Optimal Restart Times for Moments of Completion Time

Aad van Moorsel[1] and Katinka Wolter[2]

[1] University of Newcastle

School of Computing Science

Newcastle upon Tyne, NE1 7RU, United Kingdom

aad.vanmoorsel@newcastle.ac.uk

[2] Humboldt-Universität zu Berlin,

Institut für Informatik,

Unter den Linden 6, 10099 Berlin, Germany

wolter@informatik.hu-berlin.de

Abstract. Restart is an application-level technique that speeds up jobs with highly variable completion times. In this paper, we present an efficient iterative algorithm to determine the restart strategy that minimises higher moments of completion time, when the total number of restarts is finite. We demonstrate its computational efficiency in comparison with alternative algorithms. We also discuss fast approximations to determine close to optimal restart times for limiting cases.

1 Introduction

The work in this paper applies to various forms of 'restart,' which means that a task is retried after some time threshold has passed. Restart finds its application in areas ranging from randomised algorithms [3] to distributed data base queries [7], Internet agents [4] and software rejuvenation [2]. Probably the most widely exercised restart mechanism is clicking the reload button of the web browser when a download takes too long.

The basic form of restart can be conveniently modelled assuming that (1) successive downloads are statistically independent and identically distributed, and (2) new tries abort previous tries. Such model assumptions have been found realistic for Internet applications [6], and form the basis

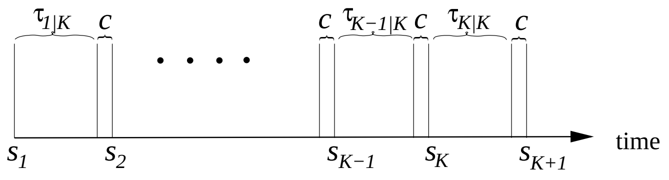

Fig. 1. Labelling restarts, total of K restarts.

for the analysis in [1, 3, 4] for a variety of restart applications. Interesting enough, even though rejuvenation and preventive maintenance techniques can be cast as restart methods, they have always been analysed using relatively intricate models with detailed state information. We believe that in very 'random' environments our simple model applies, as we discuss at length in [5].

Under the stated assumptions, we provide in this paper an efficient algorithm to determine the optimal time instants at which to initiate restarts, so that higher moments of completion time are minimised. As we showed in [5], the optimal restart strategy for the first moment can be determined in straightforward manner, both for finite and infinite number of allowed restarts. However, determining restart times that minimise higher moments of completion time is considerably more challenging, and requires an iterative approach to deal with the multiple dimensions of the optimisation problem. Our proposed algorithm leverages various expressions for the moments of completion time to simplify the minimisation problem. Because of this simplification, the algorithm outperforms more naive approaches by up to an order of magnitude, as we will show.

From our analysis it follows that it is typically not optimal to apply restarts at constant intervals when minimising higher moments, even if an infinite number of restarts is allowed. This is in contrast to the situation for the first moment, as we explain. We also provide insights into the characteristics of the optimal restart strategy through approximations under limiting conditions. It turns out that as long as enough restarts are allowed, one can use a first-moment approximation, with appropriate corrections for the first and last few restarts. Such approximations enable quick estimates for optimal restart times, and are therefore of practical importance.

2 Model

Let the random variable T represent the completion time of a job without restarts, $f(t)$ its probability density function, and $F(t)$ its distribution. The total number of restarts is K, and the overhead associated with restarting is c time units for each restart. The random variable T_K (with density $f_K(t)$ and distribution $F_K(t)$) represents the completion time with K restarts, where the restart intervals have length $\tau_{1|K}, \ldots, \tau_{K|K}$, as shown in Figure 1. The k-th interval starts at time s_k, that is, $s_1 = 0$, $s_2 = \tau_{1|K} + c, \ldots, s_{K+1} = \sum_{k=1}^{K} \tau_{k|K} + Kc$.

Setting $\tau_{K+1|K} = \infty$ for notational purposes, the density function $f_K(t)$ and survival function $\bar{F}_K(t) = 1 - F_K(t)$ depend on which restart interval t falls in, as follows [5]:

$$f_K(t) = \begin{cases} \prod_{i=1}^{k-1} \bar{F}(\tau_{i|K}) f(t - s_k) & \text{if} \quad s_k \leq t < s_k + \tau_{k|K}, \quad k = 1, \ldots, K+1 \\ 0 & \text{if} \quad s_k + \tau_{k|K} \leq t < s_{k+1}, \quad k = 1, \ldots, K \end{cases} \tag{1}$$

$$\bar{F}_K(t) = \begin{cases} \prod_{i=1}^{k-1} \bar{F}(\tau_{i|K}) \bar{F}(t - s_k) & \text{if} \quad s_k \leq t < s_k + \tau_{k|K}, \quad k = 1, \ldots, K+1 \\ \prod_{i=1}^{k} \bar{F}(\tau_{i|K}) & \text{if} \quad s_k + \tau_{k|K} \leq t < s_{k+1}, \quad k = 1, \ldots, K \end{cases}$$

The N-th moment $E[T_K^N]$, our metric of interest, is by definition:

$$E[T_K^N] = \int_0^{\infty} t^N f_K(t)dt = \sum_{k=1}^{K+1} \int_{s_k}^{s_k + \tau_{k|K}} t^N f_K(t)dt. \tag{2}$$

3 Optimisation

To find the restart times $\tau_{1|K}, \ldots, \tau_{K|K}$ that minimise $E[T_K^N]$, one could minimise (2) directly. It results in a K-dimensional minimisation problem that can be solved with off-the-shelf optimisation software. However, it is computationally expensive, since every new 'guess' for $\tau_{k|K}$ implies recomputing the integral term in (2) for all intervals $[s_l, s_l + \tau_{l|K})$ with $l \geq k$, to determine if the guess improves $E[T_K^N]$.

As an alternative, we can use an expression for $E[T_K^N]$ we derived in [5]. We present this expression here in slightly different form, using the notation $\bar{k} = K - k + 1$. It is important to grasp the intuitive meaning of $\bar{k}$: where k is the number of restarts preceding and including the

k-th, $\bar{k}$ is the number of restarts succeeding and including the k-th. We then recursively relate moments for $\bar{k}$ restarts with that for $\bar{k}-1$ restarts by adding one restart before the existing $\bar{k}-1$:

$$E[T_{\bar{k}}^N] = M_k[T^N] + \bar{F}(\tau_{k|K}) \sum_{n=0}^{N} \binom{N}{n} (\tau_{k|K} + c)^{N-n} E[T_{\bar{k}-1}^n], \tag{3}$$

where $M_k[T^N]$ denotes the 'partial moment,' defined for $\bar{k} = 1, \ldots, K$, as:

$$M_k[T^N] = \int_0^{\tau_{k|K}} t^N f(t) dt.$$

Instead of minimising (2) one can minimise (3). In this case, however, every new 'guess' for $\tau_{k|K}$ implies computing $E[T_K^N]$ 'all the way,' recursively calculating $E[T_l^N]$ for all values $l \geq k$, to determine if the guess improves $E[T_K^N]$. This also introduces much computational overhead. (In Figure 4 we will see that minimising (3) is in fact slightly less expensive than minimising (2), at least for the discussed example.)

The main idea in this paper is to not minimise (2) or (3) directly, but instead extend to higher moments an idea that worked very well in [5] for the first moment. Utilising the recursion of (3), [5] presents an algorithm that sequentially determines the restart time $\tau_{k|K}$ that minimises $E[T_{\bar{k}}] = M_k[T] + \bar{F}(\tau_{k|K})(\tau_{k|K} + c + E[T_{\bar{k}-1}])$, for $\bar{k} = 1$ to K. Its correctness relies on the fact that the optimal restart time $\tau_{k|K}$ is independent of preceding restarts, as we discuss in detail below. We named this algorithm the backward algorithm, since it determines the best restart times in reversed order, that is, first $\tau_{K|K}$, then $\tau_{K-1|K}$ until finally $\tau_{1|K}$. A single pass of K optimisations is guaranteed to provide the optimal restart times, which makes the backward algorithm computationally much more efficient than minimising either (2) or (3). (Figure 4 shows an improvement of about a factor 20.)

The backward algorithm can not be applied to higher moments, because the optimal value of a restart interval depends on the restarts that precede it. To resolve this problem, we now prove that the optimal restart time at interval k depends on preceding restart times solely through the sum of these restart times, not the individual values. That is, the optimal $\tau_{k|K}$ depends on $\tau_{1|K}, \ldots, \tau_{k-1|K}$ only through the value of s_k. Based on this, we obtain a new expression (namely expression (4)), which we combine with (3) to simplify the optimisation task.

Theorem 1. *For any strictly positive $k \leq K$, let the first $k-1$ restart times $\tau_{1|K}, \ldots, \tau_{k-1|K}$ be given. The last $\bar{k}$ restart times $\tau_{k|K}, \ldots, \tau_{K|K}$ minimise $E[T_K^N]$ if and only if they minimise $E[T_{\bar{k}} + s_k)^N]$ (where we equate restart $\tau_{k+i-1|K}$ in T_K with $\tau_{i|\bar{k}}$ in $T_{\bar{k}}$, for $i = 1, \ldots, \bar{k}$).*

Proof. First, by definition:

$$E[T_K^N] = \int_0^{s_k} t^N f_K(t)dt + \int_{s_k}^{\infty} t^N f_K(t)dt.$$

Since the left most integral term does not depend on $\tau_{k|K}, \ldots, \tau_{K|K}$, the last $\bar{k}$ optimal restart times minimise $E[T_K^N]$ if and only if they minimise $\int_{s_k}^{\infty} t^N f_K(t)dt$. If we equate $\tau_{k+i-1|K}$ with $\tau_{i|\bar{k}}$ for $i = 1, \ldots, \bar{k}$, we know from (1) that for any $t \geq 0$:

$$f_K(t + s_k) = \prod_{l=1}^{k-1} \bar{F}(\tau_{l|K}) f_{\bar{k}}(t).$$

This implies that:

$$\int_{s_k}^{\infty} t^N f_K(t)dt = \int_0^{\infty} (t + s_k)^N f_K(t + s_k)dt =$$
$$\prod_{l=1}^{k-1} \bar{F}(\tau_{l|K}) \int_0^{\infty} (t + s_k)^N f_{\bar{k}}(t)dt = \prod_{l=1}^{k-1} \bar{F}(\tau_{l|K}) E[(T_{\bar{k}} + s_k)^N].$$

The product in this expression is independent of $\tau_{k|K}, \ldots, \tau_{K|K}$, and therefore minimising $\int_{s_k}^{\infty} t^N f_K(t)dt$ (and thus $E[T_K^N]$) corresponds to minimising $E[(T_{\bar{k}} + s_k)^N]$ (with $\tau_{k+i-1|K} = \tau_{i|\bar{k}}$, $i = 1, \ldots, \bar{k}$).

Theorem 1 implies that for any k, $k = 1, \ldots, K$, determining the optimal restart time $\tau_{k|K}$ corresponds to minimising:

$$E[(T_{\bar{k}} + s_k)^N] = \sum_{n=0}^{N} \binom{N}{n} s_k^{N-n} E[T_{\bar{k}}^n], \tag{4}$$

where $E[T_{\bar{k}}^n]$ obeys (3):

$$E[T_{\bar{k}}^n] = M_k[T^n] + \bar{F}(\tau_{k|K}) \sum_{m=0}^{n} \binom{n}{m} (\tau_{k|K} + c)^{n-m} E[T_{\bar{k}-1}^m]. \tag{5}$$

We are now in a position to show that for the first moment, the optimal value for $\tau_{k|K}$ does not depend on earlier restarts, not even through their sum s_k. For that reason, the backward algorithm works correctly for the first moment. We present this as a corollary of Theorem 1.

Corollary 1. *The restart times $\tau_{k|K}, \ldots, \tau_{K|K}$ minimise $E[T_K]$ if and only if they minimise $E[T_{\bar{k}}]$ (with $\tau_{k+i-1|K} = \tau_{i|\bar{k}}$, $i = 1, \ldots, \bar{k}$).*

This corollary follows from the fact that Theorem 1 states for N=1 that minimising $E[T_K]$ corresponds to minimising $E[T_{\bar{k}} + s_k]$. Obviously, $E[T_{\bar{k}} + s_k] = E[T_{\bar{k}}] + s_k$, and since s_k is a constant, it does not influence the optimisation solution. Therefore minimising $E[T_K]$ corresponds to minimising $E[T_{\bar{k}}]$.

We will see in Figure 4 in Section 6 that (for our example) it saves about 70 percent computation time to minimise the second or third moment of completion time using (4) and (5). The reason for this speed-up is that with every 'guess' of $\tau_{k|K}$ when minimising (4) using (5), we only recompute $E[T^n_{\bar{k}}]$. That is, the algorithm neither requires to recompute integral terms for all values $l \geq k$ (as in (2)), nor terms $E[T^N_l]$ for all values $l \geq k$ (as in (3)). Effectively, we have isolated the optimisation of the k-th restart time from interference with the other restart intervals.

4 Algorithm

The resulting optimisation algorithm is given as Algorithm 1. Contrary to the backward algorithm for the first moment [5] it does not terminate in K steps, but requires to iterate until convergence (of either $E[T^N_K]$ or the restart times). One can apply generic approaches to decide which restart time τ_k to optimise at each iteration (such as the method of steepest descent). However, we propose three particular ways, which try to leverage the structure of the problem: backward, forward and alternating.

Algorithm 1 (Backward, Forward and Alternating Optimisation)

```
Input constants $N$ and $K$;
Input boolean alternating;
Set either boolean forward or backward to TRUE;
Determine $\tau_\infty$ that minimises $E[T_\infty]$;
For $n = 1$ to $N$ {
    Compute and Set $E[T_0^n]$ (moments without restarts);
    For $k = K$ to 1
        Initialise $E[T_k^n]$ using (5) with $\tau_{k|K} = \tau_\infty$;
}
While( not converged ) Do {
    If( backward ) then {
        For $k = K$ to 1 {
            Find $\tau_{k|K}$ that minimises (4), using (5) for $E[T_k^n]$;
            For $n = 1$ to $N$
                Update $E[T_k^n]$ using (5) with new value of $\tau_{k|K}$;
        }
    If( forward ) then {
        For $k = 1$ to $K$
            Find $\tau_{k|K}$ that minimises (4), using (5) for $E[T_k^n]$;
            Update $s_{k+1}$ with new value of $\tau_{k|K}$;
        }
        For $k = K$ to 1 {
            For $n = 1$ to $N$
                Update $E[T_k^n]$ using (5) with new value of $\tau_{k|K}$;
        }
    }
    If( alternating ) then swap backward and forward
}
Return $\tau_{1|K}, \ldots, \tau_{K|K}$;
```

Each minimisation step in the algorithm can be carried out with any desired general-purpose optimisation routine. Also, note that at initialisation, $E[T_\infty]$ can be minimised using the expression

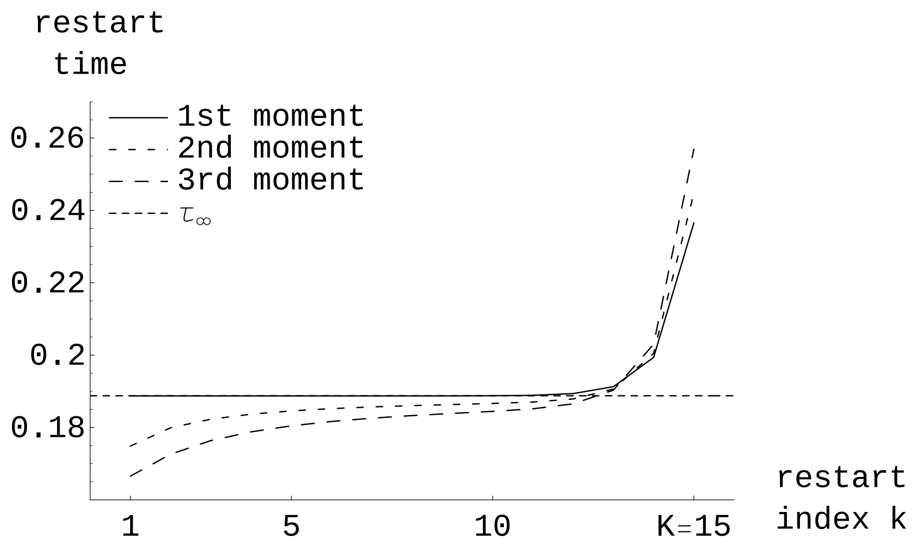

Fig. 2. Optimal restart times, with respect to the moments $E[T_{15}]$, $E[T_{15}^2]$ and $E[T_{15}^3]$, respectively.

$E[T_\infty] = (M_1[T] + \bar{F}(\tau_\infty)(\tau_\infty + c))/F(\tau_\infty)$ derived in [5]. The reason to initialise the algorithm with τ_∞ will become apparent when discussing the bulk approximation in the next section.

5 Characteristics of Optimal Restart Times

We applied our algorithm to the case that the completion time T has a lognormal distribution, with parameters $\mu = -2.31$ and $\sigma = 0.97$.[3] We determine $K = 15$ restart times that minimise the first, second and third moment of the completion time. These restart times (with an interpolating curve) are shown in Figure 2. The figure also shows τ_∞, which is the starting solution set at the initialisation step in Algorithm 1.

Figure 2 indicates that when minimising the first moment, the optimal restart time $\tau_{k|K}$ monotonically converges when k gets smaller, to a single optimum τ_∞, provided K is large enough. In fact, as we observed in [5], the backward algorithm is a fixed-point algorithm, with associated convergence properties. This also implies that if an infinite number of restarts is allowed, a constant restart time is optimal, as has been observed in [1, 3].

[3] For the current paper, there is no particular significance to the chosen parameter values. They happen to be the parameters of a lognormal fit for experimental data of HTTP GET completion times [6].

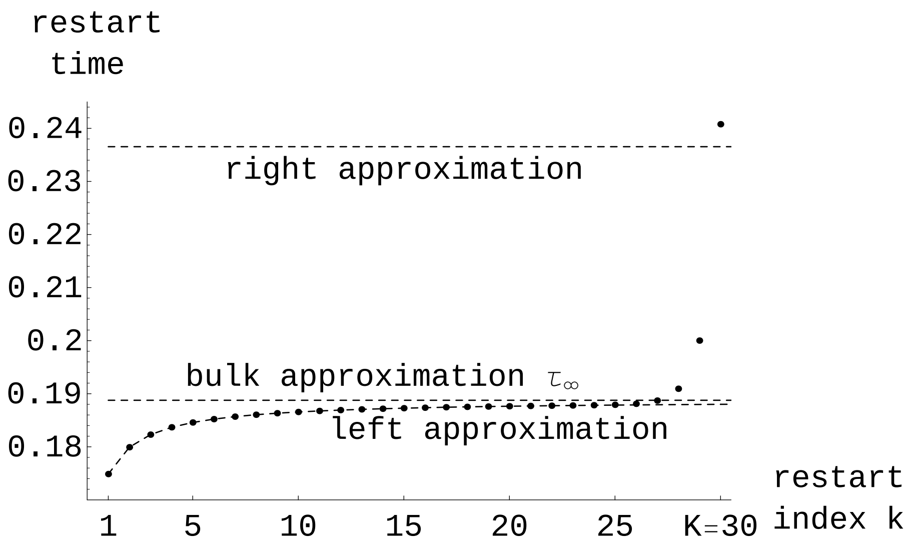

Fig. 3. The dots give restart times that minimise the second moment $E[T_{30}^2]$, the dashed lines are the approximations, as labelled.

The convergence behaviour for higher moments is not as straightforward, as also witnessed by Figure 2. Nevertheless, there are some interesting insights to be gained from explaining the more intricate convergence patterns.

The key observation is that if the number of restarts increases, the dominant term when minimising (4) involves only the first moment. Therefore the restart times that minimise the first moment are a good strategy for higher moments as well (provided K is large, and not considering the first and last few restart times). To make this more precise, assume that $k \to \infty, k \leq K$, in which case, apart from pathological cases, it must be that $s_k \to \infty$. This allows us to approximate expression (4) by the first two terms of its sum:

$$\lim_{k\to\infty} E[(T_{\bar{k}} + s_k)^N] \approx s_k^N + N s_k^{N-1} E[T_{\bar{k}}]. \tag{6}$$

Since s_k^N and Ns_K^{N-1} are constants, finding the restart times that minimise (4) is approximately equal to finding the restart times that minimise the first moment $E[T_{\bar{k}}]$. Based on this, we introduce three limiting cases, namely at the right boundary ($\tau_{k|K}$ for $k \to \infty$, and $\bar{k} \downarrow 1$), middle or 'bulk' ($\tau_{k|K}$ for $k \to \infty$ and $\bar{k} \to \infty$), and left boundary ($\tau_{k|K}$ for $k \downarrow 1$ and $\bar{k} \to \infty$). Figure 3 illustrates the main results.

Right boundary approximation. For $k \to \infty$ and $\bar{k} = 1$ the first-moment approximation of $\tau_{k|K}$ corresponds to finding the restart time that minimises $E[T_1]$ (only one restart allowed). Figure 3 shows the right approximation and we see that for $K = 30$ it is reasonable but not exceptionally close to the actual optimal restart (given by the dot). We can extend the approximations to value $\bar{k} = 2, 3, \ldots$, which results in restart times identical to those shown in Figure 2 (the curve for the first moment).

Bulk approximation τ_∞. At the 'bulk,' or the middle of the pack, we get a limiting result if both k and $\bar{k}$ go to infinity. The approximation using (6) results in optimising $E[T_\infty]$, i.e., in restart times equal to τ_∞. In Figure 3, the bulk approximation τ_∞ is indeed close to the optimal restart times. This also explains why we chose τ_∞ during initialisation in Algorithm 1: it is close to optimal for the bulk of restarts.

Left boundary approximation. At the left boundary, we can not simply apply the approximation suggested by equation (6), because k does not tend to infinity. However, we can obtain an approximation for $\tau_{k|K}$ with $k \downarrow 1$ and $\bar{k} \to \infty$, by assuming that the completion time T is distributed as T_∞ with restart interval τ_∞. This approximation is remarkably close, as seen from Figure 3. In fact, other experiments indicate that the left boundary approximation is very close irrespective of the value of K. This implies that if we allow an infinite number of restarts, we can use the left boundary approximation to determine early restarts, until it is close enough to the bulk approximation (which we would use from then on).

In conclusion, we find for the first moment of completion time that the optimal restart strategy is a constant restart time for all restarts, provided we allow an infinite number of restarts. If only a finite number of K restarts is allowed, we can optimise these using the backward algorithm, which terminates in K steps. When we consider higher moments, a constant restart time is typically not optimal, not even if we allow infinitely many restarts. Instead, we need the backward/forward iterative algorithm to compute optimal restart times for the finite case, and use the bulk and left boundary approximation for the case with infinitely many restarts.

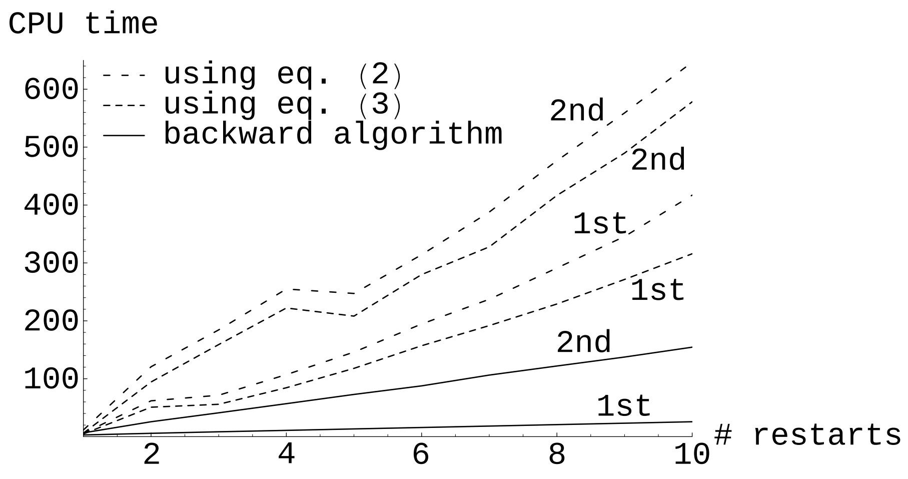

Fig. 4. CPU time used for different algorithms, applied to minimise first as well as second moment.

6 Computational Effort

In Figure 4 we plot the time used for three different methods: Algorithm 1 (backward), minimising expression (2), and minimising expression (3). In all cases we applied default minimisation algorithms in Mathematica to carry out the respective optimisation steps. Algorithm 1 outperforms the other approaches. For the first moment, the speed up is an order of magnitude (about a factor 20), which finds its explanation in assured convergence in K steps of the backward algorithm. For the higher moments, the speed up is about a factor 3 or 4. Apparently, the arguments put forward in Section 3 hold correct. We note that we tuned our Mathematica program to the best of our abilities, memorising in-between results so that the recursion in (3) and repetitive computation in (2) are handled as efficient as possible. We also set the convergence criterion identical for all three experiments, (based on convergence of $E[T_K^N]$). Hence, although we do not have access to the 'internals' of Mathematica's optimisation algorithm, we are reasonably confident that the comparison of the three approaches is fair.

Figure 5 compares three versions of Algorithm 1: backward, forward and alternating. These three exhibit similar performance. For our example, the forward algorithm turns out to require

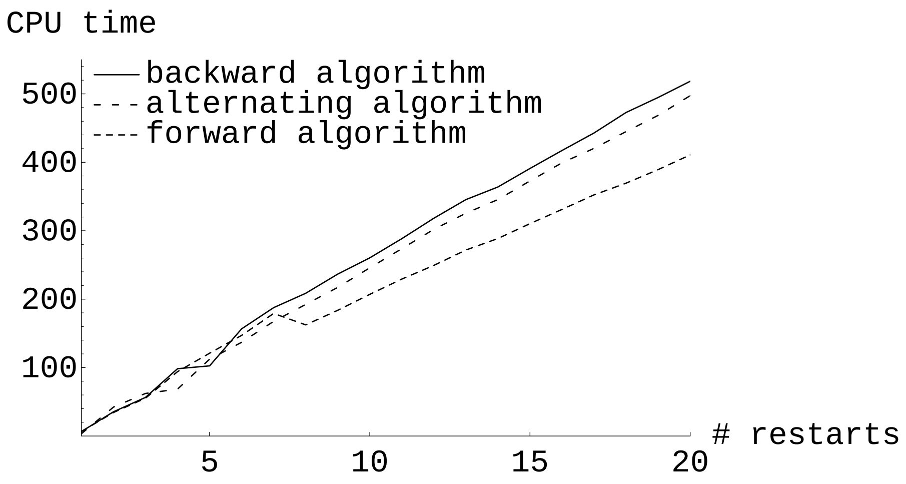

Fig. 5. CPU time used by three versions of algorithm.

one pass less through all restart times than the other two algorithms, and hence it takes less CPU time.[4]

Typically, we require not more than 5 passes through the K restart times, irrespective of the value of K. Studying the complexity of our Mathematica implementation, it turns out that running the optimisation routine is the computationally most expensive part: at step k, optimisation of $\tau_{k|K}$ takes an order of magnitude more time than the computation that updates $E[T_{\bar{k}}^N]$. Algorithm 1 uses backward and/or forward traversal through the K restart times to compute $E[T_{\bar{k}}^N]$ efficiently, but following the above reasoning, it may be more important to decrease the number of calls to the optimisation routine. Algorithm 1 may therefore be further improved by choosing the order in which to optimise restart times based on criteria such as steepest descent. This requires more experimentation.

[4] Note that because of the workings of the Mathematica optimisation algorithm, the comparison in Figure 4 had to be based on convergence of $E[T_K^N]$ as stopping criterion, while we were able to base the results for the backward/forward algorithm in Figure 5 on convergence of restart times, a stricter criterion. This explains the higher CPU usage for the backward algorithm in Figure 5 compared to Figure 4.

7 Conclusion

This paper presents an algorithm that determines restart times that minimise higher moments of completion time. This problem is particularly complicated because, contrary to minimisation of the first moment solved in [5], optimal restart times for higher moments depend not only on succeeding restarts, but also on preceding restarts. As a consequence, we resorted to an iterative algorithm. We compared our algorithm with 'naive' off-the-shelf approaches, and showed that our algorithm is up to an order of magnitude faster. We also studied restart times under limiting conditions, and derived from this fast approximations of optimal restart times.

References

1. H. Alt, L. Guibas, K. Mehlhorn, R. Karp and A. Wigderson, "A Method for Obtaining Randomized Algorithms with Small Tail Probabilities," *Algorithmica,* Vol. 16, Nr. 4/5, pp. 543–547, 1996.
2. A. Bobbio, S. Garg, M. Gribaudo, A. Horvath, M. Sereno and M. Telek, "Modeling Software Systems with Rejuvenation, Restoration and Checkpointing through Fluid Stochastic Petri Nets," in *Proceedings of PNPM '99,* Zaragoza, Spain, pp. 82-91, IEEE CS Press, Sept 1999.
3. M. Luby, A. Sinclair and D. Zuckerman, "Optimal Speedup of Las Vegas Algorithms," *Israel Symposium on Theory of Computing Systems,* pp. 128–133, 1993.
4. S. M. Maurer and B. A. Huberman, "Restart strategies and Internet congestion," in *Journal of Economic Dynamics and Control,* vol. 25, pp. 641–654, 2001.
5. A. van Moorsel and K. Wolter, "Analysis and Algorithms for Restart," accepted for publication in *First International Conference on Quantitative Evaluation of Systems*, Twente, The Netherlands, Sept. 2004.
6. P. Reinecke, A. van Moorsel and K. Wolter, "A Measurement Study of the Interplay between Application Level Restart and Transport Protocol," *Service Availability Forum*, Munich, May 2004.
7. Y. Ruan, E. Horvitz and H. Kautz, "Restart Policies with Dependence among Runs: A Dynamic Programming Approach," in *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming,* Ithaca, NY, Sept. 2002.