Journal of Uncertain Systems
Vol.10, No.1, pp.4-9, 2016
Online at: www.jus.org.uk

How to Gauge Disruptions Caused by Garbage Collection: Towards an Efficient Algorithm

Gabriel Arellano, Edward Hudgins, David Pruitt, Adrian Veliz
Eric Freudenthal, Vladik Kreinovich*
Computer Science Department, University of Texas at El Paso, El Paso, TX 79968, USA

Received 1 March 2015; Revised 21 June 2015

Abstract

Comprehensive garbage collection is employed on a variety of computing devices, including intelligent cell phones. Garbage collection can cause prolonged user-interface pauses. In order to evaluate and compare the disruptiveness of various garbage collection strategies, it is necessary to gauge disruptions caused by garbage collection. In this paper, we describe efficient algorithms for computing metrics useful for this purpose.

Keywords: garbage collection, mobile devices, gauging customer satisfaction, efficient algorithm

1 Formulation of the Problem

Practical problem: need to minimize disruptions caused by garbage collection. In many computer-based systems – including mobile devices – it is necessary to periodically perform garbage collection. This computation can interfere with the progress of interactive programs. Garbage collection can cause intermittent prolonged pauses in gesture-driven user interfaces that can severely reduce their usability; see, e.g., [1, 2, 4].

Need to gauge the quality of different garbage collection strategies. To decrease the resulting nuisance, several different algorithms have been designed scheduling garbage collection. To decide which of them is better – and whether they are better than the currently implemented garbage collection strategies – it is necessary to be able to gauge the amount of disruption corresponding to different strategies.

What information we can use to gauge the relative quality of garbage collection strategies. A natural idea is to run a cell phone in different regimes, and for each regime, to record the times of all the garbage collections that may interfere with an interactive program's execution.

Let us denote the total number of observed time cycles by N; the corresponding moments of time will be denoted by 1, 2, ..., N. Let n denote the number of garbage collections that occurred during the observed period. For each i from 1 to n, we will denote the moment of time when the i-th garbage collection started by $\underline{t}_i$, and the last moment of time of the i-th garbage collection by $\overline{t}_i$. We will call the interval $[\underline{t}_i, \overline{t}_i]$ the *i-th disruption interval*.

From the user's viewpoint, this means that for all moments of time $\underline{t}_i$, $\underline{t}_i + 1$, ..., $\overline{t}_i$, the user could potentially experience inconvenience – if the user was using her cell phone at one (or more) of these moments of time.

This sequence of pairs of values $\underline{t}_i$ and $\overline{t}_i$ is the information that we will use to gauge the quality of different garbage collection strategies.

A natural measure of user convenience: by the probability p that the user's activity will not be interrupted. Let T denote a typical time of some activity. In these terms, if an activity starts at a moment t, then this activity will continue in moments $t + 1$, $t + 2$, ..., $t + (T - 1)$. The user will experience

*Corresponding author.
Emails: garellano88@gmail.com (G. Arellano), eghudgins@miners.utep.edu (E. Hudgins), ddpruit@miners.utep.edu (D. Pruitt), iaeveliz@miners.utep.edu (A. Veliz), efreudenthal@utep.edu (E. Freudenthal), vladik@utep.edu (V. Kreinovich).

inconvenience if an interruption occurs at one of these moments of time, i.e., if one of these moments of time is in between $\underline{t}_i$ and $\overline{t}_i$ for some i.

We want to be able to check whether an activity starting at moment t encounters disruption or not. To be able to do that, we must have a record of the system's state at T moments of time, starting from the starting point t, i.e., at moments of time t, $t+1$, ..., $t+(T-1)$. We only have records corresponding to moments from 1 to N. Thus, we are only able to perform the desired check for moments of time t for which $t+(T-1) \le N$, i.e., for which $t \le N-(T-1)$.

The user can start at any moment of time $t = 1, 2, \ldots, N-(T-1)$. There is no reason to believe that some of these moments of time are more probably than the others; thus, it is reasonable to assume that these starting moments of time are equally probable. It is thus reasonable to gauge the user's inconvenience as the probability p that an activity starting at a randomly selected moment of time t will be disrupted.

In precise terms, the probability p is equal to the ratio $p = m/(N-(T-1))$, where m denotes the total number of moments of time $t \le N-(T-1)$ for which an activity of duration T starting at the moment t encounters a disruption.

Let us describe the problem of computing the desired probability p in precise terms.

Comment. The above probability p is based on the assumption that every interruption, no matter how short, can inconvenience the user. In reality, users do not notice interruptions which are sufficiently short; see, e.g., [3]. To get a more accurate measure of user convenience, it is therefore desirable to find out which sequences of interruptions are ignored by most users, and take this into account when estimating the corresponding probability.

Computing the disruption probability p: precise formulation of the problem. As an input, we have:

- an integer N that describes the duration of time interval during which we recorded disruptions;
- an integer n, the number of disruptions that we observed;
- for each i from 1 to n, the integers $\underline{t}_i$ and $\overline{t}_i$ describing the starting point and the endpoint of the i-th disruption;
- an integer T, the duration of an activity for which we are estimating the probability that this activity will be disrupted.

Based on this input, we must compute the value $p = m/(N-(T-1))$, where m is the number of moments of time $t \le N-(T-1)$ for which one of the T moments $t, t+1, \ldots, t+(T-1)$ falls within one of the n given intervals $[\underline{t}_i, \overline{t}_i]$.

Comment. Since we are interested only in moments of time $t \le T-(N-1)$, we should ignore the disruption intervals $[\underline{t}_i, \overline{t}_i]$ which do not affect these moments of time, i.e., for which $\underline{t}_i > N-(T-1)$.

Thus, without losing generality, in our analysis, we will assume that these out-of-range disruption intervals have already been discarded, i.e., that $\underline{t}_i \le N-(T-1)$ for all disruption intervals.

A straightforward way to compute the desired probability. In principle, we can compute the desired probability p as follows. For each moment t from 1 to $N-(T-1)$, we can check all moments of time s from this moment to the next moment $t+(T-1)$. If a disruption occurs during (at least) one of these moments of time, i.e., if one of these T moments of time s is in one of the intervals $[\underline{t}_i, \overline{t}_i]$, then we add this moment of time t to the list of moments for which the activity started at this moment will be disrupted.

We can then get the desired probability p by dividing the total number m of such disruptive moments of time by $N-(T-1)$.

Can we compute the desired probability faster. The above computation requires that we try all $N-(T-1)$ moments of time t one by one, and for each of these moments of time t, we test T different moments of time s. Thus, the above straightforward computation requires $\approx N \cdot T$ computational steps.

For large N, this is a huge number. Can we compute the probability faster?

We cannot go faster than $O(n)$. As the input for our computations, we use n intervals $[\underline{t}_i, \overline{t}_i]$, i.e., $2n$ numbers $\underline{t}_i$ and $\overline{t}_i$. We need to process each of these numbers, so any algorithm for computing probabilities must have at least $O(n)$ computation steps.

What we do in this paper. In this paper, we show that we *can* compute the desired probability in time $O(n)$. In other words, we describe an *asymptotically optimal* algorithm for computing the desired probability.

Usually, at most moments of time t, there is no disruption, so $n \ll N$. Thus, our $O(n)$ algorithm is indeed much faster than straightforward computations, which require time $\geq N$.

2 Analysis of the Problem and the Resulting Algorithm

Main idea. We want to count how many moments of time t lead to disruption. Our main idea is that instead of detecting such disruptive moments of time one by one, we will form intervals $[\underline{s}_j, \overline{s}_j]$ that contact all such disruptive moments of time.

We will call such intervals *activity-disrupted intervals*. We will show that there are no more than n such intervals, and that their computation takes time $\leq O(n)$.

Analysis of the problem: first activity-disrupted interval $[\underline{s}_1, \overline{s}_1]$. Let us start with the very first activity-disrupted interval $[\underline{s}_1, \overline{s}_1]$.

The first disruption occurs at moment $\underline{t}_1$. Let us consider two possibilities.

- If $\underline{t}_1 \leq T$, then even the activity starting at the very first moment of time $t = 1$ will be affected, so we have $\underline{s}_1 = 1$.
- If $\underline{t}_1 > T$, then the activity starting at moment $t = 1$ will not be affected. The first affected activity is the one starting at the moment t for which $t + (T-1) = \underline{t}_1$. Thus, in this case, we have $\underline{s}_1 = \underline{t}_1 - (T-1)$.

Both cases can be described by a single formula $\underline{s}_1 = \max(\underline{t}_1 - (T-1), 1)$.

All activities starting at moments $\underline{s}_1, \underline{s}_1 + 1, \ldots$, all the way to $\overline{t}_1$ will be affected; they will be affected by the first disruption interval $[\underline{t}_1, \overline{t}_1]$. So, we can tentatively set $\overline{s}_1 = \overline{t}_1$.

Is the activity starting at the next moment of time $\overline{t}_1 + 1$ affected? It depends on how close is the next disruption interval $[\underline{t}_2, \overline{t}_2]$.

If $\overline{t}_1 + 1 + (T-1) < \underline{t}_2$, then the activity starting at moment $\overline{t}_1 + 1$ is not affected, so upper bound $\overline{s}_1$ of the interval of activity-disrupted moments of time stays at $\overline{s}_1 = \overline{t}_1$, and we start forming the next activity-disruption interval.

On the other hand, if $\overline{t}_1 + 1 + (T-1) \geq \underline{t}_2$, this means that an activity starting at moment $\overline{t}_1 + 1$ is also disrupted – this time by the second disruption interval $[\underline{t}_2, \overline{t}_1]$ – as well as activities starting at all following moments of time $\underline{t}_1 + 2, \ldots, \overline{t}_2$. Thus, the set of activity-disrupted moments of time extends at least to $\overline{t}_2$. In this case, we tentatively set $\overline{s}_1 = \overline{t}_2$.

Whether this is a true bound depends on how close is the next disruption interval $[\underline{t}_3, \overline{t}_3]$. If this interval is not close, i.e., if $\overline{t}_2 + 1 + (T-1) < \underline{t}_3$, then we keep the value $\overline{s}_1 = \overline{t}_2$ and start forming the new activity-disruption interval $[\underline{s}_2, \overline{s}_2]$.

If the next disruption interval is close, i.e., if $\overline{t}_1 + 1 + (T-1) \geq \underline{t}_2$, then we set $\overline{s}_1 = \overline{t}_3$ and consider how close is the next disruption interval, etc.

Analysis continued: following activity-disrupted intervals $[\underline{s}_j, \overline{s}_j]$. When the first activity-disrupted interval $[\underline{s}_1, \overline{s}_1]$ is formed, this means that the next disruption interval $[\underline{t}_i, \overline{t}_i]$ is not close to $\overline{s}_1$.

So, activities disrupted by the i-th disrupted interval form the basis of the second activity-disruption interval. For an activity lasting T moments of time to be affected by the i-th disruption interval, this activity must start no later than the moment $\underline{t}_i - (T-1)$. Thus, we take $\underline{s}_2 = \underline{t}_i - (T-1)$. Tentatively, we set $\overline{s}_2 = \overline{t}_i$, since we know that activities starting at all the moments of time from $\underline{s}_2$ to $\overline{t}_i$ will be affected by the i-th disruption interval.

To find out if this is indeed the end of the second activity-disrupted interval, we check whether the next disruption interval $[\underline{t}_{i+1}, \overline{t}_{i+1}]$ is close or not:

- if the next interval is not close, i.e., if $\overline{t}_i + (T-1) < \underline{t}_{i+1}$, then we keep the value $\overline{s}_1 = \overline{t}_i$ and start forming the new activity-disruption interval.
- If the next disruption interval is close, i.e., if $\overline{t}_i + 1 + (T-1) \geq \underline{t}_{i+1}$, then we set $\overline{s}_1 = \overline{t}_{i+1}$ and consider how close is the next disruption interval, etc.

Similarly, we form the third activity-disruption interval $[\underline{s}_1, \overline{s}_3]$, etc. We stop when we reached the last moment of time $N-(T-1)$ for which our records about the first N moments of time still allow us to check whether the activity starting at this moment of time will be disrupted or not.

From activity-disrupted intervals to the desired probability p. We compute m as the total length $\sum\limits_j (\overline{s}_j - \underline{s}_j)$ of all activity-disrupted intervals, and then we compute the probability p as the desired ratio.

Resulting algorithm. As a result, we arrive at the following algorithm.

$j \leftarrow 1$; done $\leftarrow$ false; $m \leftarrow 0$; $i \leftarrow 1$;
$\underline{s}_1 \leftarrow \max(1, \underline{t}_1 - (T-1))$;
while $((\neg\text{done})\,\&\,(i \le n))$
  if $(\overline{t}_i \ge N-(T-1))$
    $\{\text{done} \leftarrow \text{true}$; $\overline{s}_j \leftarrow N-(T-1);\}$
  else
    $\overline{s}_j \leftarrow \overline{t}_i$; $i \leftarrow i+1$;
    while $((\neg\text{done})\,\&\,(\overline{s}_j + (T-1) \ge \underline{t}_i))$
      if$(\overline{t}_i \ge N-(T-1))$
        $\{\overline{s}_j \leftarrow N-(T-1)$; done $\leftarrow$ true$\}$;
      else
        $\{\overline{s}_j \leftarrow \overline{t}_i$; $i \leftarrow i+1;\}$
    endwhile;
  endif;
  $m \leftarrow m + (\overline{s}_j - \underline{s}_j)$;
  $j \leftarrow j+1$; $\underline{s}_j \leftarrow \underline{t}_i - (T-1)$;
endwhile;
$p \leftarrow m/(N-(T-1))$;

This algorithm indeed requires linear time $O(n)$. For each of n disruption intervals $[\underline{t}_i, \overline{t}_i]$, this algorithm takes a finite number C of computational steps. Thus, the above algorithm indeed requires computation time $\le C \cdot n = O(n)$ – i.e., it is indeed asymptotically optimal.

3 How to Gauge Whether Activity Disruptions are Uniformly Distributed or Mainly Concentrated in Some Time Periods

Need for an additional characteristic of user quality. The above-defined probability p describes how frequently a user will experience disruptions. However, this probability p does not tell us whether these disruptions are mostly located during a small interval of time or they are uniformly spread. For a user, this is important:

- it is one thing to have a small disruption every minute,
- it is a different thing to have flawless activities almost all the time with a very "bumpy" five-minute period.

Which characteristic shall we select? A proposal. In the previous sections, we considered a single probability p corresponding to all moments of time between 1 and $N-(T-1)$.

To capture the above difference, we propose to set a time period S which is larger than the activity duration T but smaller than the overall duration N of our recording. This enables us to consider, instead of a single time period from 1 to N, several different time periods:

- from 1 to S;
- from 2 to $S+1$; ...
- from $N-(S-1)$ to N.

For each interval from t to $t+(S-1)$, we can define the probability $p(t)$ corresponding to this interval – which can be computed, e.g., by the algorithm described in the previous section. Here, $p(t) = m(t)/(S-(T-1))$, where $m(t)$ is the numbers of points s from t to $t+(S-1)-(T-1)$ for which the activity starting at moment s will be disrupted.

If disruptions are uniformly distributed in time, then the probabilities $p(t)$ corresponding to different moments of time t will be approximately the same. If the distribution of disruptions is not uniform, then the values $p(t)$ corresponding to different moments of time t will, in general, deviate from the average probability – some values $p(t)$ will be smaller than the average, some will be larger than the average.

To gauge such deviations, the traditional statistical approach is to use standard deviation $\sigma \stackrel{\text{def}}{=} \sqrt{V}$, where V is the sample variance:

$$V = \frac{1}{N-(S-1)} \sum_{t+1}^{N-(S-1)} (p(t))^2 - \left(\frac{1}{N-(S-1)} \sum_{t=1}^{N-(S-1)} p(t) \right)^2 .$$

How can we compute the corresponding standard deviation? Towards an algorithm. To compute this characteristic, it is sufficient to compute the sums $\sum\limits_t (p(t))^2$ and $\sum\limits_t p(t)$. Since $p(t)$ is proportional to $m(t)$, this is equivalent to computing the sums $M_2 \stackrel{\text{def}}{=} \sum\limits_t (m(t))^2$ and $S_1 \stackrel{\text{def}}{=} \sum\limits_t m(t)$.

When we go from moment t to moment $t+1$, i.e., when we replace the interval $[t, t+(S-1)]$ to the interval $[t+1, (t+1)+(S-1)]$, then we can describe the difference between $m(t)$ and $m(t+1)$ as follows:

If both moments of time t and $(t+1)+(S-1)$ belong to some activity-disrupted interval(s), then, when we go from the interval $[t, t+(S-1)]$ to the interval $[t+1, (t+1)+(S-1)]$, we lose one disruption moment of time (t) but gain another one $((t+1)+(S-1)$. Thus, in this case, $m(t+1) = m(t)$.

If neither of the moments t and $(t+1)+(S-1)$ belong to any activity-disrupted interval, then we also have $m(t+1) = m(t)$.

If the moment of time t belongs to some activity-disrupted interval, while the moment $(t+1)+(S-1)$ does not belong to any activity-disrupted interval, then, when we go from the interval $[t, t+(S-1)]$ to the interval $[t+1, (t+1)+(S-1)]$, we lose one disruption moment of time (t). Thus, in this case, $m(t+1) = m(t)-1$.

If the moment of time t does not belong to any activity-disrupted interval, while the moment $(t+1)+(S-1)$ belongs to some activity-disrupted interval, then, when we go from the interval $[t, t+(S-1)]$ to the interval $[t+1, (t+1)+(S-1)]$, we gain one disruption moment of time $((t+1)+(S-1))$. Thus, in this case, $m(t+1) = m(t)+1$.

For each t, we thus have one of the four possible cases. As we consider possible values $t = 1, 2, \ldots$, the case changes in one of the following cases:

- the first case if when t is still in an activity-disrupted interval, while $t+1$ is not; this happens if $t = \overline{s}_j$ for some j;
- the second case if when t is not in any activity-disrupted interval, but $t+1$ is; this happens if $t+1 = \underline{s}_j$ for some j, i.e., if $t = \underline{s}_j - 1$;
- the third case if when $(t+1)+(S-1) = t+S$ is still in an activity-disrupted interval, while $t+S+1$ is not; this happens if $t+S = \overline{s}_j$ for some j, i.e., if $t = \overline{s}_j - S$;
- the fourth case if when $t+S$ is not in any activity-disrupted interval, but $t+S+1$ is; this happens if $t+S+1 = \underline{s}_j$, i.e., if $t_j = \underline{s}_j - S - 1$.

Thus, to compute the sums M_1 and M_2, it is sufficient to consider $4n$ points $\overline{s}_j$, $\underline{s}_j - 1$, $\overline{s}_j - S$, and $\underline{s}_j - S - 1$. We can sort these points by merging the 4 sorted lists corresponding to points of each of the four types.

Let us denote the sorted points by $s_1, \ldots, s_{4n}$. To cover the whole zone from 1 to $N-(S-1)$, we can supplement this list with $s_0 \stackrel{\text{def}}{=} 0$ and $s_{4n+1} \stackrel{\text{def}}{=} N-(S-1)$. Then, each moment of time from 1 to $N-(S-1)$ is covered by one of the intervals $[s_k+1, s_{k+1}]$. Each of the sums M_1 and M_2 can thus be computed by computing the sum over each of these intervals, and then adding these sums together: $M_\ell = \sum\limits_{k=0}^{4n} M_\ell(k)$, where $M_1(k) \stackrel{\text{def}}{=} \sum\limits_{t=s_k+1}^{s_{k+1}} m(t)$ and $M_2(k) \stackrel{\text{def}}{=} \sum\limits_{t=s_k+1}^{s_{k+1}} (m(t))^2$.

On each of the intervals $[s_k+1, s_{k+1}]$, the values $m(t)$ either remain constant or change by ± 1. If the values $m(t)$ do not change, then computing the corresponding sums is easy: $M_1(k) = m(t) \cdot (s_{k+1} - s_k)$ and $M_2(k) = (m(t))^2 \cdot (s_{k+1} - s_k)$.

If the values $m(t)$ change, this means that the corresponding values $m(t)$ cover all the integers from $m(s_k+1)$ to $m(s_{k+1})$. To compute the corresponding sums $M_1(k)$ and $M_2(k)$, it is therefore sufficient to be able to compute, for every two integers $a \le b$, the sums $\sum\limits_{i=a}^{b} i$ and $\sum\limits_{i=a}^{b} i^2$. These sums can be computed based on the known formulas

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2} \quad \text{and} \quad \sum_{i=1}^{n} i^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$$

if we take into account that for each x_i, we have $\sum\limits_{i=a}^{b} x_i = \sum\limits_{i=1}^{b} x_i - \sum\limits_{i=1}^{a-1} x_i$.

Thus, we arrive at the following algorithm.

Resulting algorithm. First, we apply the algorithm from the previous section and compute the values $\underline{s}_j$ and $\overline{s}_j$.

Now, we form four increasing sequences: $\overline{s}_j$, $\underline{s}_j - 1$, $\overline{s}_j - S$, and $\underline{s}_j - S - 1$. We merge these sequences into a single sorted sequence whose elements will be noted by $s_1, \ldots, s_{4n}$, and we add $s_0 = 0$ and $s_{4n+1} = N - (S-1)$. For each k from 0 to $4n$, we use the algorithm from Part 1 to compute the values $m(s_k+1)$ and $m(s_{k+1})$.

If $m(s_k+1) = m(s_{k+1})$, then we compute $M_1(k) = m(t_k+1) \cdot (s_{k+1} - s_k)$ and $M_2(k) = (m(t_k+1))^2 \cdot (s_{k+1} - s_k)$.

If $m(s_k+1) \ne m(s_{k+1})$, then we take $z \stackrel{\text{def}}{=} \min(m(s_k+1), m(s_{k+1})) - 1$, $b \stackrel{\text{def}}{=} \max(m(s_k+1), m(s_{k+1}))$, and compute

$$M_1(k) = \frac{b \cdot (b+1)}{2} - \frac{z \cdot (z+1)}{2}; \quad M_2(k) = \frac{b \cdot (b+1) \cdot (2b+1)}{6} - \frac{z \cdot (z+1) \cdot (2z+1)}{6}.$$

After this, we compute $M_1 = \sum\limits_{k=0}^{4n} M_1(k)$ and $M_2 = \sum\limits_{k=0}^{4n} M_2(k)$. Based on these values, we compute

$$V = \frac{M_2}{(N-(S-1)) \cdot (S-(T-1))^2} - \left(\frac{M_1}{(N-(S-1)) \cdot (S-(T-1))} \right)^2$$

and $\sigma = \sqrt{V}$.

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

References

[1] Bacon, D.F., Cheng, P., and D. Grove, Garbage collection for embedded systems, *Proceedings of the 4th ACM International Conference on Embedded Software EMSOFT'04*, pp.125–136, 2004.

[2] Cardone, G., Cirri, A., Corradi, A., Foschini, L., and D. Maio, MSF: an efficient mobile phone sensing framework, *International Journal of Distributed Sensor Networks*, vol.2013, article ID 538937, 2013.

[3] Nielsen, J., *Usability Engineering*, Academic Press, Cambridge, Masachusetts, 1993.

[4] Sachindran, N., Moss, J.E.B., and E.D. Berger, MC2: high performance garbage collection for memory constrained environments, *Proceedings of the ACM Conference on Object-Oriented Systems, Languages and Applications OOPSLA'04*, pp.81–98, 2004.