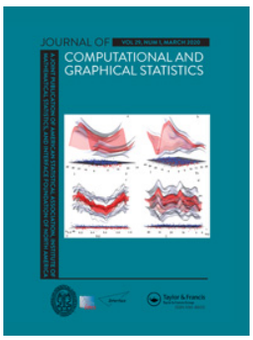

Journal of Computational and Graphical Statistics

ISSN: 1061-8600 (Print) 1537-2715 (Online) Journal homepage: https://www.tandfonline.com/loi/ucgs20

Parallelization of a Common Changepoint Detection Method

S. O. Tickle, I. A. Eckley, P. Fearnhead & K. Haynes

To cite this article: S. O. Tickle, I. A. Eckley, P. Fearnhead & K. Haynes (2020) Parallelization of a Common Changepoint Detection Method, Journal of Computational and Graphical Statistics, 29:1, 149-161, DOI: 10.1080/10618600.2019.1647216

To link to this article: https://doi.org/10.1080/10618600.2019.1647216

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2020, VOL. 29, NO. 1, 149–161
https://doi.org/10.1080/10618600.2019.1647216

Taylor & Francis
Taylor & Francis Group

Parallelization of a Common Changepoint Detection Method

S. O. Tickle[a], I. A. Eckley[b], P. Fearnhead[b], and K. Haynes[a]

[a]STOR-i Centre for Doctoral Training, Lancaster University, Lancaster, UK; [b]Department of Mathematics and Statistics, Lancaster University, Lancaster, UK

ABSTRACT

In recent years, various means of efficiently detecting changepoints have been proposed, with one popular approach involving minimizing a penalized cost function using dynamic programming. In some situations, these algorithms can have an expected computational cost that is linear in the number of data points; however, the worst case cost remains quadratic. We introduce two means of improving the computational performance of these methods, both based on parallelizing the dynamic programming approach. We establish that parallelization can give substantial computational improvements: in some situations the computational cost decreases roughly quadratically in the number of cores used. These parallel implementations are no longer guaranteed to find the true minimum of the penalized cost; however, we show that they retain the same asymptotic guarantees in terms of their accuracy in estimating the number and location of the changes. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received June 2018
Revised May 2019

KEYWORDS
Changepoint detection; Dynamic programming; Parallelization; PELT

1. Introduction

The challenge of changepoint detection has received considerable interest in recent years (see, e.g., Rigaill, Lebarbier, and Robin 2012; Chen and Nkurunziza 2017; Truong, Oudre, and Vayatis 2018, and references therein). There are many algorithms for estimating the number and location of changepoints, for example, binary segmentation, due to Scott and Knott (1974), and its variants such as circular binary segmentation, wild binary segmentation (WBS), and narrowest-over-threshold, due to Olshen et al. (2004), Fryzlewicz (2014), and Baranowski, Chen, and Fryzlewicz (2018), respectively; and dynamic programming approaches that minimize a penalized cost, such as the optimal partitioning procedure of Jackson et al. (2005) or the pruned exact linear time (PELT) method of Killick, Fearnhead, and Eckley (2012).

In many applications, there are computational constraints that can affect the choice of method. We are interested in whether parallel computing techniques can be used to speed up algorithms such as optimal partitioning or PELT. The application of parallelization is vast, with use in such areas as meta-heuristics, cloud computing, and biomolecular simulation, as discussed in Alba (2005), Mezmaz et al. (2011), Schmid et al. (2012), and Wang and Dunson (2014), among many others. Some methods are more easily parallelizable in that it is plain how to split a search space or other task between different nodes. These problems are often described as "embarrassingly parallel." For the changepoint detection problem, binary segmentation and WBS may be described as such. However, it is not so straightforward to parallelize dynamic programming methods.

This article makes a new contribution to this area by suggesting two new approaches for parallelizing a penalized cost approach. In particular, we demonstrate in Section 3 that the computational cost of dynamic programming algorithms that minimize the penalized cost, such as PELT, can be reduced by a factor that can be quadratic in the number of computer cores. Further, we demonstrate empirically that super-linear gains in speed are achievable even in reasonably small sample settings in Section 4. One disadvantage with parallelizing an algorithm such as PELT is that we are no longer guaranteed to be finding the segmentation which minimizes the penalized cost. However, these approximations do not affect the asymptotic properties of the estimator of the number and locations of the changepoints: in Section 3 we show that, for the change-in-mean problem, our proposed approaches retain the same asymptotic properties as PELT.

The changepoint problem considers the analysis of a data sequence, $y_1, \ldots, y_n$, which is ordered by some index, such as time or position along a chromosome. We use the notation $y_{s:t} = (y_s, \ldots, y_t)$ for $t \geq s$. Our interest is in segmenting the data into consecutive regions. Such a segmentation can be defined by the changepoints, $0 = \tau_0 < \tau_1 < \cdots < \tau_m < \tau_{m+1} = n$, where the set of changepoints splits the data into $m+1$ segments, with the jth segment containing data-points $y_{\tau_{j-1}+1:\tau_j}$.

As mentioned, we focus on a class of methods which involve finding the set of changepoints that minimize a given cost. The cost associated with a specific segmentation consists of two important specifications. The first of these is $\mathcal{C}(.)$, the cost incurred from a segment of the data. Common choices for $\mathcal{C}(.)$ include quadratic error loss, Huber loss, and the negative

CONTACT S. O. Tickle s.tickle1@lancaster.ac.uk STOR-i Centre for Doctoral Training, Lancaster University, Lancaster LA1 4YW, UK.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/r/JCGS.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JCGS.

log-likelihood (for an appropriate within-segment model for the data); see Yao and Au (1989), Fearnhead and Rigaill (2017), and Chen and Gupta (2000) for further discussion. For example, using quadratic error loss gives

$$\mathcal{C}(y_{s:t}) = \sum_{i=s}^{t} \left(y_i - \frac{1}{t-s+1} \sum_{j=s}^{t} y_j \right)^2 . \quad (1)$$

This cost is proportional to the negative log-likelihood for a piecewise constant signal observed with additive Gaussian noise. The second specification is β, the penalty incurred when introducing a changepoint into the model. Common choices for β include the Akaike information criterion, Schwarz information criterion, and modified Bayesian information criterion (see Rigaill et al. 2013; Haynes, Eckley, and Fearnhead 2017; Truong, Gudre, and Vayatis 2017, and references therein for further discussion). Finally, it is assumed that the cost function is additive over segments. The objective is then to find the segmentation which minimizes the cost. In other words, we wish to find

$$\underset{1\leq\tau_1<\cdots<\tau_m\leq n-1}{\arg\min} \sum_{i=1}^{m+1} \left[\mathcal{C}(y_{\tau_{i-1}+1:\tau_i}) + \beta \right]. \quad (2)$$

Sometimes this minimization is performed subject to a constraint on the minimum possible segment length. Optimal partitioning, due to Jackson et al. (2005), uses dynamic programming to solve (2) exactly in a computation time of $\mathcal{O}(n^2)$. Killick, Fearnhead, and Eckley (2012) introduced the PELT algorithm, which also solves (2) exactly and can have a substantially reduced computational cost. In situations where the number of changepoints increases linearly with n, Killick, Fearnhead, and Eckley (2012) show that PELT's expected computational cost can be linear in n. However, the worst case cost is still $\mathcal{O}(n^2)$.

The basis of these dynamic programming algorithms is a simple recursion for the minimum cost of segmenting the first t data points, $y_{1:t}$, which we denote $F(t)$. It is straightforward to show that

$$F(u) = \min_{t<u} \left\{ F(t) + \mathcal{C}\left(y_{t+1:u}\right) + \beta \right\}.$$

The intuition is that we minimize over all possible values for the most recent changepoint prior to u, with the term in brackets being the minimum cost for segmenting $y_{1:u}$ with the most recent changepoint at t. By setting $F(0) = -\beta$ and solving this recursion for $u = 1, \ldots, n$, we obtain $F(n)$, the minimum value of (2). At the same time it is possible to obtain the set of changepoints which minimize the cost, see Jackson et al. (2005) for more details.

One of our approaches to parallelizing algorithms such as PELT will use the fact that (2) can still be solved exactly when we restrict the changepoints to be from an ordered subset $\mathcal{B} = \{b_1, \ldots, b_k\} \subset \{1, \ldots, (n-1)\}$. Let $F_{\mathcal{B}}(b_s)$ denote the minimum cost of $y_{1:b_s}$ when we restrict potential changepoints to $\mathcal{B}$; this satisfies the recursion

$$F_{\mathcal{B}}(b_s) = \min_{t<s} \left\{ F_{\mathcal{B}}(b_t) + \mathcal{C}(y_{b_t+1:b_s}) + \beta \right\}.$$

Using the initial condition $F_{\mathcal{B}}(0) = -\beta$, this gives a means of recursively calculating $F_{\mathcal{B}}(b_k)$. For most cost functions, after a simple preprocessing step that is linear in n, the computational cost of solving these recursions will be, at worst, quadratic in the size of $\mathcal{B}$ rather than quadratic in n. This property is key to the near quadratic speed ups we can obtain as we increase the number of cores. For both parallelization methods we introduce, each core minimizes the penalized cost while allowing changepoints at just a subset of locations. If we have L cores, then each core considers approximately n/L possible changepoint locations. Hence, the worst-case cost of minimizing the penalized cost on a given core is roughly a factor of L^2 less than that of running PELT on the full data. Furthermore, the parallelization schemes we introduce involve no communication between cores other than a single post-processing step of the output from each core.

The general format of this article is as follows: Section 2 introduces two means of parallelizing dynamic programming methods for solving (2), which we refer to as *Chunk* and *Deal*. In each case, we provide a description of the proposed algorithm with practical suggestions for implementation, followed by a short discussion of the theoretical justifications behind these choices. We devote Section 3 to examining this latter aspect in detail. In particular, we establish the asymptotic consistency of Chunk and Deal in a specific case with recourse to the asymptotic consistency of the penalized cost function method. Section 4 compares the use of parallelization to other common approaches in a number of scenarios involving changes in mean. We conclude with a short discussion in Section 5. The proofs of all results may be found in the appendices and supplementary materials.

2. Parallelization of Dynamic Programming Methods

In this section, we introduce Chunk and Deal, two methods for parallelizing dynamic programming procedures for changepoint detection. For convenience, we shall herein refer to this exclusively as the parallelization of PELT.

We introduce the notation PELT$(y_{\mathcal{A}}, \mathcal{B})$ when referring to applying PELT to a dataset $y_{\mathcal{A}}$ but only allowing candidate changepoints to be fitted from within the set $\mathcal{B}$. Note that we trivially require $\mathcal{B} \subseteq \mathcal{A}$. Thus, for example, when performing PELT without any parallelization, we may refer to this as PELT $\left(y_{\{1,\ldots,n\}}, \{1, \ldots, n-1\}\right)$.

In addition, we refer to the *parent core* as the core which is responsible for dividing the problem into subproblems and then distributing these subproblems to the other cores available. It then receives the output from each core (i.e., a set of estimated changepoints) and fits a changepoint model across the entire sequence using the results from these other cores.

Using this notation, the general setup for the parallelization procedure then takes the following form:

- (Split phase) We divide the space $\{1, \ldots, (n-1)\}$ into (not necessarily disjoint) subsets $\mathcal{B}_1, \ldots, \mathcal{B}_L$, where L is the number of computer cores available;
- Each of the cores $i = 1, \ldots, L$ then performs PELT$(y_{\mathcal{A}_i}, \mathcal{B}_i)$, returning a candidate set, $\hat{\boldsymbol{\tau}}_i$, of changes, which are returned to the parent core;

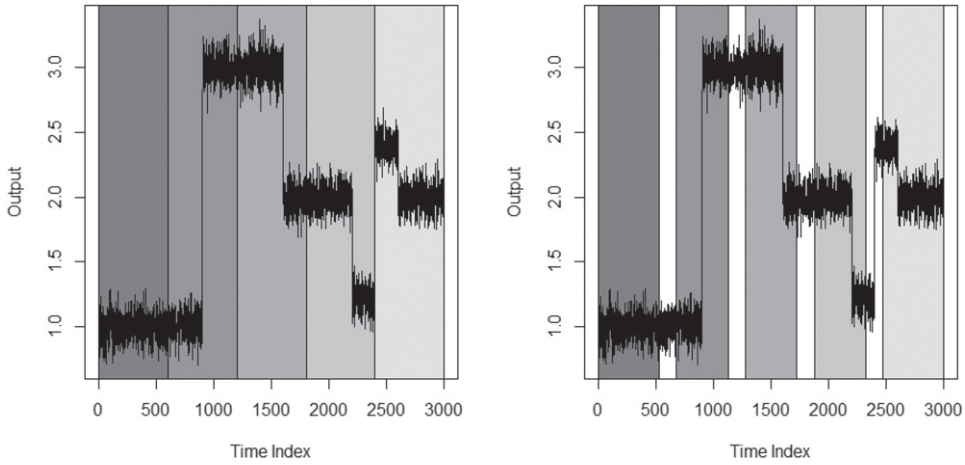

Figure 1. The time series is split into continuous segments by the Chunk procedure, in this case with 5 cores (l). An overlap is specified between the segments such that points within are considered by both adjacent cores (r).

- (Merge phase) The parent core then performs PELT$(y_{1:n}, \cup_{i=1}^{L} \hat{\boldsymbol{\tau}}_i)$, and the method returns $\hat{\boldsymbol{\tau}}$, the set of estimated changes found at this stage.

Note that in the above we require $\cup_{i=1}^{L} \mathcal{A}_i = \{1, \ldots, n\}$. Our two methods for parallelization differ in how they choose $\mathcal{A}_{1:L}$ and $\mathcal{B}_{1:L}$.

2.1. Chunk

The Chunk procedure consists of dividing the data into continuous segments and then handing each core a separate segment on which to search for changes. This splitting mechanism is shown in Figure 1. One problem with this division arises from changes which can be arbitrarily close to, or coincide with, the "boundary points" of adjacent cores. This necessitates the use of an overlap—a set of points which are considered by both adjacent cores for potential changes, also shown in Figure 1. For a time series of length n, we choose an overlap of size $V(n)$ either side of the boundary for each core (with the exception of the first and final cores, which can each trivially only overlap in one direction). The full procedure for Chunk is detailed in Algorithm 1.

Given that Algorithm 1 executes PELT multiple times, it is not immediate that Chunk represents a computational gain. We therefore briefly examine the speed of the procedure. Recall that PELT has a worst case computational cost that is quadratic in the number of possible changepoint locations. Such a quadratic cost is observed empirically when the number of changepoints is fixed as n increases. Taking this worst case computational cost, the cost of the split stage is $\mathcal{O}\left(\left(\frac{n}{L}\right)^2\right)$. The cost of the merge phase is dependent on the total number of estimated changes generated in the split phase. If we can estimate changepoint locations to sufficient accuracy, then as each change appears in at most two of the "chunks," the number of returned changes ought to be at most $2m$. Thus, the merge phase has a cost that is $\mathcal{O}(m^2)$. This intuition is confirmed later in Corollary 3.1.

These calculations suggest that by increasing L we can decrease the computational cost by a factor of close to L^2. This is observed empirically for large n and few changepoints. In situations where there are many changepoints, the computational cost for PELT can be much faster than its worst-case cost, and the computational gains will be less.

Data: A univariate dataset, $y_{1:n}$.
Result: A set of estimated **changepoint locations** $\hat{\tau}_1, \ldots, \hat{\tau}_{\hat{m}}$.
Step 1: Split the dataset into the subsets $\mathcal{B}_1, \ldots, \mathcal{B}_L$ such that
$\mathcal{B}_1 = \left\{1, \ldots, \left\lfloor \frac{n}{L} \right\rfloor + V(n)\right\}$,
$\mathcal{B}_i = \left\{(i-1)\left\lfloor \frac{n}{L} \right\rfloor - V(n), \ldots, i\left\lfloor \frac{n}{L} \right\rfloor + V(n)\right\}$
$\forall i \in \{2, \ldots, L-1\}$,
$\mathcal{B}_L = \left\{(L-1)\left\lfloor \frac{n}{L} \right\rfloor - V(n), \ldots, n\right\}$;
for $i = 1, \ldots, L$ **do**
  On core i, find $\hat{\boldsymbol{\tau}}_i = \text{PELT}\left(y_{\mathcal{B}_i}, \mathcal{B}_i\right)$;
end
Step 2: Sort $\cup_{i=1}^{L} \hat{\boldsymbol{\tau}}_i$ into ascending order;
Step 3: Calculate and return
$\left(\hat{\tau}_1, \ldots, \hat{\tau}_{\hat{m}}\right) = \text{PELT}\left(y_{1:n}, \cup_{i=1}^{L} \hat{\boldsymbol{\tau}}_i\right)$.

Algorithm 1: Chunk for the PELT procedure

To guarantee that the method does not overestimate the number of changes, some knowledge of the location error inherent in the PELT procedure is needed. This motivates the results of Section 3, which in turn imply various practical choices for the length of the overlap region, $V(n)$. In particular, using $V(n) = \left\lceil (\log n)^2 \right\rceil$ will give an effective guarantee of the accuracy of the method. Other sensible choices for $V(n)$ can be made based on the trade-off between accuracy and speed (see Section 3 for details).

2.2. Deal

The Deal procedure allows each core to segment the complete data, but restricts them to consider a subset of possible changepoint locations. This is done analogously to dealing the possible changepoints locations to the cores: so each core will receive every Lth possible location. A pictorial example of Deal is shown in Figure 2.

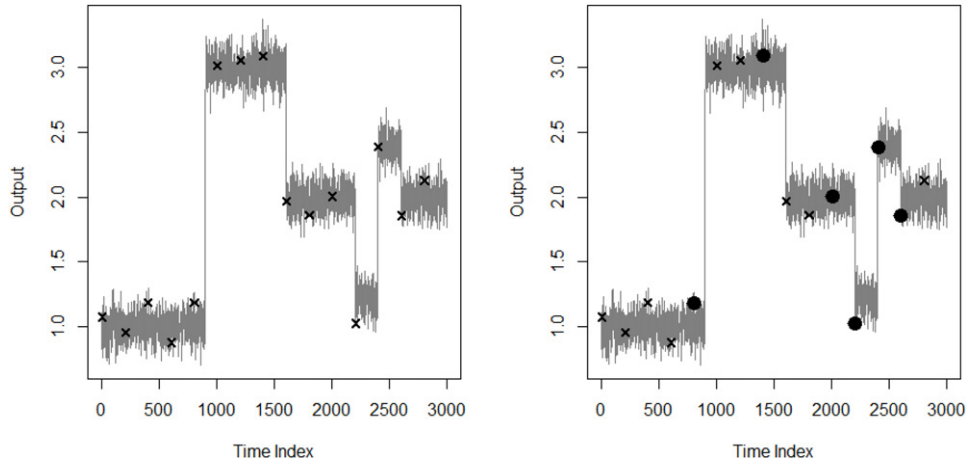

Figure 2. The time series is distributed across a number of cores by the Deal procedure. A particular core is given a certain collection of equally spaced points, for example, the points denotes by crosses (l). This core will then fit a changepoint model using only these points as candidate changes. The points estimated as changes are returned to the parent core. These points are circled (r).

Data: A univariate dataset, $y_{1:n}$.
Result: A set of estimated **changepoint locations** $\hat{\tau}_1, \ldots, \hat{\tau}_{\hat{m}}$.
Step 1: Split the dataset into subsets $\mathcal{B}_1, \ldots, \mathcal{B}_L$ such that $\mathcal{B}_i = \{i, L+i, \ldots, Q_i(L,n)L + i\}$;
for $i = 1, \ldots, L$ **do**
  On core i, find $\hat{\boldsymbol{\tau}}_i = \text{PELT}\left(y_{1:n}, \mathcal{B}_i\right)$;
end
Step 2: Sort $\cup_{i=1}^{L} \hat{\boldsymbol{\tau}}_i$ into ascending order;
Step 3: Calculate and return $\left(\hat{\tau}_1, \ldots, \hat{\tau}_{\hat{m}}\right) = \text{PELT}\left(y_{1:n}, \cup_{i=1}^{L} \hat{\boldsymbol{\tau}}_i\right)$.

Algorithm 2: Deal for the PELT procedure

Formally, we define $Q_a(b,c)$ as the largest integer such that $Q_a(b,c) \times b + (a \bmod b) < c$. The split phase then partitions $\{1, \ldots, (n-1)\}$ as follows

$$\begin{aligned}
\mathcal{B}_1 &= \{1, L+1, 2L+1, \ldots, Q_1(L,n)L + 1\}, \\
\mathcal{B}_2 &= \{2, L+2, 2L+2, \ldots, Q_2(L,n)L + 2\}, \\
&\ldots \\
\mathcal{B}_L &= \{L, 2L, 3L, \ldots, Q_L(L,n)L\}.
\end{aligned}$$

This splitting mechanism is shown in Figure 2. On the kth core, the objective function to be minimized then becomes

$$\min_{m, \tau_1, \ldots, \tau_m \in \mathcal{B}_k} \sum_{i=1}^{m+1} \left\{ \mathcal{C}(y_{(\tau_{i-1}+1):\tau_i}) + \beta \right\},$$

as discussed in Section 1. When the estimated changepoints from each core have been found and returned, the parent core then fits a changepoint model for the data sequence, using only points returned from the cores as changepoint candidates.

The full procedure for Deal is detailed in Algorithm 2.

As for the Chunk procedure, the implementation of Deal leads to computational gains. Similar to the previous section, the worst case computational time of the split phase of Deal will be $\mathcal{O}\left(\left(\frac{n}{L}\right)^2\right)$. The speed of the merge phase is again dependent on the number of changes detected at the split phase. We demonstrate in the proof of Corollary 3.1 that, with probability tending to 1, the number of changes detected by each core is at most $2m$, meaning that the worst case performance of the merge phase is $\mathcal{O}_p\left(L^2\right)$.

We remark that while the Chunk and Deal procedures do not inherit the exactness of PELT in finding the optimal solution to (2), they nevertheless track the true optimum very closely, as seen by the empirical results in Section 4.

3. Consistency of Parallelized Approaches

Our two methods, Chunk and Deal, are no longer guaranteed to minimize (2). Thus, we turn to the question as to whether, regardless, the estimates of the number and location of the changepoints they give still retain desirable asymptotic properties. We investigate for the canonical change-in-mean model with infill asymptotics.

This corresponds to our time series, $y_1, \ldots, y_n$, having changepoints corresponding to proportions $\theta_1, \ldots, \theta_m$, for some fixed m, such that, for a given n, the changepoints $\tau_1, \ldots, \tau_m$ are defined as $\tau_i = \lfloor \theta_i n \rfloor \ \forall i$. For the asymptotic setting we consider, take $\theta_{1:m}$ to be fixed.

With this framework in place, we note that the consistency results for Chunk and Deal we develop in Section 3.1 require one particular result not provided by Killick, Fearnhead, and Eckley (2012), namely consistency of PELT for the change in mean setting.

Proposition 3.1. We consider the change in mean setting for the univariate time series

$$Y_i = \delta_i + \mu_k, \text{ for } \tau_{k-1} + 1 \leq i \leq \tau_k \text{ and } k \in \{1, \ldots, m+1\}, \tag{3}$$

where $\mu_k \neq \mu_{k+1}$, for $k \in \{1, \ldots, m\}$ and $(\delta_1, \ldots, \delta_n)$ are a set of centered, independent and identically distributed Gaussian random variables with known variance σ^2. Take a series with m changes and true changepoint locations $\tau_1, \ldots, \tau_m$ (where $0 < \tau_1 < \cdots < \tau_m < n$). Apply the PELT procedure, minimizing squared error loss, with a penalty of $\beta = (2+\epsilon)\sigma^2 \log n$, for any $\epsilon > 0$, to produce an estimated set of $\hat{m}$ change locations

$0 < \hat{\tau}_1 < \cdots < \hat{\tau}_{\hat{m}} < n$. Then, for any $\alpha > 0$, $\mathbb{P}(\mathcal{E}_n^\alpha) \to 1$ as $n \to \infty$, where

$$\mathcal{E}_n^\alpha = \left\{ \hat{m} = m; \max_{i=1,\ldots,m} |\hat{\tau}_i - \tau_i| \leq \left\lceil (\log n)^{1+\alpha} \right\rceil \right\}.$$

Proof. See Section 2 of the supplementary materials. □

This result indicates that the probability of PELT misspecifying the number of changes, or the location of the true changes by more than a log-power factor, tends to 0 asymptotically. Note that this is with the Schwarz information criterion penalty in this setting, namely $2(1+\epsilon)\sigma^2 \log n$. While this proposition, and the related results given in the next section, assume the data have Gaussian distributions with common variance, it is straightforward to extend the results to sub-Gaussian random variables, or allow the variance to vary across the time-series provided the variance is upper-bounded. In the latter case we would need to replace σ^2 in the condition for the penalty with the maximum value the variance could take.

Proposition 3.1 also extends naturally to the same problem in the multivariate setting with d dimensions, with a penalty of $(d+1)(1+\epsilon)\sigma^2 \log n$ (see Section 2 of the supplementary materials for details). For the univariate case, the proof of Proposition 3.1 follows a similar pattern to that of Yao (1988), though we relax Yao's condition that an upper bound on the estimated number of changes is specified a priori.

3.1. Consistency and Computational Cost of Chunk and Deal

We now extend the consistency result in the unparallelized setting to obtain equivalent results for Chunk and Deal. If we fix the number of cores, L, as we increase n, many of the asymptotic results would follow trivially from existing results. For example, if we consider the Chunk approach and fix L as n increases then consistency would follow directly by the consistency of the analysis of data from each of the cores. Thus, in the following, we allow the number of cores to potentially increase as n increases, and use $L(n)$ to denote the number of cores used for a given sample size n.

Theorem 3.1. For the change in mean setting specified in (3), assume that for a data series of length n we have $L(n)$ cores across which to parallelize a changepoint detection procedure, and an overlap of $V(n)$ between adjacent cores. For any $\alpha > 0$ define $\mathcal{E}_n^\alpha$ as for the previous results. In addition to the assumptions of Proposition 3.1, assume that $L(n) = o(n)$ with $L(n) \to \infty$, that there exists a $\gamma > 1$ such that $V(n)/(\log n)^\gamma \to \infty$ and $V(n) = o(n)$. Then estimates from the Chunk procedure applied to a minimizing the least squared error under a penalty of $\beta = (2+\epsilon)\sigma^2 \log n$, satisfy $\mathbb{P}(\mathcal{E}_n^\alpha) \to 1$ as $n \to \infty$.

Proof. See Appendix. □

In our simulation study, we set $V(n) = \left\lceil (\log n)^2 \right\rceil$ which satisfies the condition of the theorem.

Theorem 3.2. If $L(n) \geq \left\lceil (\log n)^{1+\alpha} \right\rceil$, then the same result as for Theorem 3.1 holds with the Deal parallelization procedure.

Proof. See Appendix. □

Note that the conditions on $L(n)$ are stronger for Deal than for Chunk, with a lower bound corresponding with the maximum location error inherent in the event $\mathcal{E}_n^\alpha$. We believe the constraint on $L(n)$ is an artefact of the proof technique. Intuitively we would expect the statistical accuracy of Deal to be larger for smaller $L(n)$; as, for example, $L(n) = 1$ corresponds to optimally minimizing the cost. Practically, setting $L(n) = \lceil (\log n) \rceil$ is unlikely to be problematic for typical values of n, a notion which we confirm empirically in Section 4.

Finally, given these results, we are now in a position to give a formal statement on the worst case computational cost for both Chunk and Deal, when the computational cost of setting up a parallel environment is assumed to be negligible.

Corollary 3.1. Under the change in mean setting outlined in Proposition 3.1, with probability tending to 1 as $n \to \infty$, the worst case computational cost for Chunk when parallelizing the PELT procedure using $L(n)$ computer cores is $\mathcal{O}_p\left(\max\left(\left(\frac{n}{L(n)}\right)^2, m^2\right)\right)$, while for Deal the worst case cost is $\mathcal{O}_p\left(\max\left(\left(\frac{n}{L(n)}\right)^2, (L(n))^2\right)\right)$, compared to a worst case cost of $\mathcal{O}(n^2)$ for unparallelized PELT.

Proof. See Appendix. □

In the best case, we achieve a computational gain which is quadratic in $L(n)$. These results also show there is a limit to the gains of parallelization as we continue to increase the number of cores. This is particularly true for Deal, where larger values of $L(n)$ can lead to more candidate changepoints considered in the merge phase. For large $L(n)$ the cost of the merge phase will then dominate the overall cost of the Deal procedure. Setting $L(n) \sim n^{\frac{1}{2}}$ in Corollary 3.1 guarantees a worst case computational cost of $\mathcal{O}_p(n)$ for both Chunk and Deal, no matter the performance of PELT. We emphasize again that this result ignores the cost of setting up a parallel environment, which can lead to PELT performing better computationally for small n. Therefore, we now conduct a simulation study to understand the likely practical circumstances in which parallelization is a more efficient option.

4. Simulations

We now turn to consider the performance of these parallelized approximate methods on simulated data.

While these suggested parallelization techniques do speed up the implementation of the dynamic programming procedure underlying, say, PELT, the exactness of PELT in resolving (2) is no longer guaranteed. We therefore compare parallelized PELT with WBS, proposed by Fryzlewicz (2014), a non-exact change-

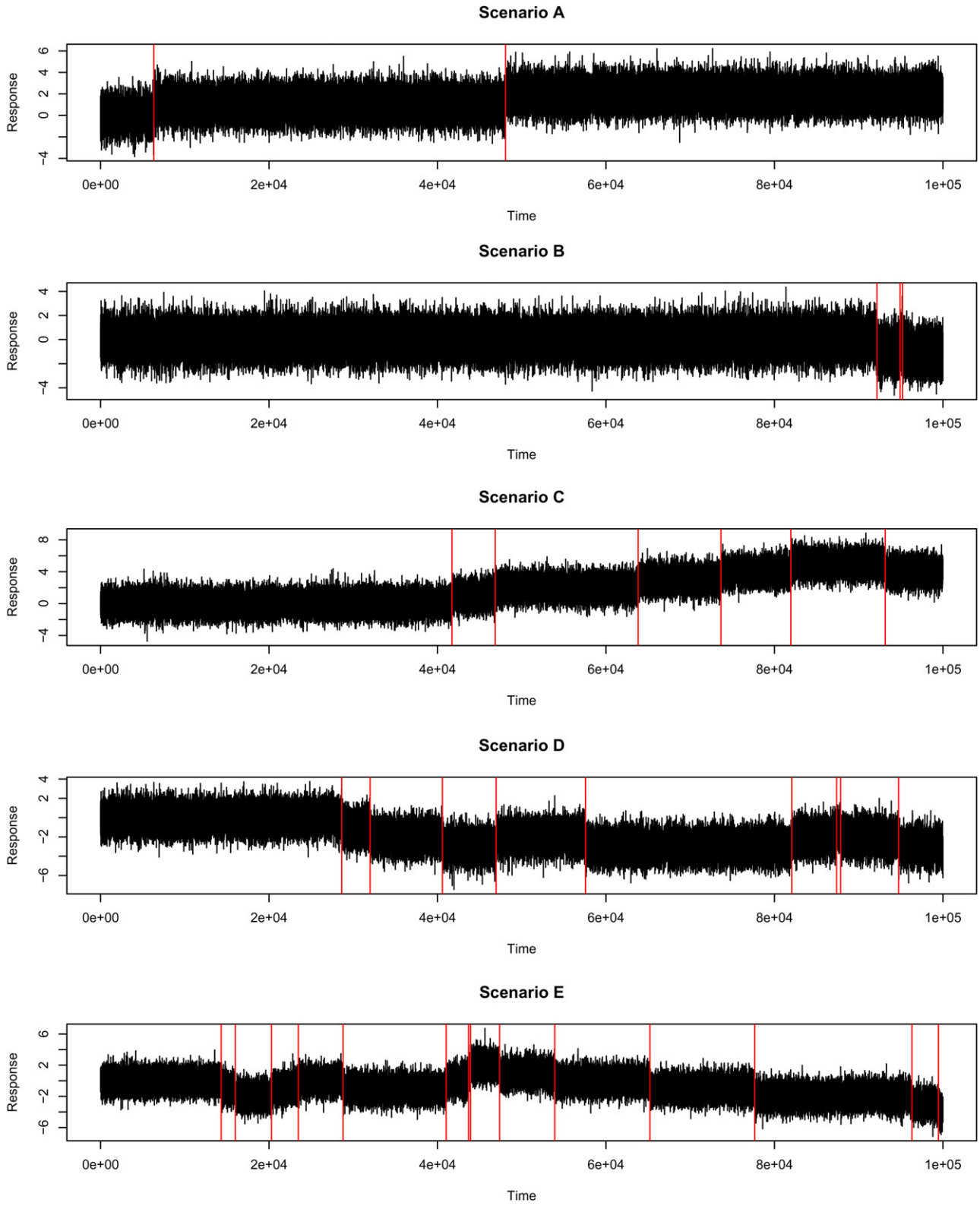

Figure 3. Five scenarios under examination in the simulation study. From top to bottom are scenarios A, B, C, D, and E with 2, 3, 6, 9, and 14 true changes, respectively.

point method which has impressive computational speed. To implement WBS, we used the wbs R package of Baranowski and Fryzlewicz (2015).

Simulated time series with piecewise normal segments were generated. Five scenarios, with changes at particular proportions of the time series, were examined in detail in the study. For a time series length of 100,000, these scenarios are shown in Figure 3.

Different lengths of series for each of the five scenarios—keeping the changepoints fixed at particular proportions in the time series as per the asymptotic regime outlined at the beginning of Section 3—were used to examine the statisti-

Table 1. The average number of false alarms recorded across all 200 repetitions for each of the five scenarios A, B, C, D, and E.

Average false alarms		Length = 10^3 $\Delta\mu$				Length = 10^4 $\Delta\mu$				Length = 10^5 $\Delta\mu$			
Scenario	Method	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
A	PELT	0.65	0.72	0.24	0.01	1.36	0.72	**0.15**	**0.00**	1.28	**0.59**	**0.10**	**0.00**
(2 changes)	Chunk4	0.67	0.87	**0.21**	**0.01**	1.49	0.72	0.16	**0.00**	1.29	**0.59**	**0.10**	**0.00**
	Deal4	0.64	0.69	0.22	0.01	1.35	0.72	**0.15**	**0.00**	1.28	**0.59**	**0.10**	**0.00**
	WBS	**0.54**	**0.66**	0.29	0.08	**1.20**	**0.66**	0.16	**0.00**	**1.26**	**0.59**	**0.10**	**0.00**
B	PELT	0.17	0.26	**0.15**	**0.01**	0.75	0.46	0.14	**0.00**	0.98	**0.83**	**0.09**	**0.00**
(3 changes)	Chunk4	**0.15**	**0.24**	0.16	**0.01**	0.70	0.46	0.14	**0.00**	0.98	**0.83**	**0.09**	**0.00**
	Deal4	0.16	0.27	**0.15**	**0.01**	0.75	0.46	0.14	**0.00**	0.98	**0.83**	**0.09**	**0.00**
	WBS	**0.15**	0.25	0.19	0.07	**0.55**	**0.45**	**0.12**	0.02	**0.97**	0.93	0.24	0.10
C	PELT	0.87	1.01	**0.68**	**0.12**	2.79	**2.08**	0.37	**0.00**	**3.94**	1.89	0.20	**0.00**
(6 changes)	Chunk4	0.89	**1.00**	0.73	0.15	**2.66**	2.11	0.37	**0.00**	3.96	**1.88**	**0.19**	**0.00**
	Deal4	**0.84**	1.02	0.69	**0.12**	2.81	**2.08**	**0.36**	**0.00**	**3.94**	1.89	0.20	**0.00**
	WBS	0.86	1.23	1.07	0.23	2.73	2.40	0.66	0.08	4.11	2.17	0.53	0.11
D	PELT	1.03	1.17	0.61	**0.09**	3.42	2.83	**0.60**	0.11	5.16	**2.73**	**0.43**	**0.00**
(9 changes)	Chunk4	1.02	1.16	0.63	0.12	**3.10**	**2.81**	**0.60**	**0.10**	**5.14**	**2.73**	**0.43**	**0.00**
	Deal4	1.01	**1.11**	**0.60**	**0.09**	3.41	2.83	0.61	0.11	5.16	**2.73**	**0.43**	**0.00**
	WBS	**0.97**	1.27	1.01	0.17	3.20	3.10	0.90	0.20	5.42	3.26	0.79	0.17
E	PELT	0.94	1.16	**0.64**	**0.07**	3.93	3.64	**0.86**	**0.07**	8.12	4.07	**0.59**	**0.05**
(14 changes)	Chunk4	0.99	1.27	0.91	0.30	**3.85**	3.64	0.90	0.10	8.16	**4.06**	**0.59**	**0.05**
	Deal4	**0.92**	**1.15**	0.65	0.09	3.91	**3.63**	**0.86**	**0.07**	8.11	4.07	**0.59**	**0.05**
	WBS	1.01	1.67	1.24	0.24	3.86	4.23	1.24	0.18	8.14	4.50	1.08	0.18

NOTE: A false alarm is defined as an estimated changepoint which is at least $\lceil(\log n)\rceil$ points from the closest true changepoint. Bold entries show the best performing algorithm.

Table 2. The average number of missed changes across all 200 repetitions for each of the 5 scenarios A, B, C, D, and E.

Average num. missed		Length = 10^3 $\Delta\mu$				Length = 10^4 $\Delta\mu$				Length = 10^5 $\Delta\mu$			
Scenario	Method	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
A	PELT	**1.78**	**1.14**	0.22	**0.01**	**1.38**	0.71	**0.14**	**0.00**	1.28	**0.59**	**0.10**	**0.00**
(2 changes)	Chunk4	1.95	1.39	**0.21**	**0.01**	1.56	0.72	0.15	**0.00**	1.29	**0.59**	**0.10**	**0.00**
	Deal4	**1.78**	1.15	0.22	**0.01**	**1.38**	0.71	**0.14**	**0.00**	1.28	**0.59**	**0.10**	**0.00**
	WBS	1.84	1.29	0.22	**0.01**	1.45	**0.66**	0.16	**0.00**	**1.26**	**0.59**	**0.10**	**0.00**
B	PELT	**2.63**	**2.06**	**1.19**	1.02	**2.47**	**1.94**	1.22	**0.00**	2.45	**0.86**	**0.09**	**0.00**
(3 changes)	Chunk4	2.65	2.15	1.22	1.03	2.48	1.95	1.22	**0.00**	2.45	**0.86**	**0.09**	**0.00**
	Deal4	**2.63**	2.08	**1.19**	1.03	**2.47**	1.95	1.25	**0.00**	2.44	**0.86**	**0.09**	**0.00**
	WBS	2.65	2.13	1.29	**0.91**	2.51	1.95	**1.06**	0.01	**2.43**	1.02	0.16	0.01
C	PELT	5.55	**4.87**	2.29	0.95	**4.85**	2.08	0.37	**0.00**	**3.94**	1.89	0.20	**0.00**
(6 changes)	Chunk4	5.69	4.99	2.56	1.00	5.01	2.11	0.37	**0.00**	3.96	**1.88**	**0.19**	**0.00**
	Deal4	**5.54**	**4.87**	2.38	0.98	4.88	**2.08**	**0.36**	**0.00**	**3.94**	1.89	0.20	**0.00**
	WBS	5.57	4.71	**1.22**	**0.08**	4.90	2.36	0.56	0.03	4.05	2.08	0.48	0.04
D	PELT	8.26	7.10	4.67	2.80	**7.51**	**4.39**	1.78	0.74	**6.43**	2.76	**0.44**	**0.00**
(9 changes)	Chunk4	8.40	7.19	4.78	2.98	7.67	4.43	1.79	0.73	**6.43**	**2.75**	**0.44**	**0.00**
	Deal4	8.26	7.07	4.68	2.87	7.53	4.40	1.81	0.74	6.45	2.76	**0.44**	**0.00**
	WBS	**8.22**	**6.66**	**2.65**	**0.66**	7.79	4.57	**1.07**	**0.07**	6.48	3.21	0.67	0.02
E	PELT	**13.0**	11.8	9.43	7.62	**12.3**	7.75	3.54	2.29	9.90	**4.75**	**0.82**	0.20
(14 changes)	Chunk4	13.2	12.1	9.91	8.04	12.4	7.89	3.63	2.40	9.95	**4.75**	**0.82**	0.20
	Deal4	**13.0**	11.9	9.53	7.71	**12.3**	7.78	3.54	2.29	9.89	4.76	**0.82**	0.20
	WBS	13.1	**11.2**	**6.09**	**1.53**	**12.3**	**7.46**	**2.51**	**0.16**	10.2	5.00	0.97	**0.04**

NOTE: A missed change is defined as a true changepoint for which no estimated change lies within $\lceil(\log n)\rceil$ points. Bold entries show the best performing algorithm.

cal power of PELT, Chunk, Deal, and WBS under 200 replications for the error terms. In addition, four change magnitudes ($\Delta\mu = 0.25$, 0.5, 1, and 2) were used to examine the behavior of the algorithms in each of the scenarios as $\Delta\mu$ was increased. When using PELT, Chunk, and Deal, we assumed a minimum spacing between consecutive changes of at least two points.

The number of false positives (which were counted as the number of estimated changes more than $\lceil\log n\rceil$ points from the closest true change) and missed changes (the number of true changes with no estimated change within $\lceil\log n\rceil$ points), as well as the maximum observed location error and average location error across all repetitions were measured. Finally, the average cost of the segmentations (using mean squared error) generated by the methods relative to the optimal given by PELT were recorded.

As can be seen from Tables 1 to 3, Chunk and Deal closely mirror WBS and PELT in statistical performance in finding approximately the same number of changes in broadly similar locations. This was particularly evident in situations where the

Table 3. The average location error between those true changes which were detected by the algorithms and the corresponding estimated change across all 200 repetitions for each of the 5 scenarios.

Average location error		Length = 10^3 $\Delta\mu$				Length = 10^4 $\Delta\mu$				Length = 10^5 $\Delta\mu$			
Scenario	Method	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
A (2 changes)	PELT	58.0	18.6	5.04	**1.23**	70.1	**11.5**	**3.25**	**1.19**	**46.0**	**11.7**	3.21	**1.26**
	Chunk4	**51.2**	18.8	**3.16**	1.24	90.3	12.1	3.35	1.18	47.4	**11.7**	3.21	**1.26**
	Deal4	61.0	**15.5**	3.21	**1.23**	57.3	**11.5**	**3.25**	**1.19**	**46.0**	**11.7**	3.21	**1.26**
	WBS	86.2	34.7	12.7	10.7	**52.4**	12.3	3.40	1.20	**46.0**	12.1	**3.18**	**1.26**
B (3 changes)	PELT	70.3	**31.5**	11.7	3.74	76.1	42.8	3.66	1.25	47.5	12.1	3.00	1.27
	Chunk4	77.5	37.2	12.5	**1.16**	72.3	41.6	3.59	**1.24**	**47.0**	12.1	3.00	1.27
	Deal4	70.3	32.9	**11.8**	3.77	74.6	41.6	3.65	**1.24**	47.1	**12.0**	**3.00**	**1.27**
	WBS	**59.9**	38.7	17.4	13.8	**32.2**	**11.0**	**3.25**	1.52	47.4	14.4	5.82	3.07
C (6 changes)	PELT	25.9	15.0	**4.38**	**1.53**	64.2	**11.9**	3.29	**1.23**	**50.3**	12.5	3.04	**1.23**
	Chunk4	26.3	**14.1**	4.53	1.77	**60.9**	12.7	3.29	**1.23**	50.7	**12.4**	**3.01**	1.24
	Deal4	25.5	14.8	**4.38**	1.54	64.3	12.0	**3.28**	**1.23**	**50.3**	12.5	3.04	**1.23**
	WBS	**21.8**	**14.1**	5.87	2.51	65.1	17.7	5.79	1.88	80.7	24.0	5.62	1.93
D (9 changes)	PELT	18.9	10.4	**3.57**	**1.43**	58.3	**13.2**	**3.52**	**1.47**	**86.0**	11.7	**3.32**	**1.25**
	Chunk4	19.6	10.9	3.71	1.54	63.6	13.8	3.68	**1.47**	86.8	**11.6**	**3.32**	**1.25**
	Deal4	18.8	**9.90**	**3.57**	1.44	**56.6**	**13.2**	3.53	**1.47**	86.2	11.7	**3.32**	**1.25**
	WBS	**17.6**	10.4	4.41	4.12	58.3	20.0	5.29	1.76	199	20.4	6.47	2.39
E (14 changes)	PELT	13.0	**8.68**	**3.78**	**1.44**	51.7	**13.3**	**3.60**	**1.39**	50.9	12.7	**3.48**	**1.44**
	Chunk4	15.0	9.88	4.91	2.09	65.8	15.0	4.14	1.73	52.0	**12.6**	**3.48**	**1.44**
	Deal4	**12.9**	9.01	**3.78**	**1.44**	**51.4**	**13.3**	3.64	**1.39**	50.8	12.8	**3.48**	**1.44**
	WBS	13.7	9.67	4.20	2.58	56.9	17.1	9.05	1.64	70.6	36.3	5.18	1.90

NOTE: Bold entries show the best performing algorithm.

Table 4. The time taken across 200 repetitions for each of the scenarios in question for PELT, Chunk, and Deal (using 4 cores).

Mean time taken (s)		Length = 10^3 $\Delta\mu$				Length = 10^4 $\Delta\mu$				Length = 10^5 $\Delta\mu$			
Scenario	Method	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
A (2 changes)	PELT	**0.06**	**0.06**	**0.05**	**0.05**	**1.61**	**1.44**	1.47	1.49	108	107	113	109
	Chunk4	1.48	1.49	1.37	1.13	1.90	1.89	1.83	**1.54**	23.9	24.0	21.1	24.1
	Deal4	1.59	1.23	1.59	1.49	1.72	1.70	**1.45**	1.69	**12.1**	**10.7**	**11.9**	**11.1**
B (3 changes)	PELT	**0.06**	**0.06**	**0.06**	**0.06**	2.23	2.27	2.35	2.57	147	144	154	165
	Chunk4	1.38	1.37	1.13	1.38	**1.78**	1.82	**1.55**	1.78	23.9	24.1	24.2	31.6
	Deal4	1.49	1.49	1.24	1.16	1.82	**1.45**	1.59	**1.59**	**16.2**	**16.5**	**16.5**	**16.4**
C (6 changes)	PELT	**0.06**	**0.05**	**0.04**	**0.03**	**1.23**	**0.94**	**0.93**	**0.88**	72.2	71.8	70.7	72.1
	Chunk4	1.48	1.13	1.38	1.48	1.84	1.50	1.73	1.85	22.3	20.0	23.2	29.7
	Deal4	1.58	1.58	1.49	1.15	1.46	1.42	1.28	1.37	**8.33**	**7.58**	**7.60**	**7.30**
D (9 changes)	PELT	**0.05**	**0.05**	**0.03**	**0.04**	**1.12**	**0.82**	**0.73**	**0.75**	60.6	55.5	56.9	55.4
	Chunk4	1.37	1.37	1.48	1.37	1.79	1.73	1.85	1.77	22.5	22.5	19.8	29.8
	Deal4	1.49	1.23	1.58	1.58	1.65	1.36	1.40	1.26	**6.66**	**6.58**	**6.26**	**6.91**
E (14 changes)	PELT	**0.05**	**0.05**	**0.04**	**0.04**	**1.03**	**0.69**	**0.63**	**0.58**	60.9	40.0	37.2	37.7
	Chunk4	1.42	1.38	1.48	1.37	2.15	1.65	1.74	1.65	28.8	14.3	16.0	16.0
	Deal4	1.50	1.58	1.48	1.23	1.55	1.38	1.56	1.33	**8.92**	**5.23**	**4.95**	**5.44**

NOTE: Bold entries show the best performing algorithm.

length of the series was 10^5. Here, the performance of Chunk and Deal becomes indistinguishable from PELT and WBS in most cases. However, as the number of changes and series length was increased, WBS was generally outperformed by both Chunk and Deal in terms of location accuracy. One additional aspect of note is that WBS was generally slightly more effective than the cost function based approaches at detecting the full set of changepoints in the scenarios with very short segments lengths (B, D, and E)—see Table 2 for the full picture.

From Table 4, we note that, in practice, Deal often outperforms Chunk in terms of computational speed for a given number of cores. This is due to the fact that the Deal procedure will rarely perform at the worst case computational speed during the split phase (which typically dominates the computation time), as one of the candidates around a true change is very likely to be chosen as a candidate changepoint (see the proof of Theorem 3.2). This means that more candidates for the most recent changepoint are pruned than for Chunk. PELT was observed to be the fastest method for the smallest value of n across all scenarios. It was at the larger values of n where the super-linear gains in speed of Chunk and Deal became apparent, as can also be seen in Figure 4, which indicates that both Chunk and Deal exhibit a super-linear gain in speed in most situations. The exception to this is the use of the Chunk algorithm in Scenario E, which has a comparatively large number of true changepoints. As a result of this, the maximum segment length in the series in Scenario E remains similar in both the PELT and Chunk settings, even as the number of cores is increased. Hence, the computation gains here are less impressive.

Table 5. The average relative computation gain of the Chunk and Deal methods relative to the PELT method across 200 repetitions for each of the scenarios in question.

Average relative gain in computation speed		Length $= 10^3$ $\Delta\mu$				Length $= 10^4$ $\Delta\mu$				Length $= 10^5$ $\Delta\mu$			
Scenario	Method	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
A (2 changes)	Chunk4	**0.04**	0.04	**0.03**	**0.04**	0.85	0.76	0.80	**0.97**	4.53	4.44	5.34	4.53
	Deal4	**0.04**	**0.05**	**0.03**	0.03	**0.94**	**0.85**	**1.01**	0.88	**8.94**	**9.97**	**9.46**	**9.83**
B (3 changes)	Chunk4	**0.04**	**0.05**	**0.06**	**0.05**	**1.25**	1.25	**1.52**	1.44	6.14	5.96	6.37	5.21
	Deal4	**0.04**	0.04	0.05	**0.05**	1.23	**1.57**	1.48	**1.62**	**9.05**	**8.71**	**9.34**	**10.0**
C (6 changes)	Chunk4	**0.04**	**0.04**	**0.03**	0.02	0.67	0.63	0.54	0.47	3.24	3.59	3.05	2.43
	Deal4	**0.04**	0.03	**0.03**	**0.03**	**0.84**	**0.66**	**0.72**	**0.64**	**8.67**	**9.47**	**9.31**	**9.88**
D (9 changes)	Chunk4	**0.04**	0.03	**0.02**	**0.03**	0.63	0.48	0.39	0.42	2.69	2.47	2.87	1.86
	Deal4	0.03	**0.04**	**0.02**	0.02	**0.68**	**0.61**	**0.52**	**0.59**	**9.10**	**8.43**	**9.08**	**8.01**
E (14 changes)	Chunk4	**0.04**	**0.03**	**0.03**	**0.03**	0.48	0.42	0.36	0.35	2.11	2.79	2.32	2.36
	Deal4	0.03	**0.03**	**0.03**	**0.03**	**0.66**	**0.50**	**0.40**	**0.44**	**6.82**	**7.64**	**7.51**	**6.94**

NOTE: These values are calculated by dividing corresponding values from Table 4. Bold entries show the best performing algorithm. Bold entries show the best performing algorithm.

Table 6. The error in average cost, calculated using the log-likelihood of the segments, resulting from executing Chunk and Deal with 4 cores.

Average cost—optimal		Length $= 10^3$ $\Delta\mu$				Length $= 10^4$ $\Delta\mu$				Length $= 10^5$ $\Delta\mu$			
Scenario	Method	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
A (2 changes)	Chunk4	1.70	1.57	0.03	0.01	3.17	0.05	0.01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Deal4	**0.01**	**0.03**	**0.01**	**0.00**	**0.02**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
B (3 changes)	Chunk4	0.12	0.51	0.13	0.09	0.19	0.01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Deal4	**0.01**	**0.05**	**0.02**	**0.04**	**0.01**	**0.01**	0.04	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
C (6 changes)	Chunk4	1.65	1.85	2.44	6.52	3.44	0.39	0.02	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Deal4	**0.03**	**0.04**	**0.07**	**0.05**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
D (9 changes)	Chunk4	2.30	2.23	2.90	7.44	4.10	1.13	1.42	0.01	0.10	**0.00**	**0.00**	**0.00**
	Deal4	**0.05**	**0.06**	**0.13**	**0.17**	**0.00**	**0.01**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
E (14 changes)	Chunk4	2.41	4.02	8.43	24.2	7.45	4.21	6.75	19.7	0.10	**0.00**	**0.00**	**0.00**
	Deal4	**0.05**	**0.11**	**0.19**	**0.29**	**0.02**	**0.02**	**0.03**	**0.10**	**0.00**	**0.00**	**0.00**	**0.00**

NOTE: This is adjusted according to the equivalent cost computed by PELT (which is optimal). Bold entries show the best performing algorithm.

An additional point of interest from Tables 4 and 5 is that PELT generally outperforms Chunk and Deal computationally when the time series is of length 10^3 or 10^4. This is due to the fact that the setting up of the parallel cluster takes around one second to complete, while the PELT algorithm takes significantly less time than this for shorter data sequences.

Finally, from Table 6, both Chunk and Deal are seen to track PELT very closely in terms of the final cost of the model. This appears to be particularly true for the datasets of greater length, where the average cost seen under both Chunk and Deal was seen to be the same as PELT (up to our stated precision) for almost all situations we investigated. In light of the behavior seen from Tables 1 to 3, however, this should not be surprising.

Caution should be exercised when discussing these results in the context of the general statistical performance of Chunk and Deal, as only the value of $L = 4$ was tested.

All simulations were run in R using a Linux OS on a 2.3GHz Intel Xeon CPU. Simulations were run in batches of 20, grouped by length of series and detection method. When testing the PELT procedure, each job within a batch was assigned a separate core without any parallelization or external packages involved. For Chunk and Deal, although jobs were again run in batches of 20, each was assigned the number of cores across which the algorithm was to be parallelized. (This was 4 in all cases except to run the simulations to generate Figure 4.) Parallelization was implemented using the doParallel and foreach packages of Calaway, Weston, and Tenenbaum (2018) and Calaway and Weston (2017), respectively. Note that the doParallel package uses multiprocessing as opposed to multithreading.

5. Discussion

We have proposed two new methods for changepoint detection, Chunk and Deal, each based on parallelizing an existing method, PELT. These methods represent a substantial computational gain in many cases, particularly for large n. In addition, by establishing the asymptotic consistency of PELT, we have been able in turn to show the asymptotic consistency of the Chunk and Deal methods, such that the error inherent to all three is $\mathcal{O}(\log n)$ in terms of the maximum location error of an estimated change relative to the corresponding true change. We have demonstrated empirically that an implication of this is that Chunk and Deal, while not inheriting the exactness of PELT, do perform well in finding changes in practice.

There are other approaches to reduce the computational cost of changepoint methods, while retaining the same asymptotic statistical properties. A suggestion, made by a reviewer, is that we could implement the Deal algorithm but with fewer candidates per core. Providing there is at least one core with a

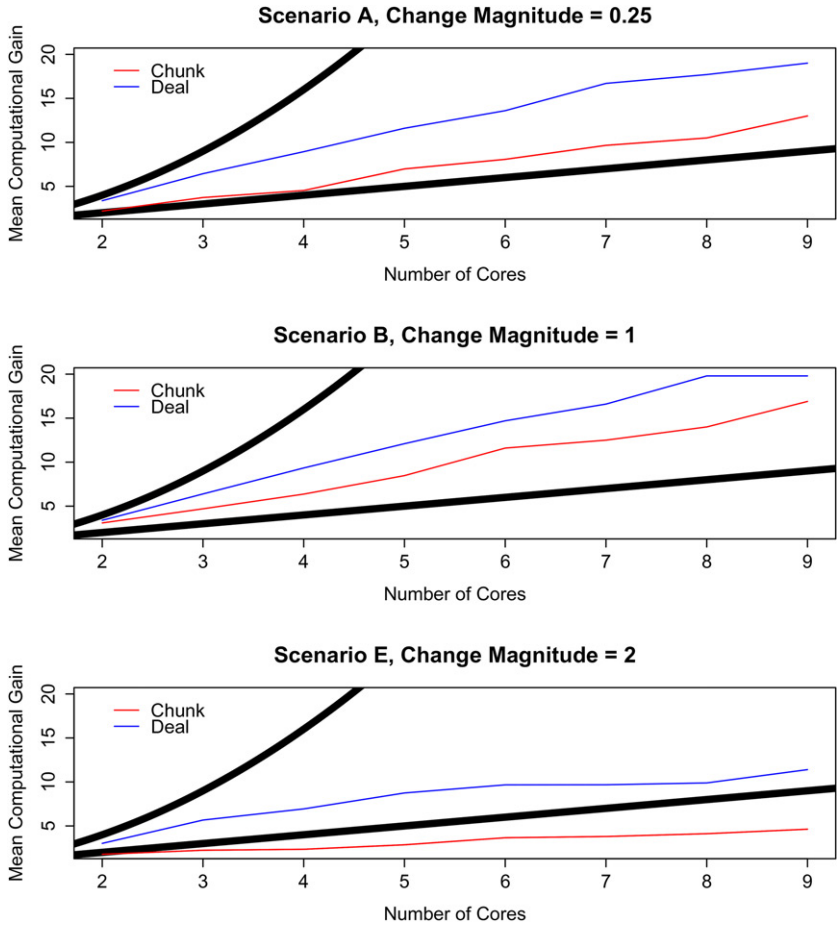

Figure 4. Mean computational gain (y) across 200 repetitions for Chunk and Deal compared to PELT across a differing number of cores (x) under three specific scenarios. The lines $y = x$ and $y = x^2$ are shown for comparison.

candidate close to the true change, say within $\log n$ of it, then under infill asymptotics of the kind discussed in Section 3 we will still detect the change with probability tending to 1 as n increases. Our empirical experience with such a method is that it can lose power at detecting changes in practical, non-asymptotic settings. Such a strategy has similarities to the ideas presented in Lu, Banerjee, and Michailidis (2018), and could be sensible in situations that they consider where n is exceedingly large, and it is computationally infeasible to analyze all the data.

Appendix

The following results will be stated with respect to a general $\alpha > 0$. Theoretically, this means that any $\alpha > 0$ can be used in Algorithm 1 or Algorithm 2, however, in the simulation study detailed in Section 4, $\lceil (\log n)^2 \rceil$ was used as the overlap length (for Chunk), while the cutoff value for closeness detailed in the merge phase (Step 3) of both procedures was taken as $\lceil (\log n) \rceil$.

Proof of Theorem 3.1. The Chunk procedure involves obtaining a set of candidate changepoints from analyzing the data sent to each core, and then finding the best segmentation using these changepoints in the merge phase. We claim that to show Chunk is consistent it is sufficient to show that, with probability tending to 1, there will be a segmentation using m of the candidate changepoints that gives an RSS that is within $o_p(\log n)$ of the RSS we obtain for the true segmentation.

This claim follows from a simple adaptation of the proof of Proposition 3.1. In that proof we show that, with probability tending to 1, for any penalty $(2+\epsilon)\sigma^2 \log n$ with $\epsilon > 0$, a segmentation with $\hat{m} > m$ changepoints will have a worse penalized cost than the true segmentation. Furthermore, any segmentation with $\hat{m} \leq m$ which is not in $\mathcal{E}_n^{\alpha}$ will miss one or more changepoints by more than $(\log n)^{1+\alpha}$ and will have a worse penalized cost than a segmentation with $\hat{m} > m$ changepoints (i.e., a segmentation obtained by adding three changepoints for each changepoint that is not estimated well enough). Thus to show our claim we need only show that, with probability tending to 1, we do not overestimate the number of changepoints.

Assume we use a penalized cost $(2+\epsilon)\sigma^2 \log n$ for Chunk. From the argument in the proof of Proposition 3.1 applied to the penalized cost with a penalty $(2+2\epsilon)\sigma^2 \log n$ we have that with probability tending to 1, for all $\hat{\tau}_{1:\hat{m}}$ with $\hat{m} > m$,

$$\begin{aligned} &\mathrm{RSS}\left(y_{1:n}; \hat{\tau}_{1:\hat{m}}\right) - \mathrm{RSS}\left(y_{1:n}; \tau_{1:m}\right) + \left(\hat{m} - m\right)(2+2\epsilon)\sigma^2 \log n > 0 \\ &\implies \mathrm{RSS}\left(y_{1:n}, \hat{\tau}_{1:\hat{m}}\right) - \left\{\mathrm{RSS}\left(y_{1:n}; \tau_{1:m}\right) + o_p\left(\log n\right)\right\} \\ &\quad + \left(\hat{m} - m\right)(2+\epsilon)\sigma^2 \log n > \epsilon\sigma^2 \log n + o_p\left(\log n\right), \end{aligned}$$

as required.

We now show that we will have a suitable set of candidate changepoints for the merge phase in two steps. The first of these steps establishes that each changepoint will be estimated within $\log \log n$.

By the set-up of Chunk each changepoint will appear in the non-overlap region of data assigned to precisely one core. Furthermore, as $L(n) \to \infty$ and $V(n) = o(n)$ then for large enough n the core that a changepoint is assigned to will have data that contains only that changepoint.

Consider the data associated with each such core. Such a core will have data with just a single changepoint and a minimum segment length that is at least $V(n)$. As for sufficiently large n $V(n) > \lceil \log n \rceil^{1+\gamma}$, for some $\gamma > 0$, then, by a simple adaptation of the argument in Section 1 of the supplementary materials, it is straightforward to show that with probability tending to 1 we will detect precisely one changepoint for this data. Standard results (see, e.g., Yao and Au 1989, Lemma 3) for detecting a single changepoint from Gaussian data shows that the error in the location is $O_p(1)$, and hence with probability tending to 1 we will detect the changepoint within an error of $\log \log n$.

As there are a finite number of changepoints, with probability tending to 1 we will detect precisely one changepoint with an error less than $\log \log n$ for all cores with a changepoint in the non-overlap region.

We now define, for a true segmentation of $\tau_{1:m}$ and sequence of length n, a *set of good segmentations*, $\mathcal{H}(\tau_{1:m}, n)$ such that

$$\mathcal{H}(\tau_{1:m}, n) = \left\{ \hat{\tau}_{1:\hat{m}} | \hat{m} = m, \left| \hat{\tau}_i - \tau_i \right| \leq \log \log n \text{ for } i \in \{1, \ldots, m\} \right\}.$$

The second phase is to show that for any set of changepoints $\hat{\tau}_{1:\hat{m}} \in \mathcal{H}(\tau_{1:m}, n)$, the maximum difference between the RSS for fitting changepoints at $\hat{\tau}_{1:m}$ and the RSS for fitting changepoints at the true locations is $O_p(\log \log n)$.

Define $\Delta\mu_k := \left| \mu_k - \mu_{k+1} \right|$. For any appropriate $\hat{\tau}_{1:m}$ we have

$$\begin{aligned}
&\text{RSS}(y_{1:n}; \hat{\tau}_{1:m}) - \text{RSS}(y_{1:n}; \tau_{1:m}) \\
&\leq \sum_{i=1}^{m+1} \left\{ \frac{1}{\tau_i - \tau_{i-1}} \left(\sum_{j=\tau_{i-1}+1}^{\tau_i} Z_j \right)^2 - \frac{1}{\hat{\tau}_i - \hat{\tau}_{i-1}} \left(\sum_{j=\hat{\tau}_{i-1}+1}^{\hat{\tau}_i} Z_j \right)^2 \right\} \\
&+ \sum_{k=1}^{m} \left(\Delta\mu_k \right)^2 \log \log n + G,
\end{aligned}$$

where the fourth term, G, depends on $\hat{\tau}_{1:m}$ with $G \sim N\left(0, 4\sigma^2 \sum_{k=1}^{m} \left(\Delta\mu_k \right)^2 \log \log n\right)$. Note that the first term in this inequality does not depend on $\hat{\tau}_{1:m}$ and has a χ^2_{m+1} distribution, and so is $O_p(\log \log n)$, the second term is negative, and the third term is a constant multiple of $\log \log n$. So it only remains to check that $G = O_p\left(\log \log n\right)$ uniformly across all members of $\mathcal{H}$. This follows trivially from standard bounds on a Gaussian distribution together with a Bonferroni correction over the $\left(2 \log \log n\right)^m$ possibilities for $\hat{\tau}_{1:m}$. □

Proof of Theorem 3.2. Recall that $L(n) \geq \left\lceil \left(\log n \right)^{1+\alpha} \right\rceil$ and $L(n) = o(n)$. The idea will be to show that the core which is "dealt" a particular true change, τ_i, will always return this true change as a candidate changepoint for the merge phase. By Yao (1988), letting $\hat{\tau}_{1:m}$ be a set of estimated changes which miss the true change τ_i by at least $\left\lceil \left(\log n \right)^{1+\alpha} \right\rceil$, then again by the proof of Corollary 1.2.1 the cost of this segmentation is strictly worse than the cost of also fitting changes at the points $\tau_i - L(n)$ and $\tau_i + L(n)$. By then considering the difference

$$\begin{aligned}
\text{Diff} := &\text{RSS}(y_{1:n}; \hat{\tau}_{1:m}, \tau_i - L(n), \tau_i + L(n)) \\
&- \text{RSS}(y_{1:n}; \hat{\tau}_{1:m}, \tau_i - L(n), \tau_i, \tau_i + L(n)),
\end{aligned}$$

in a similar fashion to the proof of Corollary 1.2.1, it can be shown that in probability

$$\frac{\text{Diff}}{L(n)} \to \left(\Delta\mu_{i-1} \right)^2,$$

where again $\Delta\mu_{i-1}$ is the absolute change in mean at the changepoint τ_i. □

Proof of Corollary 3.1. It is sufficient to prove the following Claim regarding the number of candidate changes each core returns.

Claim: With probability tending to 1, and for any candidate set given to the cores in accordance with the conditions of Theorems 3.1 and 3.2:

(I): under the Chunk procedure, the maximum number of points returned for the merge phase is bounded above by $2m$,

(ii): under Deal, the maximum number of points recorded as estimated changes is bounded above by $2m$ for each core.

Proof of Claim:

Proof of (I): We note that when $L(n)$ is constant, the result is immediate from the proof of Lemma 3.1.

When $L(n) \to \infty$, it suffices to show that across all cores which are given no true changes, the probability of any of these cores returning a true change converges to 0. Given that the number of cores which are given a change is fixed (and bounded above at $2m$—as each change could fall inside an overlap), the result is then immediate from the proof of Theorem 3.1.

Considering a single core with no true changes, we adapt the argument from the proof Proposition 3.1. For a quantity U_{k+1} which is distributed according to a χ^2_{k+1} distribution, then by Laurent and Massart (2000)

$$\mathbb{P}(U_{k+1} \geq d \log n) \leq n^{-\frac{d}{2}+\delta}, \quad \text{for any } \delta > 0.$$

Fitting $k > 0$ changes across a core will give that the residual sum of squares relative to a fit of no changes across the same core follows a χ^2_{k+1} distribution. Therefore, following the application of a Bonferroni correction across all possible placings of k changes gives that the difference between the null fit and the best possible fit of k changes is then bounded in probability as

$$\mathbb{P}(\text{Diff}_k \geq d \log n) \leq n^{-\frac{d}{2}+\delta} \times \left(\frac{n}{L(n)} \right)^k.$$

In particular, setting $d = 2k(1+\epsilon)$ and $\delta = \epsilon/2$ as before, gives that

$$\begin{aligned}
\sum_{k=1}^{n/L(n)} \mathbb{P}(\text{Diff}_k \geq 2k(1+\epsilon) \log n) &\leq \sum_{k=1}^{n/L(n)} \frac{n^{-\frac{(2k-1)\epsilon}{2}}}{(L(n))^k} \\
&= \frac{n^{-\frac{\epsilon}{2}}}{L(n)} \left(\frac{1 - n^{-\epsilon \frac{n}{L(n)}} L(n)^{-\frac{n}{L(n)}}}{1 - n^{-\epsilon} L(n)^{-1}} \right) \to 0, \quad \forall \epsilon > 0,
\end{aligned}$$

and so scaling this by $L(n)$

$$\mathbb{P}(\text{A core with no true changes overfits}) \to 0, \quad \forall \epsilon > 0.$$

Therefore, the computation time of the merge phase of Chunk is $\mathcal{O}(m^2)$ in the worst case, which along with the worst case cost from the split phase of $\mathcal{O}_p\left(\left(\frac{n}{L(n)} \right)^2 \right)$ gives the worst case computation time for the whole procedure.

Proof of (II): Define, for a given core under the Deal procedure

$$\mathcal{S}_2 = \left\{ s_1^{(1)}, s_1^{(2)}, s_2^{(1)}, s_2^{(2)}, \ldots, s_m^{(1)}, s_m^{(2)} \right\},$$

where $s_i^{(1)}$ is the final point given to the core which is strictly before τ_i, and $s_i^{(2)}$ is the first point given to the core which is after τ_i. In the same way as for the proof of Proposition 3.1, we examine the best possible segmentations which include $\mathcal{S}_2$ as a subset of the estimated changepoints for a core, and show that all are rejected in favor of $\mathcal{S}_2$ in probability. We then show that this is true across all cores in probability.

For a given core, suppose $\mathcal{S}_3$ is a set of points estimated as changes under the Deal procedure such that $\mathcal{S}_2 \subset \mathcal{S}_3$. By construction of $\mathcal{S}_2$, all points in $\mathcal{S}_3 \cap \mathcal{S}_2^c$ must lie in a region between two points of $\mathcal{S}_2$

which also does not contain any true changes. We can therefore apply the same argument as for Proposition 3.1 to the difference

$$\text{Diff} := \text{RSS}(y_{\mathcal{A}}; \mathcal{S}_2) - \text{RSS}(y_{\mathcal{A}}; \mathcal{S}_3),$$

where $\mathcal{A}$ refers to any such region between two consecutive points of $\mathcal{S}_2$ which contains a point found only in $\mathcal{S}_3$. Uniformly across such regions, and supposing $k > 0$ such estimated changes are found within $\mathcal{A}$, it can be seen that the positive term in the expression of the difference above is distributed as χ^2_{k+1}. Thus letting $\tilde{n} = \frac{n}{L(n)}$ and again with recourse to the Bonferroni correction argument as in Proposition 3.1, for a given $\epsilon > 0$

$$\sum_{k=1}^{\tilde{n}} \mathbb{P}(\text{Diff}_k \geq 2k(1+\epsilon)\log n) \leq \sum_{k=1}^{\tilde{n}} \frac{n^{-\frac{(2k-1)\epsilon}{2}}}{(L(n))^k}$$
$$= \frac{n^{-\frac{\epsilon}{2}}}{L(n)} \left(\frac{1 - n^{-\tilde{n}\epsilon} L(n)^{-\tilde{n}}}{1 - n^{-\epsilon} L(n)^{-1}} \right) \to 0, \quad \forall \epsilon > 0.$$

Note that this argument does not consider segmentations which do not contain $\mathcal{S}_2$ as a proper subset. To extend this argument, we define the following three sets of segmentations (with respect to a given core)

$$\mathcal{GS}_2 = \left\{ \hat{\boldsymbol{\tau}} : \left|\hat{\boldsymbol{\tau}}\right| = 2m; \hat{\tau}_{2t-1} \leq \tau_t, \hat{\tau}_{2t} > \tau_t, \forall t \in \{1, \ldots, m\} \right\},$$

$$\mathcal{GS}_1 = \Bigg\{ \hat{\boldsymbol{\tau}} : \left|\hat{\boldsymbol{\tau}}\right| \leq 2m; \left|\hat{\boldsymbol{\tau}} \cap \left\{\tau_t + 1, \ldots, \tau_{t+1}\right\}\right| \geq 1, \forall t \in \{0, \ldots, m\}; \left|\hat{\boldsymbol{\tau}} \cap \left\{\tau_t + 1, \ldots, \tau_{t+1}\right\}\right| = 1, t \notin \{0, m\} \Bigg\},$$

$$\mathcal{GS}_0 = \left\{ \hat{\boldsymbol{\tau}} : \left|\hat{\boldsymbol{\tau}}\right| \leq 2m; \left|\hat{\boldsymbol{\tau}} \cap \left\{\tau_t + 1, \ldots, \tau_{t+1}\right\}\right| = 0, \text{ some } t \right\}.$$

Note that $\mathcal{S}_2 \in \mathcal{GS}_2$ and that the argument showing that any segmentation $\mathcal{S}_3$ containing $\mathcal{S}_2$ is rejected uniformly in favor of $\mathcal{S}_2$ may be extended to any element of $\mathcal{GS}_2$ to show that any segmentation with more than $2m$ estimated changes in total and which has at least two estimated changes between each true change is uniformly dominated by a corresponding element of $\mathcal{GS}_2$.

In the same way, let us now consider extensions from a general element, $\mathcal{T}_1 \in \mathcal{GS}_1$, where here an extension is defined as a superset of $\mathcal{T}_1$ which also contains additional estimated changes from regions between two estimated changes within $\mathcal{T}_1$ not containing a true change. Letting, for example

$$\mathcal{T}_1 = \left\{ s_1^{(1)}, s_1^{(2)}, \ldots, s_{i-1}^{(2)}, s_i^{(k)}, s_{i+1}^{(1)}, \ldots, s_m^{(2)} \right\} \subset \mathcal{S}_2,$$

for some $k \in \{1, 2\}$ and $i \in \{1, \ldots, m\}$. Then any extensions of $\mathcal{T}_1$ consists of placing any further estimated changes in any of the regions between the changes above with the exception of either (if $k = 1$) the region $\left(s_i^{(1)}, s_{i+1}^{(1)}\right)$ or (if $k = 2$) the region $\left(s_{i-1}^{(2)}, s_i^{(2)}\right)$. Let $\mathcal{T}_1'$ be an arbitrary such extension, and again let $\mathcal{A}$ be any region between two consecutive points of $\mathcal{T}_1$ which contains a point found only in $\mathcal{T}_1'$. As before, uniformly across such regions, and supposing again that $k > 0$ such estimated changes are found within $\mathcal{A}$, letting

$$\text{Diff} := \text{RSS}(y_{\mathcal{A}}; \mathcal{T}_1) - \text{RSS}(y_{\mathcal{A}}; \mathcal{T}_1'),$$

then again Diff is distributed as χ^2_{k+1}. With recourse to the same argument as before (noting again that any such region $\mathcal{A}$ will have at most $\tilde{n} = \frac{n}{L(n)}$ candidate points for the extension—no matter which base element of $\mathcal{GS}_1$ we pick), and extending to other elements of $\mathcal{GS}_1$, we conclude that any segmentation with more than $2m$ estimated changes which places just one estimated change between two true changes in at least one case will be rejected uniformly (and for all cores) in favor of an element of $\mathcal{GS}_1$.

Finally, we consider all segmentations with more than $2m$ changes which place no estimated changes between two true changes in at least one case. We again compare with $\mathcal{T}_0 \in \mathcal{GS}_0$. Letting, for example

$$\mathcal{T}_0 = \left\{ s_1^{(1)}, s_2^{(2)}, \ldots, s_{i-1}^{(2)}, s_{i+1}^{(1)}, \ldots, s_m^{(2)} \right\},$$

for some $i \in \{1, \ldots, m\}$. Then any extensions of $\mathcal{T}_0$ consists of placing any further estimated changes in any of the regions between the changes above with the exception of the region $\left(s_{i-1}^{(2)}, s_{i+1}^{(1)}\right)$. Let $\mathcal{T}_0'$ be an arbitrary such extension, and again let $\mathcal{A}$ be any region between two consecutive points of $\mathcal{T}_0$ which contains a point found only in $\mathcal{T}_0'$. Then again letting

$$\text{Diff} := \text{RSS}(y_{\mathcal{A}}; \mathcal{T}_0) - \text{RSS}(y_{\mathcal{A}}; \mathcal{T}_0'),$$

then for $k > 0$ changes in the region $\mathcal{A}$, Diff is distributed as χ^2_{k+1}. We can again extend this argument to extensions of other elements of $\mathcal{GS}_0$ to conclude that segmentations with more than $2m$ changes which have no estimated changepoints between two consecutive true changes in at least one case will be uniformly rejected in favor of an element of $\mathcal{GS}_0$.

Therefore, as any segmentation with more than $2m$ changes for any core is an extension of an element of $\mathcal{GS}_0$, $\mathcal{GS}_1$, or $\mathcal{GS}_2$ (as such a segmentation must contain a region between two consecutive true changes with at least three estimated changes), then across all cores, a segmentation must be picked from within one of the classes $\mathcal{GS}_0$, $\mathcal{GS}_1$, or $\mathcal{GS}_2$ in probability. Thus, the maximum number of estimated changepoints that a core can return in the Deal procedure is $2m$.

The number of candidates returned for the merge phase of the Deal procedure is therefore bounded in probability by $2mL(n)$, so that the maximum computation time of the merge phase is $\mathcal{O}_p\left(L(n)^2\right)$ in the worst case, giving the total worst case computation time for the whole procedure. □

Supplementary Materials

chunk: R function for implementation of the Chunk procedure. (.R file)
Chunk_functions: Background functions called by chunk. (.R file)
cost_calculator: Calculates the cost of a particular fit returned by any of the method. (.R file)
cost_functions: Library of cost functions which can be entered as arguments into the main executions of the methods for various changepoint problems (e.g., Gaussian change in mean, change in rate parameter in the exponential setting). (.R file)
dataset_generation: Function used to generate the simulated data on which the methods were tested in this article. (.R file)
deal: R function for implementation of the Deal procedure. (.R file)
Deal_functions: Background functions called by deal. (.R file)
extras: Contains several intermediate functions called in the course of simulation study. (.R file)
PELT: R function for implementation of the unparallelized PELT procedure. (.R file)
performance_measure: Function called in the course of the simulation study which returns many of the performance metrics recorded in this article. (.R file)
README: Short instruction file detailing the best use of the accompanying .R files.
simstudyPAPERwithcost: Contains the main function for running the simulation study as seen in this article. (.R file)
supplementarymaterials: Proofs of several of the results stated in this paper. (.pdf file)

Acknowledgments

The authors would like to thank the reviewers for extremely helpful comments and feedback.

Funding

Tickle is grateful for the support of the EPSRC (grant number EP/L015692/1), while Eckley and Fearnhead gratefully acknowledge the financial support of EPSRC grant EP/N031938/1. The authors also acknowledge British Telecommunications plc (BT) for financial support, and are grateful to Kjeld Jensen and Dave Yearling in BT Research & Innovation for helpful discussions.

References

Alba, E. (2005), *Parallel Metaheuristics*, Hoboken, NJ: Wiley. [149]

Baranowski, R., Chen, Y., and Fryzlewicz, P. (2018), "Narrowest-Over-Threshold Change-Point Detection," arXiv no. 1609.00293v2, pp. 1–62. [149]

Baranowski, R., and Fryzlewicz, P. (2015), "wbs: Wild Binary Segmentation for Multiple Change-Point Detection," Version 1.3. [154]

Calaway, R., and Weston, S. (2017), "foreach: Provides Foreach Looping Construct for R," Version 1.4.4. [157]

Calaway, R., Weston, S., and Tenenbaum, D. (2018), "doParallel: Foreach Parallel Adaptor for the 'Parallel' Package," Version 1.0.14. [157]

Chen, F., and Nkurunziza, S. (2017), "On Estimation of the Change Points in Multivariate Regression Models With Structural Changes," *Communications in Statistics—Theory and Methods*, 46, 7157–7173. [149]

Chen, J., and Gupta, A. K. (2000), *Parametric Statistical Changepoint Analysis*, Boston, MA: Birkhäuser. [150]

Fearnhead, P., and Rigaill, G. (2017), "Changepoint Detection in the Presence of Outliers," arXiv no. 1609.07363v2, pp. 1–29. [150]

Fryzlewicz, P. (2014), "Wild Binary Segmentation for Multiple Change-Point Detection," *The Annals of Statistics*, 42, 2243–2281. [149,153]

Haynes, K., Eckley, I., and Fearnhead, P. (2017), "Computationally Efficient Changepoint Detection for a Range of Penalties," *Journal of Computational and Graphical Statistics*, 26, 134–143. [150]

Jackson, B., Scargle, J., Barnes, D., Arabhi, S., Alt, A., Gioumousis, P., Gwin, E., Sangtrakulcharoen, P., Tan, L., and Tsai, T. (2005), "An Algorithm for Optimal Partitioning of Data on an Interval," *IEEE Signal Processing*, 12, 105–108. [149,150]

Killick, R., Fearnhead, P., and Eckley, I. (2012), "Optimal Detection of Changepoints With a Linear Computational Cost," *Journal of the American Statistical Association*, 107, 1590–1598. [149,150,152]

Laurent, B., and Massart, P. (2000), "Adaptive Estimation of a Quadratic Functional by Model Selection," *The Annals of Statistics*, 28, 1302–1338. [159]

Lu, Z., Banerjee, M., and Michailidis, G. (2018), "Intelligent Sampling and Inference for Multiple Change Points in Extremely Long Data Sequences," arXiv no. 1710.07420v2, pp. 1–43. [158]

Mezmaz, M., Melab, M., Kessaci, Y., Lee, Y., Talbi, E.-G., Zomaya, A., and Tuyttnes, D. (2011), "A Parallel Bi-objective Hybrid Metaheuristic for Energy-Aware Scheduling for Cloud Computing Systems," *Journal of Parallel and Distributed Computing*, 71, 1497–1508. [149]

Olshen, A., Venkatraman, E., Lucito, R., and Wigler, M. (2004), "Circular Binary Segmentation for the Analysis of Array-Based DNA Copy Number Data," *Biostatistics*, 5, 557–572. [149]

Rigaill, G., Hocking, T. D., Bach, F., and Vert, J. P. (2013), "Learning Sparse Penalties for Change-Point Detection Using Max Margin Interval Regression," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, eds. S. Dasgupta and D. McAllester. [150]

Rigaill, G., Lebarbier, E., and Robin, S. (2012), "Exact Posterior Distributions and Model Selection Criteria for Multiple Change-Point Detection Problems," *Statistics and Computing*, 22, 917–929. [149]

Schmid, N., Christ, C., Christen, M., Eichenberger, A., and van Gunsteren, W. (2012), "Architecture, Implementation and Parallelisation of the GROMOS Software for Biomolecular Simulation," *Computer Physics Communications*, 183, 890–903. [149]

Scott, A., and Knott, M. (1974), "A Cluster Analysis Method for Grouping Means in the Analysis of Variance," *Biometrics*, 30, 507–512. [149]

Truong, C., Gudre, L., and Vayatis, N. (2017), "Penalty Learning for Changepoint Detection," in *Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece. [150]

Truong, C., Oudre, L., and Vayatis, N. (2018), "A Review of Changepoint Detection Methods," arXiv no. 1801.00718, pp. 1–31. [149]

Wang, X., and Dunson, D. B. (2014), "Parallelizing MCMC via Weierstrass Sampler," arXiv no. 1312.4605v2, pp. 1–35. [149]

Yao, Y.-C. (1988), "Estimating the Number of Change-Points via Schwarz' Criterion," *Statistics & Probability Letters*, 6, 181–189. [153,159]

Yao, Y.-C., and Au, S. T. (1989), "Least-Squares Estimation of a Step Function," *Sankhya*, 51, 370–381. [150,159]