and $(s_{b+1,v} + g\varepsilon, s_{b+1,v} + (g+1)\varepsilon)$, for $g = 0, \ldots, D(n-v-1) - 1$.

Now, we introduce two groups of obstacles in the interval $(t_{b,u}, s_{b+1,v})$. The intervals in the first group are called *α-obstacles* and those in the other group *β-obstacles*. We have $D(v - u + \frac{1}{2})$ obstacles of each of these two types. The first α-obstacle is called the *α-bottleneck*, and it is the interval $(t_{b,u} + \varepsilon/4, t_{b,u} + 3\varepsilon/4)$, with length $\varepsilon/2$ and weight $k+2$. The remaining α-obstacles are $(t_{b,u} + 3\varepsilon/4 + g\varepsilon, t_{b,u} + 3\varepsilon/4 + (g+1)\varepsilon)$, for $g = 0, \ldots, D(v - u + \frac{1}{2}) - 2$ and they all have length ε and weight $k+1$. Analogously, the β-obstacles (other than the last one) are $(t_{b,u} + \varepsilon/4 + g\varepsilon, t_{b,u} + \varepsilon/4 + (g+1)\varepsilon)$, for $g = 0, \ldots, D(v - u + \frac{1}{2}) - 2$, all with length ε and weight $k+1$. The last β-obstacle, called the *β-bottleneck*, is $(s_{b+1,v} - 3\varepsilon/4, s_{b+1,v} - \varepsilon/4)$, and it has length $\varepsilon/2$ and weight $k+2$. (See Fig. 3, for illustration.)

Note that between λ'_b and λ_{b+1} we have $D(u - \frac{1}{2}) + D(n-v-1)$ plain obstacles, $D(v - u + \frac{1}{2})$ α-obstacles, and $D(v - u + \frac{1}{2})$ β-obstacles.

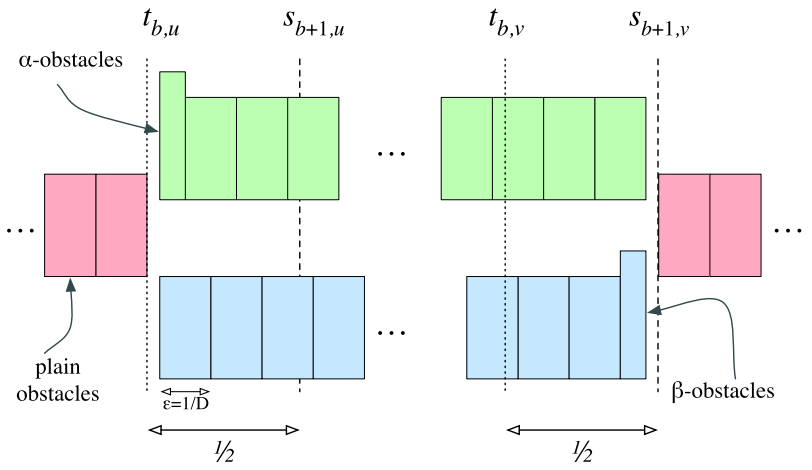

Fig. 3 Three types of obstacle intervals. The heights of the rectangles represent their weights, $k+1$ or $k+2$

Lemma 3 (a) *Any solution of instance* $\mathcal{J}$ *has at most* $D^2B + D(n-1)(B-1)$ *obstacle intervals*. (b) *If some solution of instance* $\mathcal{J}$ *has exactly* $D^2B + D(n-1)(B-1)$ *obstacle intervals then, for any bundle* $b \leq B-2$, *it must contain either the* α*-bottleneck or the* β*-bottleneck between* λ'_b *and* λ_{b+1}.

Proof (a) Consider a bundle $b \leq B-2$, and let (u, v) be the edge associated with b, where $1 \leq u < v \leq n-2$. We claim that any solution contains at most $D(v-u+\frac{1}{2})$ obstacles between $t_{b,u}$ and $s_{b+1,v}$. This can be justified as follows: Order all the obstacles in this range in order of increasing left endpoints, starting with the α-bottleneck (and ending with the β-bottleneck). This will give us a sequence of $2D(v-u+\frac{1}{2})$ intervals where each one (except the last one) intersects the next one. No two intersecting obstacles can be in the solution, because of the weight constraint. Therefore at most half of the obstacles in this sequence can be in the solution—proving our claim.

There are $D(u-\frac{1}{2}) + D(n-v-1)$ plain obstacles between λ'_b and λ_{b+1}. By the previous paragraph, any solution can contain at most $D(v-u+\frac{1}{2})$ α- or β-obstacles in this range, for the total of $D(n-1)$ obstacles. Multiplying it by $B-1$ bundles and adding D^2B plain obstacles in central slots, we obtain (a).

(b) Consider a bundle $b \leq B-2$ whose associated edge is (u, v), for $1 \leq u < v \leq n-2$. By the way the maximum number of obstacles in (a) is realized, it is sufficient to show that if a solution contains $D(v-u+\frac{1}{2})$ obstacles between $t_{b,u}$ and $s_{b+1,v}$ then it must contain at least one of the two bottlenecks in this range. Suppose, towards contradiction, that it does not. Order these obstacles from left to right, as in (a). Without the two bottlenecks, the ordering will contain $2D(v-u+\frac{1}{2})-2$ obstacles, and the solution can contain at most half of them, that is at most $D(v-u+\frac{1}{2})-1$ intervals—a contradiction. □

Theorem 4 *Problem* IntvPack-Card *is strongly* $\mathbb{NP}$*-complete.*

Proof Let $\mathcal{J}$ be the instance of IntvPack-Card constructed above. It is sufficient to prove that G has a vertex cover of size k if and only if $\mathcal{J}$ has a solution. The proof mimics the proof of Theorem 3, "simulating" the constraints from that proof using obstacle intervals.

($\Rightarrow$) Suppose that U is a vertex cover of G of cardinality k. By our assumption, U does not contain vertices 0 and $n-1$. We define a solution S of $\mathcal{J}$. The bundle intervals in S are specified by the sets S_b of intervals selected from each bundle b. We simply let $S_b = U$ for each b, as before. This will give us kB intervals. Since we choose the same k intervals from each bundle, these bundle intervals will have total weight at most k at each cut point. Next, we add to S all D^2B plain obstacles in central slots. Finally, for each bundle $b = 0, \ldots, B-2$ we proceed as follows: We add to S all $D(u-\frac{1}{2}) + D(n-v-1)$ plain obstacles between λ'_b and λ_{b+1}. If $e_a = (u, v)$, where $1 \le u < v \le n-2$, is the edge associated with b, then either $u \in U$ or $v \in U$. If $u \in U$ then we add to S all α-obstacles between λ'_b and λ_{b+1}; otherwise we add to S all β-obstacles in this range. In either case, we add $D(v-u+\frac{1}{2})$ of those obstacles. Then the total number of obstacles between λ'_b and λ_{b+1} will be $D(n-1)$, so, overall, we will have $|S| = kB + D^2B + D(n-1)(B-1) = \ell$.

It remains to verify the bound on weight. Consider any γ. This γ is intersected by at most one obstacle and at most k bundle intervals. If this obstacle is not an α- or β-bottleneck, then its weight is $k+1$, so $w(cut_\gamma(S)) \le k + (k+1) = W$. Suppose that this obstacle is the α-obstacle (the case of the β-obstacle is symmetric) between λ'_b and λ_{b+1}, where b is associated with an edge $e_a = (u, v)$, for $u < v$. By the definition of S, since we included the α-bottleneck in S, we must have $u \in U$. Further, by the definition of the α-bottleneck for b, we also have $t_{b,u} < \gamma < s_{b+1,u}$—in other words, γ is not contained in any bundle interval corresponding to u. Thus at most $k-1$ bundle intervals intersect γ, implying that $w(cut_\gamma(S)) \le k-1+(k+2) = W$.

($\Leftarrow$) Now suppose that S is a solution for $\mathcal{J}$. First, we argue that S must contain at least one obstacle in each central slot. The idea here is simple: since D is so large and each central slot has D^2 obstacles, if we did not include any obstacles from some central slot, S cannot have ℓ intervals, even if we included in S all other intervals in the instance, ignoring feasibility. More formally, if S does not contain all intervals from some central slot, then it has at most $D^2(B-1)$ plain obstacles in central slots. Since other obstacles have length at least $\varepsilon/2$, S can contain at most $(\lambda_{B-1} - \lambda'_0)2D \le 2DnB$ of those other obstacles. The number of bundle intervals in S is at most Bn. By simple calculation, $B \le n^4$. So the total number of intervals in S would be at most $D^2(B-1) + 2DnB + nB \le D^2B - 4n^{12} + 4n^{10} + n^5 < D^2B < \ell$, as $n \ge 4$.

By the previous paragraph, S has at least one obstacle interval in each central slot. This implies that S contains at most k intervals from each bundle. Thus, applying Lemma 3, we obtain the following:

(a) S has at most k intervals from each of the B bundles,
(b) S has at most D^2 plain obstacles in each of the B central slots,
(c) S has at most $D(n-1)$ obstacles (of type plain, α or β) in each of the $B-1$ intervals between two consecutive central slots.

Since $\ell = kB + D^2B + D(n-1)(B-1)$, S must contain the *exact* numbers of intervals given above in each of the categories (a), (b) and (c).

The first important consequence of the observation above is that S contains exactly k intervals from each bundle. Thus we can represent S, yet again, by the sets S_b of intervals from each bundle b, and we will have $|S_b| = k$ for all k. Further, the previous paragraph implies that each slot contains an obstacle, so at each cut the total weight of the bundle intervals is at most k. Therefore, as in the previous section, we will have $S_b \preceq S_{b+1}$ for $b = 0, \ldots, B-2$. By Lemma 2, there must be a phase p where $S_{pm} = S_{pm+1} = \cdots = S_{(p+1)m}$.

As before, we claim that $U = S_{pm}$ is a vertex cover. Recall that S contains $D^2B + D(n-1)(B-1)$ obstacle intervals. By Lemma 3, the only way this is possible is when S contains at least one bottleneck in each range between λ'_b and λ_{b+1}, for each bundle $b = 0, \ldots, B-2$. Let $e_a = (u, v) \in E$, where $1 \le u < v \le n-2$, be any edge and take the bundle $b = pm + a$ in phase p associated with edge e_a. Without loss of generality, suppose that S contains the α-bottleneck between λ'_b and λ_{b+1}, that is the interval $(t_{b,u} + 3\varepsilon/4 + g\varepsilon, t_{b,u} + 3\varepsilon/4 + (g+1)\varepsilon)$. Taking any γ from this interval, this γ must be intersected by at most $k-1$ bundle intervals, which is possible only if $u \in S_b = U$. In other words, e_a is covered by U. Since this holds for any edge e_a, we can conclude that U is a vertex cover of G. □

6 Strong $\mathbb{NP}$-completeness of IntvPack-Weight

The $\mathbb{NP}$-completeness proof for IntvPack-Weight follows the idea from the previous section, with the following modifications: Change the size of the obstacles to $p(n)$, for some large polynomial $p()$. As before, the bottlenecks are 1 unit higher, $p(n) + 1$. Now let's take $W = p(n) + k$ and $L = kB + M$, where M is the total weight of plain obstacles plus the total weight of all α-obstacles. Since $p(n)$ is large, any solution must take the maximum number of obstacles, which gives us the same constraint as before, and the rest of the argument remains the same.

Theorem 5 *Problem* IntvPack-Weight *is strongly* $\mathbb{NP}$*-complete.*

Acknowledgements This project was carried out, in part, at the workshop on 'Adaptive, output-sensitive, on-line, and parameterized algorithms', Schloss Dagstuhl, Germany, April 19–24, 2009. M. Chrobak has been supported by the NSF Grant CCF-0729071. G. Woeginger has been supported by the Netherlands Organization for Scientific Research (NWO), grant 639.033.403, and by BSIK grant 03018.

We would also like to thank Khaled Elbassioni for pointing to us the connection between interval packing and unsplittable flows on line graphs.

References

1. Albers, S., Arora, S., Khanna, S.: Page replacement for general caching problems. In: Proc. 10th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'99), pp. 31–40 (1999)

2. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: Proc. 38th Annual ACM Symposium on Theory of Computing (STOC'06), pp. 721–729 (2006)
3. Bansal, N., Buchbinder, N., Naor, J.: Randomized competitive algorithms for generalized caching. In: Proc. 40th Annual ACM Symposium on Theory of Computing (STOC'08), pp. 235–244 (2008)
4. Bansal, N., Friggstad, Z., Khandekar, R., Salavatipour, M.R.: A logarithmic approximation for unsplittable flow on line graphs. In: Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09), pp. 702–709 (2009)
5. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. J. ACM **48**, 1069–1090 (2000)
6. Belady, L.A.: A study of replacement algorithms for virtual-storage computer. IBM Syst. J. **5**, 78–101 (1966)
7. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
8. Brehob, M., Wagner, S., Torng, E., Enbody, R.: Optimal replacement is NP-hard for non-standard caches. IEEE Trans. Comput. **53**, 73–76 (2004)
9. Cao, P., Irani, S.: Cost-aware www proxy caching algorithms. In: Proc. USENIX Symposium on Internet Technologies and Systems, pp. 193–206 (1997)
10. Chrobak, M., Karloff, H.J., Payne, T.H., Vishwanathan, S.: New results on server problems. SIAM J. Discrete Math. **4**, 172–181 (1991)
11. Fiat, A.: Unpublished manuscript, 1997
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of $\mathbb{NP}$-Completeness. Freeman, San Francisco (1979)
13. Irani, S.: Page replacement with multi-size pages and applications to web caching. Algorithmica **33**, 384–409 (1997)
14. Young, N.E.: On-line file caching. Algorithmica **33**, 371–383 (2002)