Publication IV

Lauri Ahlroth, Olli Pottonen, André Schumacher. Approximately uniform online checkpointing with bounded memory. Invited to *Algorithmica special issues of 17th Annual International Computing and Combinatorics Conference*, 13 pages, under review, December 2011.

An earlier conference version appeared with the title “Approximately uniform online checkpointing” in *Proceedings of 17th Annual International Computing and Combinatorics Conference*, Dallas, Texas, USA, 297-306, August 2011, ©Springer.

COCOON'11 Special Issues manuscript No.
(will be inserted by the editor)

Approximately Uniform Online Checkpointing With Bounded Memory

Lauri Ahlroth · Olli Pottonen · André Schumacher

Received: date / Accepted: date

Abstract In many complex computational processes one may want to store a sample of the process' history for later use by placing checkpoints. In this paper we consider the problem of maintaining, in an online fashion, a collection of k checkpoints as an approximately uniformly spaced sample in the history of a continuous-time process. We present deterministic algorithms tailored for small values of k and a general one for arbitrary k. The algorithms are proven to be close to optimum for several different measures.

Keywords checkpointing · online algorithm · uniform spacing

This research has been supported by the Academy of Finland under grant 128823. The third author has been supported also partly by the Helsinki Graduate School of Computer Science and Engineering. The main progress of the paper has been done while the second author was affiliated with Aalto University, Department of Information and Computer Science. A preliminary version of this work appeared in the 17th Annual International Computing and Combinatorics Conference, Dallas, Texas, USA, 2011.

L. Ahlroth
Helsinki Institute for Information Technology
Aalto University, Department of Information and Computer Science
P.O.Box 15400, FI-00076 Aalto, Finland
E-mail: Lauri.Ahlroth@aalto.fi

O. Pottonen
Departament de Llenguatges i Sistemes Informatics
Universitat Politecnica de. Catalunya, Barcelona, Spain
E-mail: Olli.Pottonen@iki.fi

A. Schumacher
Aalto University, Department of Information and Computer Science
P.O.Box 15400, FI-00076 Aalto, Finland
E-mail: Andre.Schumacher@aalto.fi

1 Introduction

Checkpointing means storing the state of a computer process at some moments for further use. In the most obvious application the checkpoints are backups that can be used to restore the process after an error. This application is considered by e.g. Ziv and Bruck [13], who developed an online checkpointing algorithm that minimizes the cost of placing checkpoints, and Nicola and Spanje [8], who compare checkpointing methods for databases.

Checkpointing can be used more generally to maintain a sample from the history of the process. For example when adaptively compressing a large file, decompressing a part of it requires knowledge of the compression statistics at the time of compressing that particular part. If the statistics are obtained from a nearby checkpoint, the process is greatly accelerated. Bern et al. [3] present an algorithm for this kind of checkpointing problem.

When solving certain differential equations one needs to iterate over the intermediate results in reverse order. If storing all intermediate data is infeasible, a specific checkpointing algorithm is needed to avoid repeating the same computations numerous times. Sturm and Walther solve this problem in offline [10] and online [11] settings.

Checkpointing also employs some principles from the field of streaming algorithms [2, 7], which have received increasing attention in recent years. Streaming algorithms typically operate on a single pass over a large data input while using sublinear memory or time with respect to input size. This description seems to fit many real-time checkpointing models.

The decompression example above is one case where a sample of the history is required. Other such cases include a simulation or experiment where the intention is to study the evolution of the process afterwards and rerun parts of the process with different input. When there is no application specific information, spreading the checkpoints uniformly throughout the history appears to be the most natural goal. By uniform we mean that the distances between consecutive checkpoints are equal, which should not be confused with random samples from a uniform distribution.

In this paper we develop algorithms for maintaining the checkpoint placement as approximately uniform in an online setting. Our model is different from models previously studied, e.g., in [3, 13], since we do not impose costs for either placing or modifying checkpoints. However, we constrain an algorithm to use at most k checkpoints and limit its actions to choose between placing a new checkpoint at the current state of the process, if available, replacing a previous checkpoint, or waiting. Our performance measure reflects uniformness of the checkpoint placement. The resulting problem has an interesting combinatorial structure, and we are not aware of any prior work considering this type of checkpointing problem.

Recently, Teramoto et al. [12] and Asano [1] studied the related problem of finding point sequences to be inserted into the unit interval that achieve a close to uniform spacing along the sequence of insertion. Although related, our model is different from [1, 12] since we only allow the presence of a constant

number of points at any time. Further, due to the online nature of our problem, an algorithm can place checkpoints only at the current time horizon, which is progressing as time proceeds. Another related problem is to compute low-discrepancy sequences [4, 6, 9] of the unit interval. Although mainly beneficial in high dimensions, a well-known example of a low-discrepancy sequence is the one-dimensional van der Corput sequence [5, as cited in [6]] whose first m elements represent a reasonably uniformly spaced sample on the unit interval for each $m \geq 1$.

This paper is organized as follows. In Section 2 we formally define the checkpointing problem. Section 3 discusses online algorithms for solving it. Some of our algorithms are tailored to small values for k, while one algorithm performs slightly worse for small k but has asymptotically optimal performance, as we show in Section 4. We also consider the problem under alternative natural uniformness measures and present a few results in Section 5. Section 6 contains our conclusions and briefly discusses future work.

2 Model

We consider a process which starts at time 0 and halts at time T, where T is not known in advance. Time can be modeled as a real valued or integral variable, that is, time can be either continuous or discrete. In this paper we discuss primarily algorithms that work with continuous time. They can be transformed to discrete time algorithms of comparable performance by simple rounding. Even if the performance gets worse for small T due to rounding effects, the difference vanishes asymptotically.

An algorithm has enough memory for $k \geq 2$ checkpoints, which save the state of the process at a certain point in time. There are no limits or costs associated with storing checkpoints. The goal is to have the checkpoints placed as evenly as possible on the interval $[0, T]$. As boundary conditions we assume a permanent checkpoint on time 0 and a rolling checkpoint at T, neither of which is included in the limit of k replaceable checkpoints.

During the execution of an algorithm, the k checkpoints and the boundaries divide the time interval from start to present into $k + 1$ intervals of lengths $l_0, l_1, \ldots, l_k$. The actions that the algorithm can take are rather limited. At any moment it can either wait or place a checkpoint at the present state, removing one of the earlier checkpoints unless there still are unplaced checkpoints available. Considering the intervals, the algorithm can at any moment either let the last interval grow, or merge two consecutive intervals, thereby introducing a new interval of zero length.

If an algorithm performs perfectly, at time T the checkpoints are placed uniformly, i.e., $l_0 = l_1 = \cdots = l_k$. As a measure of uniformness of the checkpoint placements we define the *gap ratio*

$$r(t) = \frac{\max_{0 \leq j \leq k} l_j(t)}{\min_{0 \leq j \leq k-1} l_j(t)} \ , \tag{1}$$

which is to be kept small at all times. Two other measures are considered briefly in Section 5. This measure (1) is essentially the one-dimensional special case of the ratio introduced in [12] for the study of a related problem. The last interval l_k is excluded when taking the minimum in the denominator, since otherwise the inevitable action of placing a checkpoint at the present state would lead to an infinite modified gap ratio for any algorithm. Note that as the interval lengths l_j vary over time, the ratio r varies also, but for a fixed algorithm both the lengths and the ratio depend only on time.

We consider deterministic algorithms and their worst-case performance, meaning that T is chosen in the least favorable way. However, in the absence of a lower bound on T the problem becomes meaningless. Every algorithm has a positive time t_1 when it is going to place the first checkpoint. For $T < t_1$ the process halts before any checkpoints are placed, implying extremely bad algorithmic performance. To avoid this initialization problem we assume $T \geq c > 0$ for some constant c which is known to the algorithm. The actual value of c does not matter as we can scale time as we please. Hence the performance metric considered in this work is

$$g = \sup_{T \geq c} r(T) \ , \tag{2}$$

where $r(t)$ is defined in (1). We refer to g as the *maximum gap ratio*. In online algorithms terminology, g matches the competitive ratio of an online algorithm compared to an optimal offline algorithm that knows T beforehand and achieves $r(T) = 1$. Note that by taking the supremum over the halting time T we essentially require the checkpoints to remain approximately uniformly spaced at all times.

3 Checkpointing Algorithms

In this section we present several algorithms for the checkpointing problem. We start with a simple algorithm that is in fact asymptotically optimal for $k \to \infty$, as we show in Section 4. For small k, however, we are able to prove better performance guarantees for algorithms discussed in Section 3.3.

All of our algorithms are based on the following idea of *cyclicity*. If one can find a checkpoint placing procedure that transforms the interval length configuration as

$$(l_0,\ l_1,\ \ldots,\ l_k) \to (\gamma\, l_0,\ \gamma\, l_1,\ \ldots,\ \gamma\, l_k) \tag{3}$$

for some $\gamma > 1$, then by essentially scaling down the unit of time and thus the l_j by γ one again obtains the initial configuration. Repeating the procedure leads to an algorithm that can run indefinitely. Note that the gap ratio $r(t)$ in (1) is invariant under uniform scaling of the l_j values. Hence, the maximum gap ratio of this algorithm can be determined by examining just the first transition cycle.

```
Place a checkpoint on each integer between 1 and k;
j ← 2;
while True do
    Wait until current time t is divisible by j;
    Place a checkpoint at present time t, removing the checkpoint of the earliest
    time not divisible by j;
    if t = kj then
        j ← 2j;
    end
end
```

Algorithm 1: Powers-of-two algorithm

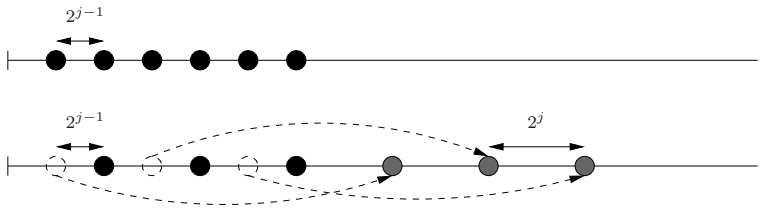

Fig. 1 Illustration of the Powers-of-two algorithm, $k = 6$.

3.1 Powers-of-Two Algorithm

There is a simple algorithm that works for any k and has maximum gap ratio $g = 2$. It only places checkpoints at integer times, and the checkpoints are placed so that the interval lengths are powers of two. Also checkpoints are placed on consecutive multiples of a power of two, and these facts inspired the name.

We start by placing the k checkpoints on the integers $1, 2, \ldots, k$, which gives a uniformly spaced sample on the interval $[0, k]$. The idea is to scale this configuration with $\gamma = 2$, eventually yielding a uniformly spaced sample on $[0, 2k]$. This is achieved by simply removing every other checkpoint on $[0, k]$ and placing new checkpoints on $[k+1, 2k]$ so that we finally have a checkpoint on every even number. From here on we continue cyclically by applying the same procedure in increasing scales.

Pseudocode for this method is given as Algorithm 1. Observe that j in the pseudocode is always a power of two, and j is doubled when we have a checkpoint on each of $j,\ 2j,\ 3j,\ \ldots,\ kj$ where kj is the current time. One should note that Algorithm 1 is similar to the greedy algorithm proposed in [12] if one considers the set of active checkpoints at any given time. Here, however, we need to consider the eviction of a checkpoint from the selection once the maximum number has been reached and a new checkpoint is placed at the end of current time horizon.

3.2 Golden Ratio Approach

The previously discussed Powers-of-two algorithm is appealing due to its simplicity. For small values of k, however, one is able to find algorithms that guarantee a maximum gap ratio strictly smaller than 2. Following the cyclicity discussion in the beginning of Section 3, it remains to explicitly build the chain of transitions (3) such that the gap ratio stays below a constant strictly less than 2.

One method of building these transitions is to require interval lengths l_j to be integer powers of a common base ϕ. When merging two intervals, the resulting interval should have a length that is again a power of ϕ. Hence, we consider ϕ_n defined as the unique positive root of

$$1 + \phi_n = \phi_n^n \tag{4}$$

where $n \geq 2$. Now ϕ_2 is the golden ratio, and we refer to ϕ_n as the *generalized golden ratio*.

In our case (4) formulates that by merging two consecutive intervals of lengths 1 and ϕ_n one obtains an interval of length ϕ_n^n. Further, by scaling both sides of the equation by ϕ_n^a it is apparent how the algorithm can also merge larger intervals into a single interval that is still a power of ϕ_n.

It is straightforward to show $\phi_n \in (1, 2)$ and

$$\phi_n^{n-1} = 1 + \frac{1}{\phi_n} \in \left(\frac{3}{2}, 2\right) \ . \tag{5}$$

Hence, an algorithm is an improvement over the Powers-of-two algorithm if it satisfies $r(t) \leq \phi_n^{n-1}$ for a complete cycle. In the following we present several algorithms which satisfy this condition and achieve $g = \phi_n^{n-1}$.

3.3 Golden Ratio Algorithms

For $k = 2$ define $F_j = \phi_2^j, j \geq 0$, so that F_j satisfies the recursion $F_{j+1} = F_j + F_{j-1}$, which follows from (4). Now consider Algorithm 2 which places a checkpoint at each F_j, always erasing the oldest checkpoint. At any point in time the checkpoints occur at F_{j-1} and F_j for some j, the interval lengths being $l_0 = F_{j-1}, l_1 = F_j - F_{j-1} = F_{j-2}, l_2 \leq F_{j-1}$. Most importantly, the gap ratio ratio (1) stays always at $F_j/F_{j-1} = \phi_2$. This algorithm is illustrated in Fig. 2.

A discrete time variant of the algorithm maintains checkpoints at two latest Fibonacci numbers. In this variant the gap ratio (1) approaches ϕ_2 as time tends to infinity.

In order to present similar algorithms for $k = 3, 4, 5$ we introduce a tabular representation of the algorithms as shown in Figure 3. Each row of a table lists the relative lengths of the intervals. Going from one row to the next, we either just wait and increase the length of the last interval, or place a new checkpoint at the present time, thereby merging two earlier intervals. These intervals are

Place checkpoints at F_1 and F_2;
$j \leftarrow 1$;
while *True* **do**
    Wait until time F_{j+1};
    Place checkpoint at present time F_{j+1}, removing checkpoint at F_{j-1};
    $j \leftarrow j + 1$;
end

Algorithm 2: Golden ratio algorithm, $k = 2$.

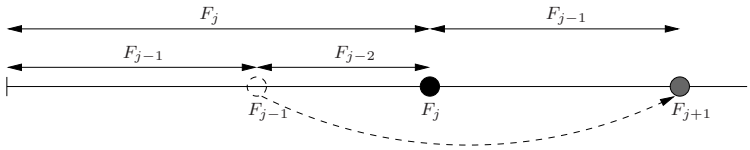

Fig. 2 Illustration of the golden ratio algorithm, $k = 2$.

$$\begin{array}{ccc} \phi & 1 & 0 \\ \underbrace{\phi \quad 1}_{} & & \phi \\ \phi^2 & \phi & 0 \end{array}$$

(a) $k = 2$

$$\begin{array}{cccc} \phi & 1 & \phi & 0 \\ \underbrace{\phi \quad 1}_{} & & \phi & \phi \\ \phi^2 & \phi & \phi & 0 \\ \phi^2 & \phi & \underbrace{\phi \quad \phi^2 - \phi}_{} & \\ \phi^2 & \phi & \phi^2 & 0 \end{array}$$

(b) $k = 3$

$$\begin{array}{ccccc} \omega & 1 & \omega^2 & \omega & 0 \\ \underbrace{\omega \quad 1}_{} & & \omega^2 & \omega & \omega^2 \\ \omega^3 & \omega^2 & \omega & \omega^2 & 0 \\ \omega^3 & \omega^2 & \underbrace{\omega \quad \omega^2}_{} & & \omega^3 \\ \omega^3 & \omega^2 & \omega^4 & \omega^3 & 0 \end{array}$$

(c) $k = 4$

$$\begin{array}{cccccc} \omega & 1 & \omega^2 & \omega & \omega^2 & 0 \\ \underbrace{\omega \quad 1}_{} & & \omega^2 & \omega & \omega^2 & \omega^2 \\ \omega^3 & \omega^2 & \omega & \omega^2 & \omega^2 & 0 \\ \omega^3 & \omega^2 & \omega & \omega^2 & \underbrace{\omega^2 \quad \omega^3 - \omega^2}_{} & \\ \omega^3 & \omega^2 & \omega & \omega^2 & \omega^3 & 0 \\ \omega^3 & \omega^2 & \underbrace{\omega \quad \omega^2}_{} & & \omega^3 & \omega^3 \\ \omega^3 & \omega^2 & \omega^4 & \omega^3 & \omega^3 & 0 \\ \omega^3 & \omega^2 & \omega^4 & \omega^3 & \underbrace{\omega^3 \quad \omega^4 - \omega^3}_{} & \\ \omega^3 & \omega^2 & \omega^4 & \omega^3 & \omega^4 & 0 \end{array}$$

(d) $k = 5$

Fig. 3 Tabular representations of algorithms for different numbers of checkpoints. Note the shorthands $\phi = \phi_2$, $\omega = \phi_3$.

marked with an underbrace. Each table represents one cycle of the algorithm, resulting in a scaled up version of the initial configuration.

Algorithm 2 is represented in Figure 3(a), and similar algorithms for $k = 3, 4, 5$ in Figures 3(b), 3(c) and 3(d). They have maximum gap ratios $g = \phi_2, \phi_3^2$ and ϕ_3^2, respectively.

$$\begin{array}{ccc} 1 & 1 & 0 \\ \underbrace{1 \quad 1} & & \sqrt{2} \\ 2 & \sqrt{2} & 0 \\ 2 & \underbrace{\sqrt{2} \quad 2-\sqrt{2}} & \\ 2 & 2 & 0 \end{array}$$

Fig. 4 An optimal algorithm for $k = 2$.

We tried to apply the same golden ratio approach for constructing algorithms for $k = 6$. However an exhaustive search indicated that this is impossible, at least with base $\phi_n, n \leq 10$. It is still possible that there is another kind of algorithm that achieves a lower gap ratio than Algorithm 1, but in Section 4 we will see that only a slightly lower g is attainable at best.

3.4 Optimal Algorithm for $k = 2$

For $k = 2$, there is still a better algorithm that can be proven optimal, presented in Fig. 3.4.

The structure of this algorithm is very close to the earlier golden ratio framework. Instead of Eq. (4) the interval lengths merge through the rather trivial identity $1 + 1 = \phi_s^2$, and all intervals but the last have a length that is an integer power of $\phi_s = \sqrt{2}$. Furthermore the algorithm achieves $g = \sqrt{2}$, which is proven optimal as a special case of Theorem 1 in the next section.

4 Lower Bound for Maximum Gap Ratio

Even if the Powers-of-two algorithm does not lead to the optimal performance for small k values, we now show that asymptotically there are no better algorithms.

Theorem 1 *An online algorithm with k checkpoints has maximum gap ratio $g \geq 2^{1-\frac{1}{\lceil \frac{k+1}{2} \rceil}}$.*

Proof Assume there are k checkpoints, and denote $m = \lceil (k+1)/2 \rceil$. Fix an initial situation where all k checkpoints have been placed. We can assume that the algorithm always merges an interval of minimum length with an adjacent interval, because if it does not, $g > 2$ follows immediately and the claim of the theorem holds.

Let x_i be the length of the shortest existing interval after i checkpoints have been placed. Each of the original $k + 1$ intervals (just before the first merge) has length at most gx_0, and at least m merges are needed before they all cease to exist. Hence $x_{m-1} \leq gx_0$.

Let

$$\alpha = \sqrt[m-1]{\frac{x_{m-1}}{x_0}} \leq \sqrt[m-1]{g} \, . \tag{6}$$

Table 1 Summary of found g bounds for different values of k, and the ratios between upper and lower bounds.

k	2	3	4	5	≥ 6
Upper bound	$\sqrt{2} \approx 1.414$	$\phi_2 \approx 1.618$	$\phi_3^2 \approx 1.755$	$\phi_3^2 \approx 1.755$	2
Lower bound	$\sqrt{2} \approx 1.414$	$\sqrt{2} \approx 1.414$	$\sqrt[3]{4} \approx 1.587$	$\sqrt[3]{4} \approx 1.587$	$2^{1-o(1)}$
Ratio	1	1.145	1.106	1.106	$2^{-o(1)}$

Since α is the geometric mean of the ratios $x_1/x_0, \ldots, x_{m-1}/x_{m-2}$, not all of them can have value less than α. That is, $\alpha \geq x_{j+1}/x_j$ for some j.

After $j+1$ merges the shortest interval length is x_{j+1} and the longest is at least $2x_j$, and their ratio is at least $2x_j/x_{j+1} \geq 2/\alpha$. Now we have

$$g \geq \frac{2x_j}{x_{j+1}} \geq \frac{2}{\alpha} \geq \frac{2}{\sqrt[m-1]{g}} = 2g^{-\frac{1}{m-1}} \tag{7}$$

which implies

$$g \geq 2^{\frac{m-1}{m}} = 2^{1-\frac{1}{m}} = 2^{1-\frac{1}{\lceil \frac{k+1}{2} \rceil}} . \tag{8}$$

This finishes the proof. □

The algorithms in Section 3 achieve maximum gap ratios quite close to the lower bound of Theorem 1. See Table 4 for a summary of the results.

5 Bounds on Maximum and Average Distance to Checkpoint

In addition to the modified gap ratio (1), one can analyze other natural uniformness measures for a collection of checkpoint intervals. These include the average and maximum distance from a point in time to the nearest earlier checkpoint. Normalized to have value 1 with the uniform spacing $l_j \equiv \frac{T}{k+1}$, these measures become

$$r_{\mathrm{ad}}(T) := \frac{k+1}{T^2} \sum_{0 \leq j \leq k} l_j^2 \tag{9}$$

and

$$r_{\mathrm{md}}(T) := \frac{k+1}{T} \max_{0 \leq j \leq k} l_j , \tag{10}$$

respectively. In this section we present some results on r_{ad} and r_{md}.

The following Theorem 2 states an inequality between the measures.

Theorem 2 *If $r(T) \leq \beta$, then $r_{ad}(T) \leq r_{md}(T) \leq \beta \, \frac{k+1}{k}$.*

Proof Note that

$$T = \sum_{0 \le j \le k} l_j \ge \sum_{0 \le j \le k-1} l_j \ge \ k\, l_m,$$

implying $l_m \le \frac{T}{k}$. Similarly

$$r_{\mathrm{md}} = \frac{k+1}{T} \max_{0 \le j \le k} l_j \le \frac{k+1}{T} \beta\, l_m \le \beta\, \frac{k+1}{k}.$$

Also we have that

$$\frac{\sum_{0 \le j \le k} l_j^2}{T} \le \frac{\max_{0 \le j \le k} l_j \cdot \sum_{0 \le j \le k} l_j}{\sum_{0 \le j \le k} l_j} \le \max_{0 \le j \le k} l_j,$$

which implies $r_{\mathrm{ad}}(T) \le r_{\mathrm{md}}(T)$. □

By a simple argument one obtains a lower bound for the worst-case value of the new measures. For the proof we need to define the shorthand notations

$$S_1 := \sum_{i=0}^{k-1} l_i, \qquad S_2 := \sum_{i=0}^{k-1} l_i^2.$$

Theorem 3 *For any algorithm, the worst-case values for r_{ad} and r_{md} are at least $\frac{k+1}{k}$.*

Proof By Theorem 2 it is enough to consider the r_{ad} case. Right after placing a checkpoint one must have $l_k = 0$, and hence $T = S_1$. By a standard power mean inequality for k reals one has

$$kS_2 \ge S_1^2, \tag{11}$$

which implies

$$r_{\mathrm{ad}} = (k+1)\frac{S_2}{S_1^2} \ge \frac{k+1}{k}.$$

□

The results of Theorem 2 can be improved for the algorithms presented in Section 3. To find the exact worst-case values for each algorithm easily, we use cyclicity and the following lemma.

Lemma 1 *For any algorithm, r_{md} achieves its maximum value at a time when $l_k = 0$. If the algorithm places checkpoints such that the last interval always satisfies $l_k \le \frac{1+\frac{3}{k}}{2} S_1$, then r_{ad} is also guaranteed to achieve its maximum at a time when $l_k = 0$.*

Table 2 Summary of our best worst-case bounds for r_{ad} and r_{md} for different k and comparison to the known lower bounds of Theorem 3.

k	2	3	4	5	≥ 6
r_{ad}	1.545	1.412	1.300	1.249	$1.125\frac{k+1}{k}$
$\frac{k+1}{k}$	1.500	1.333	1.250	1.200	$*$
$r_{\mathrm{ad}}/\frac{k+1}{k}$	1.030	1.059	1.040	1.041	1.125
r_{md}	1.758	1.789	1.624	1.565	$2\frac{k+1}{k}$
$r_{\mathrm{md}}/\frac{k+1}{k}$	1.172	1.342	1.299	1.304	2

Proof We start from the latter claim. Consider r_{ad} as a function of l_k with range from 0 to the interval length l where a new checkpoint is placed. The second differential of the mapping $l_k \mapsto \frac{S_2+l_k^2}{(S_1+l_k)^2}$ is proportional to $3S_2 + S_1^2 - 2S_1 l_k$, which is nonnegative by the assumption of the lemma and Eq. (11). Thus the maximum of r_{ad} on the given range is obtained at either of the endpoints. At the latter endpoint with $l_k = l$, placement of a new checkpoint merges two intervals and introducing a new zero-length interval, increasing the value of r_{ad}. We conclude the maximum of r_{ad} is among the configurations where $l_k = 0$.

The argument for r_{md} is similar but uses only the first differential. □

The l_k condition of lemma 1 is readily verified for all algorithms presented in this paper. Violation would occur only if the last interval is long enough compared to the sum of remaining interval lengths, which is bad for uniformness in itself.

Now, finding worst-case bounds for r_{ad} and r_{md} is straightforward. Consider first the Powers-of-two algorithm.

Corollary 1 *For Powers-of-two, one has* $r_{ad}(T) \leq \frac{9}{8}\frac{k+1}{k}$ *for all* $T \geq c$.

Proof It is sufficient to show the result for the first algorithmic cycle. Considering the intervals with $0 \leq a \leq k$ intervals already merged to the double length, straightforward calculus shows the highest value of r_{ad} is reached with $a = k/3$. An upper bound for r_{ad} is hence

$$r_{\mathrm{ad}} \leq \frac{k+1}{(\frac{2k}{3}l + \frac{k}{3}2l)^2}\left(\frac{2k}{3}l^2 + \frac{k}{3}4l^2\right) = \frac{9}{8}\frac{k+1}{k}.$$

□

The maximum of r_{md} for Powers-of-two is 2 for odd k and $2\frac{k+1}{k+2}$ for even k. For algorithms presented in Figures 3 and 3.4 one needs to find the maximum over a constant number of values. We omit the details and present the final results and comparison to lower bounds in Table 5. Note that for $k = 2$, the algorithm presented in Fig. 3.4 outperforms the golden ratio version 3(a), and we present only the lower worst-case bound.

The comparison to Theorem 1 reveals that in terms of minimizing r_{ad}, Powers-of-two is within at most a factor 1.125 of the best possible. The golden ratio algorithms are even closer to proven lower bounds in the small k cases. Considering r_{md}, the gap between lower and upper bounds is a bit higher. We leave tightening of these gaps as future work.

6 Conclusions and Future Work

We considered the problem of maintaining a collection of k checkpoints as an approximately uniformly spaced sample in the history of a continuous-time process in an online fashion. Although the problem shares some similarities with previously studied checkpoint problems, the authors are not aware of any prior work addressing this problem, which has a fundamental combinatorial structure.

We propose an online algorithm for the checkpoint problem that is based on the idea of using multiples of powers of two as the checkpoint values. Although surprisingly simple, we show that the algorithm is asymptotically optimal with respect to our performance measure. For a smaller number of available checkpoint times we also provide several other algorithms and prove bounds on their performance. These algorithms are based on a generalization of the golden ratio, which directly leads to a performance guarantee slightly better than for the powers-of-two algorithm.

We also briefly study the performance of our algorithms in the light of two alternative natural performance measures, the average and the maximum distance of any point to a checkpoint. For the average distance, we show that the Powers-of-two algorithm performs close to optimal. We also provide bounds for the performance of online algorithms for a given number of checkpoints.

We note that the same type of constructions we used for obtaining provably good algorithms for small k may be applicable to other related problems, possible to a generalization of our problem to higher dimensions. Analyzing randomized algorithms would also be a natural future extension of the work.

Acknowledgements The authors are thankful to Pekka Orponen for some helpful comments.

References

1. Asano, T.: Online uniformity of integer points on a line. Information Processing Letters 109(1), 57 – 60 (2008)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 1–16. ACM (2002)
3. Bern, M., Greene, D.H., Raghunathan, A., Sudan, M.: On-line algorithms for locating checkpoints. Algorithmica 11, 33–52 (1994)
4. Chazelle, B.: The Discrepancy Method: Randomness and Complexity. Cambridge University Press (2000)

5. van der Corput, J.G.: Verteilungsfunktionen. Proc. Nederl. Akad. Wetensch. 38, 813–821 (1935)
6. Kuipers, L., Niederreiter, H.: Uniform Distribution of Sequences. Dover Publications, Inc. (2006)
7. Muthukrishnan, S.: Data streams: Algorithms and applications. Foundations and Trends in Theoretical Computer Science 1(2), 117–236 (2005)
8. Nicola, V., van Spanje, J.: Comparative analysis of different models of checkpointing and recovery. IEEE Transactions on Software Engineering 16, 807–821 (1990)
9. Niederreiter, H.: Random number generation and quasi-Monte Carlo methods. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1992)
10. Stumm, P., Walther, A.: Multistage approaches for optimal offline checkpointing. SIAM J. Sci. Comput. 31(3), 1946–1967 (2009)
11. Stumm, P., Walther, A.: New algorithms for optimal online checkpointing. SIAM Journal on Scientific Computing 32, 836–854 (2010)
12. Teramoto, S., Asano, T., Doerr, B., Katoh, N.: Inserting points uniformly at every instance. IEICE - Trans. Inf. Syst. E89-D, 2348–2356 (August 2006)
13. Ziv, A., Bruck, J.: An on-line algorithm for checkpoint placement. IEEE Transactions on Computers 46, 976–985 (1997)