Approximation Algorithms for Speeding up Dynamic Programming and Denoising aCGH data

CHARALAMPOS E. TSOURAKAKIS, Carnegie Mellon University
RICHARD PENG, Carnegie Mellon University
MARIA A. TSIARLI, University of Pittsburgh
GARY L. MILLER, Carnegie Mellon University
RUSSELL SCHWARTZ, Carnegie Mellon University

The development of cancer is largely driven by the gain or loss of subsets of the genome, promoting uncontrolled growth or disabling defenses against it. Denoising array-based Comparative Genome Hybridization (aCGH) data is an important computational problem central to understanding cancer evolution. In this work we propose a new formulation of the denoising problem which we solve with a "vanilla" dynamic programming algorithm which runs in $O(n^2)$ units of time. Then, we propose two approximation techniques. Our first algorithm reduces the problem into a well-studied geometric problem, namely halfspace emptiness queries, and provides an ϵ additive approximation to the optimal objective value in $\tilde{O}(n^{\frac{4}{3}+\delta}\log(\frac{U}{\epsilon}))$ time, where δ is an arbitrarily small positive constant and $U = \max\{\sqrt{C}, (|P_i|)_{i=1,\ldots,n}\}$ ($P = (P_1, P_2, \ldots, P_n)$, $P_i \in \mathbb{R}$, is the vector of the noisy aCGH measurements, C a normalization constant). The second algorithm provides a $(1\pm\epsilon)$ approximation (multiplicative error) and runs in $O(n \log n/\epsilon)$ time. The algorithm decomposes the initial problem into a small (logarithmic) number of Monge optimization subproblems which we can solve in linear time using existing techniques.

Finally, we validate our model on synthetic and real cancer datasets. Our method consistently achieves superior precision and recall to leading competitors on the data with ground truth. In addition, it finds several novel markers not recorded in the benchmarks but supported in the oncology literature.

Availability: Code, datasets and results available from the first author's web page `http://www.math.cmu.edu/~ctsourak/DPaCGHsupp.html`.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems; G.2.1 [**Combinatorics**]: Combinatorial Algorithms; J.3 [**Life and Medical Sciences**]: Biology and genetics

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Approximation algorithms, dynamic programming, halfspace range queries, computational biology, denoising aCGH data, time series segmentation

---

C.E.T was supported in this work by NSF grant ccf1013110 and by U.S. National Institute of Health award 1R01CA140214. R.P. was supported in this work by Natural Sciences and Engineering Research Council of Canada (NSERC), under Grant PGS M-377343-2009. G.L.M. was supported in this work by the National Science Foundation under Grant No. CCF-0635257. R.S. was supported in this work by U.S. National Institute of Health award 1R01CA140214. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Institute of Health and National Science Foundation, or other funding parties.
Author's addresses: C.E. Tsourakakis, Department of Mathematical Sciences, Carnegie Mellon University; R. Peng, School of Computer Science, Carnegie Mellon University; M. Tsiarli, Center for Neuroscience, University of Pittsburgh; G.L. Miller, School of Computer Science, Carnegie Mellon University; R. Schwartz, Department of Biological Sciences and School of Computer Science, Carnegie Mellon University.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

ACM Reference Format:
Tsourakakis C.E., Peng R., Tsiarli M.A., Miller G.L., Schwartz R. 2011. Approximation Algorithms for Speeding up Dynamic Programming and Denoising aCGH data. ACM J. Exp. Algor. 9, 4, Article 39 (March 2011), 27 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Tumorigenesis is a complex phenomenon often characterized by the successive acquisition of combinations of genetic aberrations that result in malfunction or disregulation of genes. There are many forms of chromosome aberration that can contribute to cancer development, including polyploidy, aneuploidy, interstitial deletion, reciprocal translocation, non-reciprocal translocation, as well as amplification, again with several different types of the latter (e.g., double minutes, HSR and distributed insertions [Albertson and Pinkel 2003]). Identifying the specific recurring aberrations, or sequences of aberrations, that characterize particular cancers provides important clues about the genetic basis of tumor development and possible targets for diagnostics or therapeutics. Many other genetic diseases are also characterized by gain or loss of genetic regions, such as Down Syndrome (trisomy 21) [Lejeune et al. 1959], Cri du Chat (5p deletion) [Lejeune et al. 1963], and Prader-Willi syndrome (deletion of 15q11-13) [Butler et al. 1986] and recent evidence has begun to suggest that inherited copy number variations are far more common and more important to human health than had been suspected just a few years ago [Zhang et al. 2009]. These facts have created a need for methods for assessing DNA copy number variations in individual organisms or tissues.

In this work, we focus specifically on array-based comparative genomic hybridization (aCGH) [Bignell et al. 2004; Pollack et al. 1999; Kallioniemi et al. 1992; Pinkel et al. 1998], a method for copy number assessment using DNA microarrays that remains, for the moment, the leading approach for high-throughput typing of copy number abnormalities. The technique of aCGH is schematically represented in Figure 1. A test and a reference DNA sample are differentially labeled and hybridized to a microarray and the ratios of their fluorescence intensities is measured for each spot. A typical output of this process is shown in Figure 1 (3), where the genomic profile of the cell line GM05296 [Snijders et al. 2001] is shown for each chromosome. The x-axis corresponds to the genomic position and the y-axis corresponds to a noisy measurement of the ratio $\log_2 \frac{T}{R}$ for each genomic position, typically referred to as "probe" by biologists. For healthy diploid organisms, R=2 and T is the DNA copy number we want to infer from the noisy measurements. For more details on the use of aCGH to detect different types of chromosomal aberrations, see [Albertson and Pinkel 2003].

Converting raw aCGH log fluorescence ratios into discrete DNA copy numbers is an important but non-trivial problem. Finding DNA regions that consistently exhibit chromosomal losses or gains in cancers provides a crucial means for locating the specific genes involved in development of different cancer types. It is therefore important to distinguish, when a probe shows unusually high or low fluorescence, whether that aberrant signal reflects experimental noise or a probe that is truly found in a segment of DNA that is gained or lost. Furthermore, successful discretization of array CGH data is crucial for understanding the process of cancer evolution, since discrete inputs are required for a large family of successful evolution algorithms, e.g., [Desper et al. 1999; 2000]. It is worth noting that manual annotation of such regions, even if possible [Snijders et al. 2001], is tedious and prone to mistakes due to several sources of noise (impurity of test sample, noise from array CGH method, etc.).

Based on the well-established observation that near-by probes tend to have the same DNA copy number, we formulate the problem of denoising aCGH data as the problem of approximating a signal P with another signal F consisting of a few piecewise constant

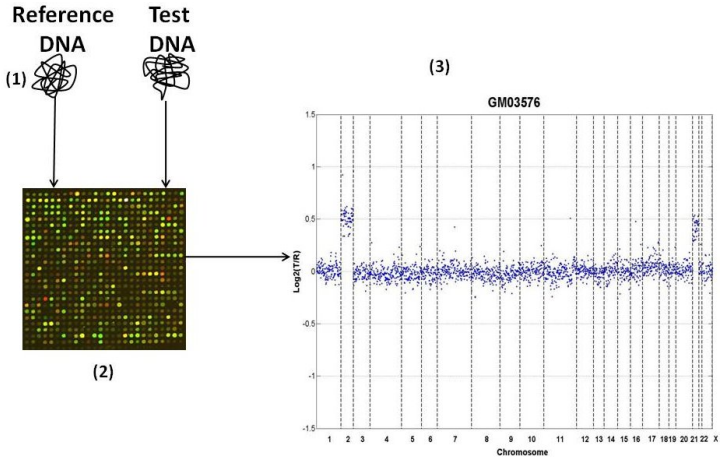


Fig. 1. Schematic representation of array CGH. Genomic DNA from two cell populations (1) is differentially labeled and hybridized in a microarray (2). Typically the reference DNA comes from a normal subject. For humans this means that the reference DNA comes from a normal diploid genome. The ratios on each spot are measured and normalised so that the median $\log_2$ ratio is zero. The final result is an ordered tuple containing values of the fluorescent ratios in each genomic position per each chromosome. This is shown in (3) where we see the genomic profile of the cell line GM05296 [Snijders et al. 2001]. The problem of denoising array CGH data is to infer the true DNA copy number T per genomic position from a set of noisy measurements of the quantity $\log_2 \frac{T}{R}$, where R=2 for normal diploid humans.

segments. Specifically, let $P = (P_1, P_2, \ldots, P_n) \in \mathbb{R}^n$ be the input signal -in our setting the sequence of the noisy aCGH measurements- and let C be a constant. Our goal is to find a function $F : [n] \to \mathbb{R}$ which optimizes the following objective function:

$$\min_F \sum_{i=1}^{n} (P_i - F_i)^2 + C \times (|\{i : F_i \neq F_{i+1}\}| + 1). \tag{1}$$

The best known exact algorithm for solving the optimization problem defined by Equation 1 runs in $O(n^2)$ time but as our results suggest this running time is likely not to be tight. It is worth noting that existing techniques for speeding up dynamic programming [Yao 1982; Eppstein et al. 1988; Eppstein et al. 1992a] do not apply to our problem. In this work, we provide two approximation algorithms for solving this recurrence. The first algorithm approximates the objective within an additive error ϵ which we can make as small as we wish and its key idea is the reduction of the problem to halfspace range queries, a well studied computational geometric problem [Agarwal and Erickson 1999]. The second algorithm carefully decomposes the problem into a "small" (logarithmic) number of subproblems which satisfy the quadrangle inequality (Monge property). The main contributions of this work can be summarized as follows.

— We propose a new formulation of the aCGH denoising problem which we solve using a dynamic programming algorithm in $O(n^2)$ time.
— We provide a technique which approximates the optimal value of our objective function within additive ϵ error and runs in $\tilde{O}(n^{\frac{4}{3}+\delta} \log(\frac{U}{\epsilon}))$ time, where δ is an arbitrarily small positive constant and $U = \max\{\sqrt{C}, (|P_i|)_{i=1,\ldots,n}\}$.
— We provide a technique for approximate dynamic programming which solves the corresponding recurrence within a multiplicative factor of (1+ϵ) and runs in $O(n \log n/\epsilon)$.

— We validate our proposed model on both synthetic and real data. Specifically, our segmentations result in superior precision and recall compared to leading competitors on benchmarks of synthetic data and real data from the Coriell cell lines. In addition, we are able to find several novel markers not recorded in the benchmarks but supported in the oncology literature.

The remainder of this paper is organized as follows: Section 2 briefly presents the related work. Section 3 presents the vanilla dynamic programming algorithm which runs in $O(n^2)$. Section 4 analyzes properties of the recurrence which will be exploited in Sections 5 and 6 where we describe the additive and multiplicative approximation algorithms respectively. In Section 7 we validate our model by performing an extensive biological analysis of the findings of our segmentation. Finally, in Section 8 we conclude.

2. RELATED WORK

In Section 2.1, we provide a brief overview of the existing work on the problem of denoising aCGH data. In Section 2.2, we present existing techniques for speeding up dynamic programming and in Section 2.3 we discuss the problem of reporting points in halfspaces.

2.1. Denoising aCGH data

Many algorithms and objective functions have been proposed for the problem of discretizing and segmenting aCGH data. Many methods, starting with [Fridlyand et al. 2004], treat aCGH segmentation as a hidden Markov model (HMM) inference problem. The HMM approach has since been extended in various ways, e.g., through the use of Bayesian HMMs [Guha et al. 2006], incorporation of prior knowledge of locations of DNA copy number polymorphisms [Shah et al. 2006], and the use of Kalman filters [Shi et al. 2007]. Other approaches include wavelet decompositions [Hsu et al. 2005], quantile regression [Eilers and de Menezes 2005], expectation-maximization in combination with edge-filtering [Myers et al. 2004], genetic algorithms [Jong et al. 2004], clustering-based methods [Xing et al. 2007; Wang et al. 2005], variants on Lasso regression [Tibshirani and Wang 2008; Huang et al. 2005], and various problem-specific Bayesian [Barry and Hartigan 1993], likelihood [Hupé et al. 2004], and other statisical models [Lipson et al. 2005]. A dynamic programming approach, in combination with expectation maximimization, has been previously used by Picard et al. [Picard et al. 2007]. In [Lai et al. 2005] and [Willenbrock and Fridlyand 2005] an extensive experimental analysis of available methods has been conducted. Two methods stand out as the leading approaches in practice. One of these top methods is CGHSEG [Picard et al. 2005], which assumes that a given CGH profile is a Gaussian process whose distribution parameters are affected by abrupt changes at unknown coordinates/breakpoints. The other method which stands out for its performance is Circular Binary Segmentation [Olshen et al. 2004] (CBS), a modification of binary segmentation, originally proposed by Sen and Srivastava [Sen and Srivastava 1975], which uses a statistical comparison of mean expressions of adjacent windows of nearby probes to identify possible breakpoints between segments combined with a greedy algorithm to locally optimize breakpoint positions.

2.2. Speeding up Dynamic Programming

Dynamic programming is a powerful problem solving technique introduced by Bellman [Bellman 2003] with numerous applications in biology, e.g., [Picard et al. 2005; Hirschberg 1975; Waterman and Smith 1986], in control theory, e.g., [Bertsekas 2000], in operations research and many other fields. Due to its importance, a lot of re-

search has focused on speeding up basic dynamic programming implementations. A successful example of speeding up a naive dynamic programming implementation is the computation of optimal binary search trees. Gilbert and Moore solved the problem efficiently using dynamic programming [Gilbert and Moore 1959]. Their algorithm runs in $O(n^3)$ time and for several years this running time was considered to be tight. In 1971 Knuth [Knuth 1971] showed that the same computation can be carried out in $O(n^2)$ time. This remarkable result was generalized by Frances Yao in [Yao 1982; 1980]. Specifically, Yao showed that this dynamic programming speedup technique works for a large class of recurrences. She considered the recurrence $c(i,i) = 0$, $c(i,j) = \min_{i<k\leq j}(c(i,k-1) + c(k,j)) + w(i,j)$ for $i < j$ where the weight function w satisfies the quadrangle inequality (see Section 2.4) and proved that the solution of this recurrence can be found in $O(n^2)$ time. Eppstein, Galil and Giancarlo have considered similar recurrences where they showed that naive $O(n^2)$ implementations of dynamic programming can run in $O(n \log n)$ time [Eppstein et al. 1988]. Larmore and Schieber [Larmore and Schieber 1991] further improved the running time, giving a linear time algorithm when the weight function is concave. Klawe and Kleitman give in [Klawe and Kleitman 1990] an algorithm which runs in $O(n\alpha(n))$ time when the weight function is convex, where $\alpha(\cdot)$ is the inverse Ackermann function. Furthermore, Eppstein, Galil, Giancarlo and Italiano have also explored the effect of sparsity [Eppstein et al. 1992a; 1992b], another key concept in speeding up dynamic programming. Aggarwal, Klawe, Moran, Shor, Wilber developed an algorithm, widely known as the SMAWK algorithm, [Aggarwal et al. 1986] which can compute in $O(n)$ time the row maxima of a totally monotone $n \times n$ matrix. The connection between the Knuth-Yao technique and the SMAWK algorithm was made clear in [Bein et al. 2009], by showing that the Knuth-Yao technique is a special case of the use of totally monotone matrices. The basic properties which allow these speedups are the convexity or concavity of the weight function. Such properties date back to Monge [Monge 1781] and are well studied in the literature, see for example [Burkard et al. 1996].

Close to our work lies the work on histogram construction, an important problem for database applications. Jagadish et al. [Jagadish et al. 1998] originally provided a simple dynamic programming algorithm which runs in $O(kn^2)$ time, where k is the number of buckets and n the input size and outputs the best V-optimal histogram. Guha, Koudas and Shim [Guha et al. 2006] propose a $(1+\epsilon)$ approximation algorithm which runs in linear time. Their algorithms exploits monotonicity properties of the key quantities involved in the problem. Our $(1+\epsilon)$ approximation algorithm in Section 2.4 uses a decomposition technique similar to theirs.

2.3. Reporting Points in a Halfspace

Let S be a set of points in $\mathbb{R}^d$ and let k denote the size of the output, i.e., the number of points to be reported. Consider the problem of preprocessing S such that for any halfspace query γ we can report efficiently whether the set $S \cap \gamma$ is empty or not. This problem is a well studied special case of the more general range searching problem. For an extensive survey see the work by Agarwal and Erickson [Agarwal and Erickson 1999]. For $d = 2$, the problem has been solved optimally by Chazelle, Guibas and Lee [Chazelle et al. 1985]. For $d = 3$, Chazelle and Preparata in [Chazelle and Preparata 1985] gave a solution with nearly linear space and $O(\log n + k)$ query time, while Aggarwal, Hansen and Leighton [Aggarwal et al. 1990] gave a solution with a more expensive preprocessing but $O(n \log n)$ space. When the number of dimensions is greater than 4, i.e., $d \geq 4$, Clarkson and Shor [Clarkson and Shor. 1989] gave an algorithm that requires $O(n^{\lfloor d/2 \rfloor + \epsilon})$ preprocessing time and space, where ϵ is an arbitrarily small positive constant, but can subsequently answer queries in $O(\log n + k)$ time. Matoušek in [Matousek 1991] provides improved results on the problem, which

are used by Agarwal, Eppstein, Matoušek [Agarwal et al. 1992] in order to create dynamic data structures that trade off insertion and query times. We refer to Theorem 2.1(iii) of their paper [Agarwal et al. 1992]:

THEOREM 2.1 (AGARWAL, EPPSTEIN, MATOUŠEK [AGARWAL ET AL. 1992]). *Given a set S of n points in $\mathbb{R}^d$ where $d \geq 3$ and a parameter m between n and $n^{\lfloor \frac{d}{2} \rfloor}$ the halfspace range reporting problem can be solved with the following performance: $O(\frac{n}{m^{1/\lfloor d/2 \rfloor}} \log n)$ query time, $O(m^{1+\epsilon})$ space and preprocessing time, $O(m^{1+\epsilon}/n)$ amortized update time.*

Substituting for $d = 4$, $m = n^{\frac{4}{3}}$ we obtain the following corollary, which will be used as a subroutine in our proposed method:

COROLLARY 2.2. *Given a set S of n points in $\mathbb{R}^4$ the halfspace range reporting problem can be solved with $O(n^{\frac{1}{3}} \log n)$ query time, $O(n^{\frac{4}{3}+\delta})$ space and preprocessing time, and $O(n^{\frac{1}{3}+\delta})$ update time, where δ is an arbitrarily small positive constant.*

2.4. Monge Functions and Dynamic Programming

Here, we refer to one of the results in [Larmore and Schieber 1991] which we use in Section 2.4 as a subroutine for our proposed method. A function w defined on pairs of integer indices is Monge (concave) if for any 4-tuple of indices $i_1 < i_2 < i_3 < i_4$, $w(i_1, i_4) + w(i_2, i_3) \geq w(i_1, i_3) + w(i_2, i_4)$. Furthermore, we assume that f is a function such that the values $f(a_j)$ for all j are easily evaluated. The following results holds:

THEOREM 2.3 ([LARMORE AND SCHIEBER 1991]). *Consider the one dimensional recurrence $a_i = \min_{j<i}\{f(a_j) + w(j, i)\}$ for $i = 1, \ldots, n$, where the basis a_0 is given. There exists an algorithm which solves the recurrence online in $O(n)$ time*[1].

3. $O(N^2)$ DYNAMIC PROGRAMMING ALGORITHM

In order to solve the optimization problem defined by Equation 1, we define the key quantity OPT_i given by the following recurrence:

$$
\begin{aligned}
&OPT_0 = 0 \\
&OPT_i = \min_{0 \leq j \leq i-1} [OPT_j + w(i, j)] + C, \text{ for } i > 0 \\
&\text{where } w(i, j) = \sum_{k=j+1}^{i} \left(P_k - \frac{\sum_{m=j+1}^{i} P_m}{i - j} \right)^2 .
\end{aligned}
$$

The above recurrence has a straightforward interpretation: OPT_i is equal to the minimum cost of fitting a set of piecewise constant segments from point P_1 to P_i given that index j is a breakpoint. The cost of fitting the segment from $j + 1$ to i is C. The weight function $w()$ is the minimum squared error for fitting a constant segment on points $\{P_{j+1}, \ldots, P_i\}$, which is obtained for the constant segment with value $\frac{\sum_{m=j+1}^{i} P_m}{i-j}$, i.e., the average of the points in the segment. This recursion directly implies a simple dynamic programming algorithm. We call this algorithm CGHTRIMMER and the pseudocode is shown in Algorithm 1. The main computational bottleneck of CGHTRIMMER is the computation of the auxiliary matrix M, an upper diagonal matrix for which m_{ij}

[1]Thus, obtaining $O(n)$ speedup compared to the straight-forward dynamic programming algorithm which runs in $O(n^2)$ units of time.

is the minimum squared error of fitting a segment from points $\{P_i, \ldots, P_j\}$. To avoid a naïve algorithm that would simply find the average of those points and then compute the squared error, resulting in $O(n^3)$ time, we use Claim 1.

ALGORITHM 1: CGHTRIMMER algorithm

Input: Signal $P = (P_1, \ldots, P_n)$, Regularization parameter C
Output: Optimal Segmentation with respect to our objective (see Equation 1)

`/* Compute an` $n \times n$ `matrix` M, `where` $M_{ji} = \sum_{k=j}^{i} \left(P_k - \frac{\sum_{m=j}^{i} P_m}{i-j+1} \right)^2$. `*/`

`/*` A `is an auxiliary matrix of averages, i.e.,` $A_{ji} = \frac{\sum_{k=j}^{i} P_k}{i-j+1}$. `*/`

Initialize matrix $A \in \mathbb{R}^{n \times n}$, $A_{ij} = 0, i \neq j$ and $A_{ii} = P_i$.
for $i = 1$ *to* n **do**
  for $j = i + 1$ *to* n **do**
    $A_{i,j} \leftarrow \frac{j-i}{j-i+1} A_{i,j-1} + \frac{1}{j-i+1} P_j$
  end
end
for $i = 1$ *to* n **do**
  for $j = i + 1$ *to* n **do**
    $M_{i,j} \leftarrow M_{i,j-1} + \frac{j-i}{j-i+1} (P_j - A_{i,j-1})^2$
  end
end
`/* Solve the Recurrence.` `*/`
for $i = 1$ *to* n **do**
  $\text{OPT}_i \leftarrow \min_{0 \leq j \leq i-1} OPT_j + M_{j+1,i} + C$
  $\text{BREAK}_i \leftarrow \arg\min_{1 \leq j \leq i} OPT_{j-1} + M_{j,i} + C$
end

CLAIM 1. *Let $\alpha_{(j)}$ and $m_{(j)}$ be the average and the minimum squared error of fitting a constant segment to points $\{P_1, \ldots, P_j\}$ respectively. Then,*

$$\alpha_{(j)} = \frac{j-1}{j} \alpha_{(j-1)} + \frac{1}{j} P_j, \tag{2}$$

$$m_{(j)} = m_{(j-1)} + \frac{j-1}{j} (P_j - \alpha_{(j-1)})^2. \tag{3}$$

The interested reader can find a proof of Claim 1 in [Knuth 1981]. Equations 2 and 3 provide us a way to compute means and least squared errors online. Algorithm 1 first computes matrices A and M using Equations 2, 3 and then iterates (last *for* loop) to solve the recurrence by finding the optimal breakpoint for each index i. The total running time is $O(n^2)$ (matrices A and M matrices have $O(n^2)$ entries and each requires $O(1)$ time to compute). Obviously, Algorithm 1 uses $O(n^2)$ units of space.

4. ANALYSIS OF THE TRANSITION FUNCTION

In the following, let $S_i = \sum_{j=1}^{i} P_j$. The transition function for the dynamic programming for $i > 0$ can be rewritten as:

$$OPT_i = \min_{j<i} OPT_j + \sum_{m=j+1}^{i} P_m^2 - \frac{(S_i - S_j)^2}{i-j} + C. \tag{4}$$

The transition can be viewed as a weight function $w(j,i)$ that takes the two indices j and i as parameters such that:

$$w(j,i) = \sum_{m=j+1}^{i} P_m^2 - \frac{(S_i - S_j)^2}{i-j} + C \tag{5}$$

Note that the weight function does not have the Monge property, as demonstrated by the vector $P = (P_1, \ldots, P_{2k+1}) = (1,2,0,2,0,2,0,\ldots 2,0,1)$. When $C = 1$, the optimal choices of j for $i = 1, \ldots, 2k$ are $j = i-1$, i.e., we fit one segment per point. However, once we add in $P_{2k+1} = 1$ the optimal solution changes to fitting all points on a single segment. Therefore, preferring a transition to j_1 over one to j_2 at some index i does not allow us to discard j_2 from future considerations. This is one of the main difficulties in applying techniques based on the increasing order of optimal choices of j, such as the method of Eppstein, Galil and Giancarlo [Eppstein et al. 1988] or the method of Larmore and Schieber [Larmore and Schieber 1991], to reduce the complexity of the $O(n^2)$ algorithm we described in Section 3.

We start by defining DP_i for $i = 0,1,..,n$, the solution to a simpler optimization problem.

Definition 4.1. Let DP_i, $i = 0,1,..,n$, satisfy the following recurrence

$$DP_i = \begin{cases} \min_{j<i} DP_j - \frac{(S_i - S_j)^2}{i-j} + C & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases} \tag{6}$$

The following observation stated as Lemma 4.2 plays a key role in Section 5.

LEMMA 4.2. *For all i, OPT_i can be written in terms of DP_i as*

$$OPT_i = DP_i + \sum_{m=1}^{i} P_m^2.$$

PROOF. We use strong induction on i. For $i = 0$ the result trivially holds. Let the result hold for all $j < i$. Then,

$$\begin{aligned} DP_i &= \min_{j<i} DP_j - \frac{(S_i - S_j)^2}{i-j} + C \\ &= \min_{j<i} OPT_j - \frac{(S_i - S_j)^2}{i-j} + \sum_{m=j+1}^{i} P_m^2 - \sum_{m=1}^{i} P_m^2 + C \\ &= OPT_i - \sum_{m=1}^{i} P_m^2 \end{aligned}$$

Hence, $OPT_i = DP_i + \sum_{m=1}^{i} P_m^2$ for all i. □

Observe that the second order moments involved in the expression of OPT_i are absent from DP_i. Let $\tilde{w}(j,i)$ be the shifted weight function, i.e., $\tilde{w}(j,i) = -\frac{(S_i - S_j)^2}{i-j} + C$. Clearly, $w(j,i) = \tilde{w}(j,i) + \sum_{m=i}^{j} P_m^2$.

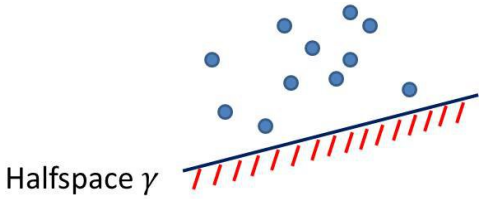


Fig. 2. Answering whether or not $\bar{DP}_i \leq x + C$ reduces to answering whether the point set $\{(j, \tilde{DP}_j, 2S_j, S_j^2 + \tilde{DP}_j j) \in \mathbb{R}^4, j < i\}$ has a non-empty intersection with the halfspace $\gamma = \{y \in \mathbb{R}^4 : a_i y \leq b_i\}$ where a_i and b_i are a 4-dimensional constant vector and a constant which depend on i respectively. This type of queries can be solved efficiently, see [Agarwal et al. 1992].

5. ADDITIVE APPROXIMATION USING HALFSPACE QUERIES

In this Section, we present a novel algorithm which runs in $\tilde{O}(n^{\frac{4}{3}+\delta} \log(\frac{U}{\epsilon}))$ time and approximates the optimal objective value within additive ϵ error. We derive the algorithm gradually in the following and upon presenting the necessary theory we provide the pseudocode (see Algorithm 2) at the end of this Section. Our proposed method uses the results of [Agarwal et al. 1992] as stated in Corollary 2.2 to obtain a fast algorithm for the additive approximation variant of the problem. Specifically, the algorithm initializes a 4-dimensional halfspace query data structure. The algorithm then uses binary searches to compute an accurate estimate of the value DP_i for $i = 1, \ldots, n$. As errors are introduced at each term, we use $\tilde{DP}_i$ to denote the approximate value of DP_i calculated by the binary search, and $\bar{DP}_i$ to be the optimum value of the transition function computed by examining the approximate values $\tilde{DP}_j$ for all $j < i$. Formally,

$$\bar{DP}_i = \min_{j<i} \left[\tilde{DP}_j - \underbrace{\frac{(S_i - S_j)^2}{i - j}}_{\tilde{w}(j,i)} \right] + C.$$

Since the binary search incurs a small additive error at each step, it remains to show that these errors accumulate in a controlled way. Theorem 5.1 states that a small error at each step suffices to give an overall good approximation. We show inductively that if $\tilde{DP}_i$ approximates $\bar{DP}_i$ within ϵ/n, then $\tilde{DP}_i$ is within $i\epsilon/n$ additive error from the optimal value DP_i for all i.

THEOREM 5.1. *Let $\tilde{DP}_i$ be the approximation of our algorithm to DP_i. Then, the following inequality holds:*

$$|DP_i - \tilde{DP}_i| \leq \frac{\epsilon i}{n} \tag{7}$$

PROOF.

We use induction on the number of points. Using the same notation as above, let $\bar{DP}_i = \min_{j<i} \tilde{DP}_j - w(j,i) + C$. By construction the following inequality holds:

$$|\bar{DP}_i - \tilde{DP}_i| \leq \frac{\epsilon}{n} \; \forall i = 1, \ldots, n \tag{8}$$

When $i = 1$ it is clear that $|DP_1 - \tilde{DP}_1| \leq \frac{\epsilon}{n}$. Our inductive hypothesis is the following:

$$|DP_j - \tilde{DP}_j| \leq \frac{j\epsilon}{n} \; \forall j < i \tag{9}$$

It suffices to show that the following inequality holds:

$$|DP_i - \bar{DP}_i| \leq \frac{(i-1)\epsilon}{n} \tag{10}$$

since then by the triangular inequality we obtain:

$$\frac{i\epsilon}{n} \geq |DP_i - \bar{DP}_i| + |\bar{DP}_i - \tilde{DP}_i| \geq |DP_i - \tilde{DP}_i|.$$

Let $j^*, \bar{j}$ be the optimum breakpoints for DP_i and $\bar{DP}_i$ respectively, $j^*, \bar{j} \leq i-1$.

$$\begin{aligned} DP_i &= DP_{j^*} + \tilde{w}(j^*, i) + C \\ &\leq DP_{\bar{j}} + \tilde{w}(\bar{j}, i) + C \\ &\leq \tilde{DP}_{\bar{j}} + \tilde{w}(\bar{j}, i) + C + \frac{\bar{j}\epsilon}{n} \text{(by 9)} \\ &= \bar{DP}_i + \frac{\bar{j}\epsilon}{n} \\ &\leq \bar{DP}_i + \frac{(i-1)\epsilon}{n} \end{aligned}$$

Similarly we obtain:

$$\begin{aligned} \bar{DP}_i &= \tilde{DP}_{\bar{j}} + \tilde{w}(\bar{j}, i) + C \\ &\leq \tilde{DP}_{j^*} + \tilde{w}(j^*, i) + C \\ &\leq DP_{j^*} + \tilde{w}(j^*, i) + C + \frac{j^*\epsilon}{n} \text{(by 9)} \\ &= DP_i + \frac{j^*\epsilon}{n} \\ &\leq DP_i + \frac{(i-1)\epsilon}{n} \end{aligned}$$

Combining the above two inequalities, we obtain 10. □

By substituting $i = n$ in Theorem 5.1 we obtain the following corollary which proves that $\tilde{DP}_n$ is within ϵ of DP_n.

COROLLARY 5.2. *Let $\tilde{DP}_n$ be the approximation of our algorithm to DP_n. Then,*

$$|DP_n - \tilde{DP}_n| \leq \epsilon. \tag{11}$$

To use the analysis above in order to come up with an efficient algorithm we need to answer two questions: (a) How many binary search queries do we need in order to obtain the desired approximation? (b) How can we answer each such query efficiently? The answer to (a) is simple: as it can easily be seen, the value of the objective function is upper bounded by $U^2 n$, where $U = \max\{\sqrt{C}, |P_1|, \ldots, |P_n|\}$. Therefore, $O(\log(\frac{U^2 n}{\epsilon/n})) = \tilde{O}(\log(\frac{U}{\epsilon}))$ iterations of binary search at each index i are sufficient to obtain the desired approximation. We reduce the answer to (b) to a well studied computational geometric problem. Specifically, fix an index i, where $1 \leq i \leq n$, and consider the general form of the binary search query $\bar{DP}_i \leq x + C$, where $x + C$ is the value on which we query. Note that we use the expression $x + C$ for convenience, i.e., so that the constant C will be simplified from both sides of the query. This query translates itself to the following

decision problem, see also Figure 2. Does there exist an index j, such that $j < i$ and the following inequality holds:

$$x \geq \tilde{DP}_j - \frac{(S_i - S_j)^2}{i-j} \Rightarrow xi + S_i^2 \geq (x, i, S_i, -1)(j, \tilde{DP}_j, 2S_j, S_j^2 + j\tilde{DP}_j)^T?$$

Hence, the binary search query has been reduced to answering a *halfspace query*. Specifically, the decision problem for any index i becomes whether the intersection of the point set $\text{POINTS}_i = \{(j, \tilde{DP}_j, 2S_j, S_j^2 + \tilde{DP}_j j) \in \mathbb{R}^4, j < i\}$ with a hyperplane is empty. By Corollary 2.2 [Agarwal et al. 1992], for a point set of size n, this can be done in $\tilde{O}(n^{\frac{1}{3}+\delta})$ per query and $O(n^{\frac{1}{3}} \log n)$ amortized time per insertion of a point. Hence, the optimal value of DP_i can be found within an additive constant of ϵ/n using the binary search in $\tilde{O}(n^{\frac{1}{3}} \log(\frac{U}{\epsilon}))$ time.

Therefore, we can proceed from index 1 to n, find the approximately optimal value of OPT_i and insert a point corresponding to it into the query structure. We obtain an algorithm which runs in $\tilde{O}(n^{\frac{4}{3}+\delta} \log(\frac{U}{\epsilon}))$ time, where δ is an arbitrarily small positive constant. The pseudocode is given in Algorithm 2.

ALGORITHM 2: Approximation within additive ϵ using 4D halfspace queries

```
Initialize 4D halfspace query structure Q
for i = 1 to n do
    low ← 0
    high ← nU^2
    while high − low > ε/n do
        m ← (low + high)/2
        /* This halfspace emptiness query is efficiently supported by Q */
        flag ← (∃j such that xi + S_i^2 ≥ (x, i, S_i, −1)(j, D̃P_j, 2S_j, S_j^2 + jD̃P_j)^T) if flag then
            high ← m
        else
            low ← m
        end
    end
    D̃P_i ← (low + high)/2
    /* Point insertions are efficiently supported by Q */
    Insert point (i, D̃P_i, 2S_i, S_i^2 + D̃P_i i) in Q
end
```

6. MULTISCALE MONGE DECOMPOSITION

In this Section we present an algorithm which runs in $O(n \log n/\epsilon)$ time to approximate the optimal shifted objective value within a multiplicative factor of $(1+\epsilon)$. Our algorithm is based on a new technique, which we consider of indepenent interest. Let $w'(j,i) = \sum_{m=j+1}^{i} (i-j)P_m^2 - (S_i - S_j)^2$. We can rewrite the weight function w as a function of w', namely as $w(j,i) = w'(j,i)/(i-j) + C$. The interested reader can easily check that we can express $w'(j,i)$ as a sum of non-negative terms, i.e.,

$$w'(j,i) = \sum_{j+1 \leq m_1 < m_2 \leq i} (P_{m_1} - P_{m_2})^2.$$

Table I. Summary of proof of Lemma 6.1.

	$w'(i_1,i_4)+w'(i_2,i_3)$	$w'(i_1,i_3)+w'(i_2,i_4)$
(S_1,S_1)	1	1
(S_2,S_2)	2	2
(S_3,S_3)	1	1
(S_1,S_2)	1	1
(S_1,S_3)	1	0
(S_2,S_3)	1	1

Recall from Section 3 that the weight function w is not Monge. The next lemma shows that the weight function w' is a Monge function.

LEMMA 6.1. *The weight function $w'(j,i)$ is Monge (concave), i.e., for any $i_1 < i_2 < i_3 < i_4$, the following holds:*

$$w'(i_1,i_4)+w'(i_2,i_3) \geq w'(i_1,i_3)+w'(i_2,i_4).$$

PROOF. Since each term in the summation is non-negative, it suffices to show that any pair of indices, (m_1,m_2) is summed as many times on the left hand side as on the right hand side. If $i_2+1 \leq m_1 < m_2 \leq i_3$, each term is counted twice on each side. Otherwise, each term is counted once on the left hand side since $i_1+1 \leq m_1 < m_2 \leq i_4$ and at most once on the right hand side since $[i_1+1,i_3] \cap [i_2+1,i_4] = [i_2+1,i_3]$. □

The proof of Lemma 6.1 is summarized in Table I. Specifically, let S_j be the set of indices $\{i_j+1,\ldots,i_{j+1}\}$, $j=1,2,3$. Also, let (S_j,S_k) denote the set of indices (m_1,m_2) which appear in the summation such that $m_1 \in S_j, m_2 \in S_k$. The two last columns of Table I correspond to the left- and right-hand side of the Monge inequality (as in Lemma 6.1) and contain the counts of appearances of each term.

Our approach is based on the following observations:

(1) Consider the weighted directed acyclic graph (DAG) on the vertex set $V = \{0,\ldots,n\}$ with edge set $E = \{(j,i) : j < i\}$ and weight function $w : E \to \mathbb{R}$, i.e., edge (j,i) has weight $w(j,i)$. Solving the aCGH denoising problem reduces to finding a shortest path from vertex 0 to vertex n. If we perturb the edge weights within a factor of $(1+\epsilon)$, as long as the weight of each edge is positive, then the optimal shortest path distance is also perturbed within a factor of at most $(1+\epsilon)$.
(2) By Lemma 6.1 we obtain that the weight function is not Monge essentially because of the $i-j$ term in the denominator.
(3) Our goal is to approximate w by a Monge function w' such that $c_1 w' \leq w \leq c_2 w'$ where c_1, c_2 should be known constants.

In the following we elaborate on the latter goal. Fix an index i and note that the optimal breakpoint for that index is some index $j \in \{1,\ldots,i-1\}$. We will "bucketize" the range of index j into $m = O(\log_{1+\epsilon}(i)) = O(\log n/\epsilon)$ buckets such that the k-th bucket, $k=1,\ldots,m$, is defined by the set of indices j which satisfy

$$l_k = i-(1+\epsilon)^k \leq j \leq i-(1+\epsilon)^{k-1} = r_k.$$

This choice of bucketization is based on the first two observations which guide our approach. Specifically, it is easy to check that $(1+\epsilon)^{k-1} \leq i-j \leq (1+\epsilon)^k$. This results, for any given i, to approximating $i-j$ by a constant for each possible bucket, leading to $O(\log n/\epsilon)$ different Monge functions (one per bucket) while incurring a multiplicative error of at most $(1+\epsilon)$. However, there exists a subtle point, as also Figure 3 indicates. We need to make sure that each of the Monge functions is appropriatelly defined so that when we consider the k-th Monge subproblem, the optimal breakpoint j_k should

satisfy $j_k \in [l_k, r_k]$. Having achieved that (see Lemma 6.2) we can solve efficiently the recurrence. Specifically, OPT_i is computed as follows:

$$\begin{aligned} OPT_i &= \min_{j<i}\left[OPT_j + \frac{w'(i,j)}{i-j}\right] + C \\ &= \min_{k}\left[\min_{j\in[l_k,r_k]} OPT_j + \frac{w'(i,j)}{i-j}\right] + C \\ &\approx \min_{k}\left[\min_{j\in[l_k,r_k]} OPT_j + \frac{w'(i,j)}{(1+\epsilon)^{k-1}}\right] + C \\ &= \min_{k}\left[\min_{j\in[l_k,r_k]} OPT_j + \frac{w'(i,j)}{c_k}\right] + C. \end{aligned}$$

The following is one of the possible ways to define the m Monge weight functions. In what follows, λ is a sufficiently large positive constant.

$$w_k(j,i) = \begin{cases} 2^{n-i+j}\lambda & i-j < c_k = (1+\epsilon)^{k-1} \\ 2^{i-j}\lambda & i-j > (1+\epsilon)c_k = (1+\epsilon)^k \\ w'(j,i)/c_k & \text{otherwise} \end{cases} \tag{12}$$

LEMMA 6.2. *Given any vector P, it is possible to pick λ such that w_k is Monge for all $k \geq 1$. That is, for any 4-tuple $i_1 < i_2 < i_3 < i_4$, $w_k(i_1,i_4) + w_k(i_2,i_3) \geq w_k(i_1,i_3) + w_k(i_2,i_4)$.*

PROOF. Since $w'(j,i) = \sum_{j+1\leq m_1<m_2\leq i}(P_{m_1} - P_{m_2})^2 \leq (2K)^2n^2$ where $K = \max_{1\leq i\leq n}|P_i|$, we can pick λ such that $w_k(j,i) \geq w'(j,i)$. The rest of the proof is case-work based on the lengths of the intervals i_3-i_1, i_4-i_2, and how they compare with c_k and $(1+\epsilon)c_k$. There are 12 such cases in total. We may assume $i_3-i_1 \leq i_4-i_2$ without loss of generality, leaving thus 6 cases to be considered.
If $c_k \leq i_3-i_1, i_4-i_2 \leq (1+\epsilon)c_k$, then:

$$\begin{aligned} w_k(i_1,i_3) + w_k(i_2,i_4) &= (1/c_k)(w'(i_1,i_3) + w'(i_2,i_4)) \\ &\leq (1/c_k)(w'(i_1,i_4) + w'(i_2,i_3)) \text{ (by Lemma 6.1)} \\ &\leq w_k(i_1,i_4) + w_k(i_2,i_3). \end{aligned}$$

If $i_3-i_1 < c_k$ and $i_4-i_2 \leq (1+\epsilon)c_k$. Then as $i_2 > i_1$, $i_3-i_2 \leq i_3-i_1-1$ and we have:

$$\begin{aligned} w_k(i_2,i_3) &= 2^{n-i_3+i_2}\lambda \\ &\geq 2\cdot 2^{n-i_3-i_1}\lambda \\ &\geq w_k(i_1,i_3) + w_k(i_2,i_4) \end{aligned}$$

The cases of $c_k \leq i_3-i_1$ and $c_k(1+\epsilon) < i_4-i_2$ can be done similarly. Note that the cases of $i_3-i_1, i_4-i_2 < c_k$ and $(1+\epsilon)c_k < i_3-i_1, i_4-i_2$ are also covered by these.
The only case that remain is $i_3-i_1 < c_k$ and $(1+\epsilon)c_k < i_4-i_2$. Since $i_3-i_2 < i_3-i_1 < c_k$, we have:

$$\begin{aligned} w_k(i_2,i_3) &= 2^{n-i_3+i_2}\lambda \\ &> 2^{n-i_4+i_2}\lambda \\ &= w_k(i_2,i_4) \end{aligned}$$

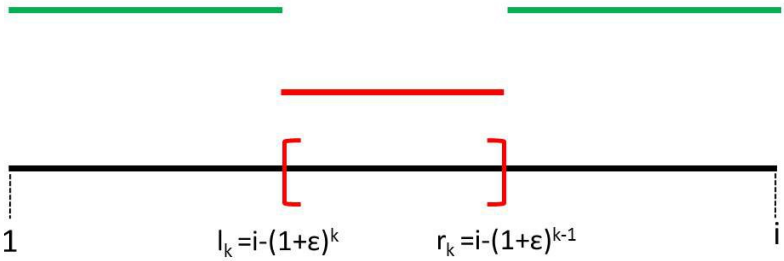


Fig. 3. Solving the k-th Monge problem, $k = 1, \ldots, m = O(\log_{1+\epsilon}(i))$, for a fixed index i. The k-th interval is the set of indices $\{j : l_k \leq j \leq r_k, l_k = i-(1+\epsilon)^k, r_k = i-(1+\epsilon)^{k-1}\}$. Ideally, we wish to define a Monge function w'_k whose maximum inside the k-th interval (red color) is smaller than the minimum outside that interval (green color). This ensures that the optimal breakpoint for the k-th Monge subproblem lies inside the red interval.

Similarly $w_k(i_1, i_4) = 2^{i_4-i_1}\lambda > 2^{i_3-i_1}\lambda = w_k(i_1, i_3)$. Adding them gives the desired result. □

The pseudocode is shown in Algorithm 3. The algorithm computes the OPT values online time based on the above analysis. In order to solve each Monge sub-problem our method calls the routine of Theorem 2.3. For each index i, we compute OPT_i by taking the best value over queries to all k of the Monge query structures, then we update all the structures with this value. Note that storing values of the form $2^k\lambda$ using only their exponent k suffices for comparison, so introducing $w_k(j, i)$ doesn't result in any change in runtime. By Theorem 2.3, for each Q_k, finding $\min_{j<i} Q_k.a_j + w_k(j, i)$ over all i takes $O(n)$ time. Hence, the total runtime is $O(n \log n/\epsilon)$.

ALGORITHM 3: Approximation within a factor of ϵ using Monge function search

```
/* The weight function w_k(j,i) is Monge. Specifically,
   w_k(j,i) = C + ( Σ_{m=j+1}^{i} (i-j)P_m^2 - (S_i-S_j)^2/(1+ε)^k ) for all j which satisfy
   (1+ε)^{k-1} ≤ i-j ≤ (1+ε)^k .                                          */
Maintain m = log n / log(1+ε) Monge function search data structures Q_1, ..., Q_m where Q_k
corresponds to the Monge function w_k(j,i).
OPT_0 ← 0
/* Recursion basis                                                        */
for k = 1 to m do
    Q_k.a_0 ← 0
end
for i = 1 to n do
    OPT_i ← ∞
    for k = 1 to m do
        /* Let a_j^(k) denote Q_k.a_j, i.e., the value a_j of the k-th data structure Q_k
           */
        localmin_k ← min_{j<i} a_j^(k) + w_k(j,i)
        OPT_i ← min{OPT_i, localmin_k + C}
    end
end
for k = 1 to m do
    Q_k.a_i ← OPT_i
end
```

Table II. Datasets, papers and the URLs where the datasets can be downloaded. ⊙ and ■ denote which datasets are synthetic and real respectively.

	Dataset	Availability
⊙	Lai et al.	[Lai et al. 2005] http://compbio.med.harvard.edu/
⊙	Willenbrock et al.	[Willenbrock and Fridlyand 2005] http://www.cbs.dtu.dk/~hanni/aCGH/
■	Coriell Cell lines	[Snijders et al. 2001] http://www.nature.com/ng/journal/v29/n3/
■	Berkeley Breast Cancer	[Neve et al. 2006] http://icbp.lbl.gov/breastcancer/

7. VALIDATION OF OUR MODEL

In this Section we validate our model using the exact algorithm, see Section 3. In Section 7.1 we describe the datasets and the experimental setup. In Section 7.2 we show the findings of our method together with a detailed biological analysis.

7.1. Experimental Setup and Datasets

Our code is implemented in MATLAB[2]. The experiments run in a 4GB RAM, 2.4GHz Intel(R) Core(TM)2 Duo CPU, Windows Vista machine. Our methods were compared to existing MATLAB implementations of the CBS algorithm, available via the Bioinformatics toolbox, and the CGHSEG algorithm [Picard et al. 2005], courteously provided to us by Franc Picard. CGHSEG was run using heteroscedastic model under the Lavielle criterion [Lavielle 2005]. Additional tests using the homoscedastic model showed substantially worse performance and are omitted here. All methods were compared using previously developed benchmark datasets, shown in Table II. Follow-up analysis of detected regions was conducted by manually searching for significant genes in the Genes-to-Systems Breast Cancer Database http://www.itb.cnr.it/breastcancer [Viti et al. 2009] and validating their positions with the UCSC Genome Browser http://genome.ucsc.edu/. The Atlas of Genetics and Cytogenetics in Oncology and Haematology http://atlasgeneticsoncology.org/ was also used to validate the significance of reported cancer-associated genes. It is worth pointing out that since aCGH data are typically given in the log scale, we first exponentiate the points, then fit the constant segment by taking the average of the exponentiated values from the hypothesized segment, and then return to the log domain by taking the logarithm of that constant value. Observe that one can fit a constant segment by averaging the log values using Jensen's inequality, but we favor an approach more consistent with the prior work, which typically models the data assuming i.i.d. Gaussian noise in the linear domain.

How to pick C?. The performance of our algorithm depends on the value of the parameter C, which determines how much each segment "costs." Clearly, there is a tradeoff between larger and smaller values: excessively large C will lead the algorithm to output a single segment while excessively small C will result in each point being fit as its own segment. We pick our parameter C using data published in [Willenbrock and Fridlyand 2005]. The data was generated by modeling real aCGH data, thus capturing their nature better than other simplified synthetic data and also making them a good training dataset for our model. We used this dataset to generate a Receiver Operating Characteristic (ROC) curve using values for C ranging from 0 to 4 with increment 0.01 using one of the four datasets in [Willenbrock and Fridlyand 2005] ("above 20"). The

[2]Code available at URL http://www.math.cmu.edu/~ctsourak/CGHTRIMMER.zip. Faster C code is also available, but since the competitors were implemented in MATLAB, all the results in this Section refer to our MATLAB implementation.

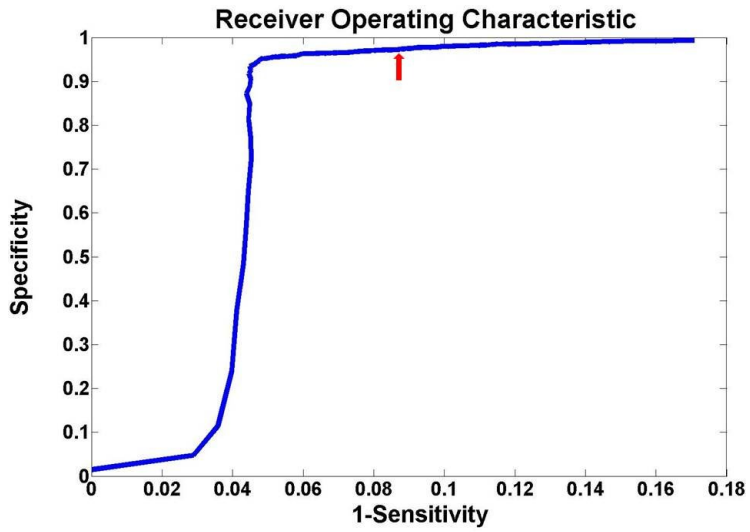


Fig. 4. ROC curve of CGHTRIMMER as a function of C on data from [Willenbrock and Fridlyand 2005]. The red arrow indicates the point (0.91 and 0.98 recall and precision respectively) corresponding to C=0.2, the value used in all subsequent results.

resulting curve is shown in Figure 4. Then, we selected $C = 0.2$, which achieves high precision/specificity (0.98) and high recall/sensitivity (0.91). All subsequent results reported were obtained by setting C equal to 0.2.

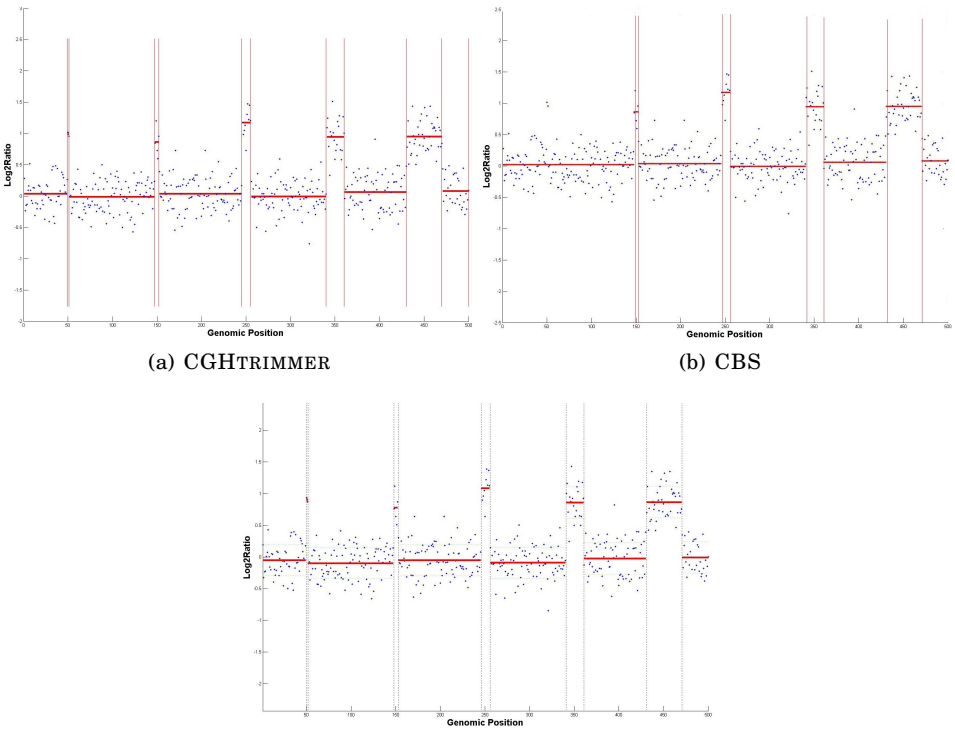
7.2. Experimental Results and Biological Analysis

We show the results on synthetic data in Section 7.2.1, on real data where the ground truth is available to us in Section 7.2.2 and on breast cancer cell lines with no ground truth in Section 7.2.3.

7.2.1. Synthetic Data. We use the synthetic data published in [Lai et al. 2005]. The data consist of five aberrations of increasing widths of 2, 5, 10, 20 and 40 probes, respectively, with Gaussian noise N(0,0.25^2). Figure 5 shows the performance of CGHTRIMMER, CBS, and CGHSEG. Both CGHTRIMMER and CGHSEG correctly detect all aberrations, while CBS misses the first, smallest region. The running time for CGHTRIMMER is 0.007 sec, compared to 1.23 sec for CGHSEG and 60 sec for CBS.

7.2.2. Coriell Cell Lines. The first real dataset we use to evaluate our method is the Coriell cell line BAC array CGH data [Snijders et al. 2001], which is widely considered a "gold standard" dataset. The dataset is derived from 15 fibroblast cell lines using the normalized average of $\log_2$ fluorescence relative to a diploid reference. To call gains or losses of inferred segments, we assign to each segment the mean intensity of its probes and then apply a simple threshold test to determine if the mean is abnormal. We follow [Bejjani et al. 2005] in favoring ± 0.3 out of the wide variety of thresholds that have been used [Ng et al. 2006].

Table III summarizes the performance of CGHTRIMMER, CBS and CGHSEG relative to previously annotated gains and losses in the Corielle dataset. The table shows notably better performance for CGHTRIMMER compared to either alternative method. CGHTRIMMER finds 22 of 23 expected segments with one false positive. CBS finds 20 of 23 expected segments with one false positive. CGHSEG finds 22 of 23 expected segments with seven false positives. CGHTRIMMER thus achieves the same recall as



(a) CGHTRIMMER (b) CBS

(c) CGHSEG

Fig. 5. Performance of CGHTRIMMER, CBS, and CGHSEG on denoising synthetic aCGH data from [Lai et al. 2005]. CGHTRIMMER and CGHSEG exhibit excellent precision and recall whereas CBS misses two consecutive genomic positions with DNA copy number equal to 3.

CGHSEG while outperforming it in precision and the same precision as CBS while outperforming it in recall. In cell line GM03563, CBS fails to detect a region of two points which have undergone a loss along chromosome 9, in accordance with the results obtained using the Lai et al. [Lai et al. 2005] synthetic data. In cell line GM03134, CGHSEG makes a false positive along chromosome 1 which both CGHTRIMMER and CBS avoid. In cell line GM01535, CGHSEG makes a false positive along chromosome 8 and CBS misses the aberration along chromosome 12. CGHTRIMMER, however, performs ideally on this cell line. In cell line GM02948, CGHTRIMMER makes a false positive along chromosome 7, finding a one-point segment in 7q21.3d at genomic position 97000 whose value is equal to 0.732726. All other methods also make false positive errors on this cell line. In GM7081, all three methods fail to find an annotated aberration on chromosome 15. In addition, CGHSEG finds a false positive on chrosome 11.

CGHTRIMMER also substantially outperforms the comparative methods in run time, requiring 5.78 sec for the full data set versus 8.15 min for CGHSEG (an 84.6-fold speedup) and 47.7 min for CBS (a 495-fold speedup).

7.2.3. Breast Cancer Cell Lines. To illustrate further the performance of CGHTRIMMER and compare it to CBS and CGHSEG, we applied it to the Berkeley Breast Cancer cell line database [Neve et al. 2006]. The dataset consists of 53 breast cancer cell lines that capture most of the recurrent genomic and transcriptional characteristics of 145 primary breast cancer cases. We do not have an accepted “answer key” for this data

Table III. Results from applying CGHTRIMMER, CBS, and CGHSEG to 15 cell lines. Rows with listed chromosome numbers (e.g., GM03563/3) corresponded to known gains or losses and are annotated with a check mark if the expected gain or loss was detected or a "No" if it was not. Additional rows list chromosomes on which segments not annotated in the benchmark were detected; we presume these to be false positives.

Cell Line/Chromosome	CGHTRIMMER	CBS	CGHSEG
GM03563/3	✓	✓	✓
GM03563/9	✓	No	✓
GM03563/False	-	-	-
GM00143/18	✓	✓	✓
GM00143/False	-	-	-
GM05296/10	✓	✓	✓
GM05296/11	✓	✓	✓
GM05296/False	-	-	4,8
GM07408/20	✓	✓	✓
GM07408/False	-	-	-
GM01750/9	✓	✓	✓
GM01750/14	✓	✓	✓
GM01750/False	-	-	-
GM03134/8	✓	✓	✓
GM03134/False	-	-	1
GM13330/1	✓	✓	✓
GM13330/4	✓	✓	✓
GM13330/False	-	-	-
GM03576/2	✓	✓	✓
GM03576/21	✓	✓	✓
GM03576/False	-	-	-
GM01535/5	✓	✓	✓
GM01535/12	✓	No	✓
GM01535/False	-	-	8
GM07081/7	✓	✓	✓
GM07081/15	No	No	No
GM07081/False	-	-	11
GM02948/13	✓	✓	✓
GM02948/False	7	1	2
GM04435/16	✓	✓	✓
GM04435/21	✓	✓	✓
GM04435/False	-	-	8,17
GM10315/22	✓	✓	✓
GM10315/False	-	-	-
GM13031/17	✓	✓	✓
GM13031/False	-	-	-
GM01524/6	✓	✓	✓
GM01524/False	-	-	-

set, but it provides a more extensive basis for detailed comparison of differences in performance of the methods on common data sets, as well as an opportunity for novel discovery. While we have applied the methods to all chromosomes in all cell lines, space limitations prevent us from presenting the full results here. The interested reader can reproduce all the results including the ones not presented here[3]. We therefore arbitrarily selected three of the 53 cell lines and selected three chromosomes per cell line that we believed would best illustrate the comparative performance of the methods. The Genes-to-Systems Breast Cancer Database [4] [Viti et al. 2009] was used to identify

[3] http://www.math.cmu.edu/~ctsourak/DPaCGHsupp.html
[4] http://www.itb.cnr.it/breastcancer

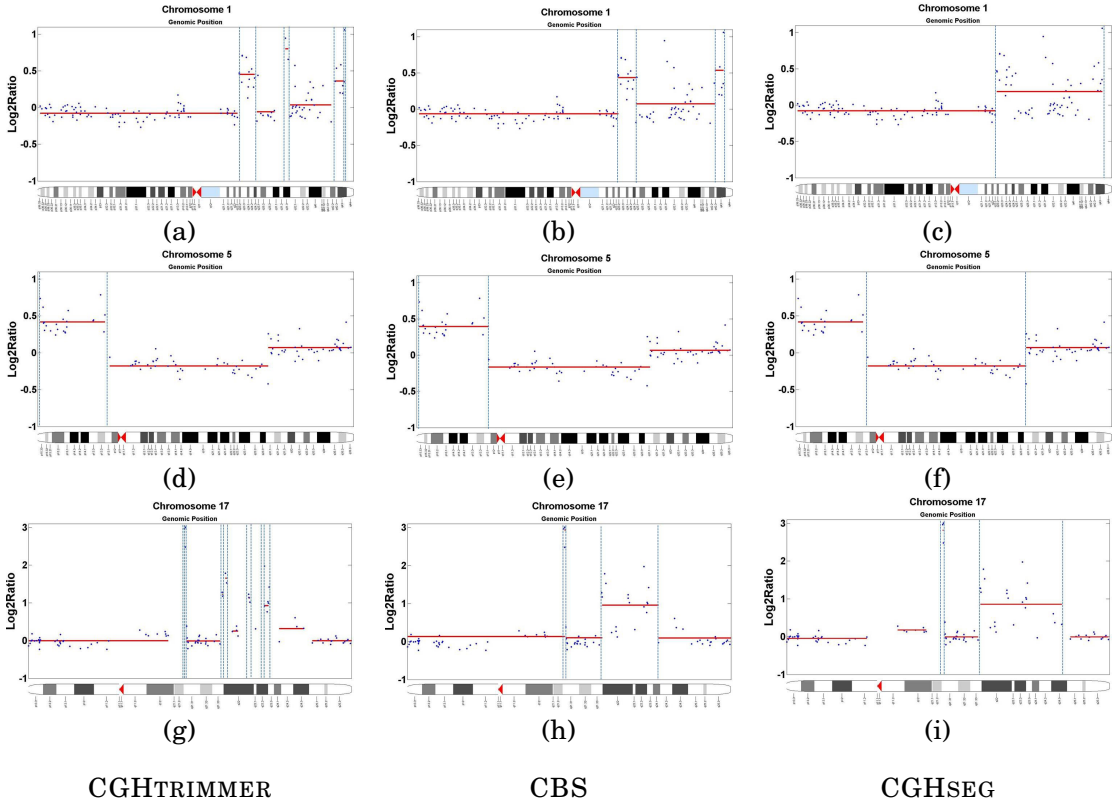


Fig. 6. Visualization of the segmentation output of CGHTRIMMER, CBS, and CGHSEG for the cell line BT474 on chromosomes 1 (a,b,c), 5 (d,e,f), and 17 (g,h,i). (a,d,g) CGHTRIMMER output. (b,e,h) CBS output. (c,f,i) CGHSEG output. Segments exceeding the $\pm$ 0.3 threshold [Bejjani et al. 2005] are highlighted.

known breast cancer markers in regions predicted to be gained or lost by at least one of the methods. We used the UCSC Genome Browser [5] to verify the placement of genes.

We note that CGHTRIMMER again had a substantial advantage in run time. For the full data set, CGHTRIMMER required 22.76 sec, compared to 23.3 min for CGHSEG (a 61.5-fold increase), and 4.95 hrs for CBS (a 783-fold increase).

Cell Line BT474:. Figure 6 shows the performance of each method on the BT474 cell line. The three methods report different results for chromosome 1, as shown in Figures 6(a,b,c), with all three detecting amplification in the q-arm but differing in the detail of resolution. CGHTRIMMER is the only method that detects region 1q31.2-1q31.3 as aberrant. This region hosts gene NEK7, a candidate oncogene [Kimura and Okano 2001] and gene KIF14, a predictor of grade and outcome in breast cancer [Corson et al. 2005]. CGHTRIMMER and CBS annotate the region 1q23.3-1q24.3 as amplified. This region hosts several genes previously implicated in breast cancer [Viti et al. 2009], such as CREG1 (1q24), POU2F1 (1q22-23), RCSD1 (1q22-q24), and BLZF1 (1q24). Finally, CGHTRIMMER alone reports independent amplification of the gene CHRM3, a marker of metastasis in breast cancer patients [Viti et al. 2009].

For chromosome 5 (Figures 6(d,e,f)), the behavior of the three methods is almost identical. All methods report amplification of a region known to contain many

[5] `http://genome.ucsc.edu/`

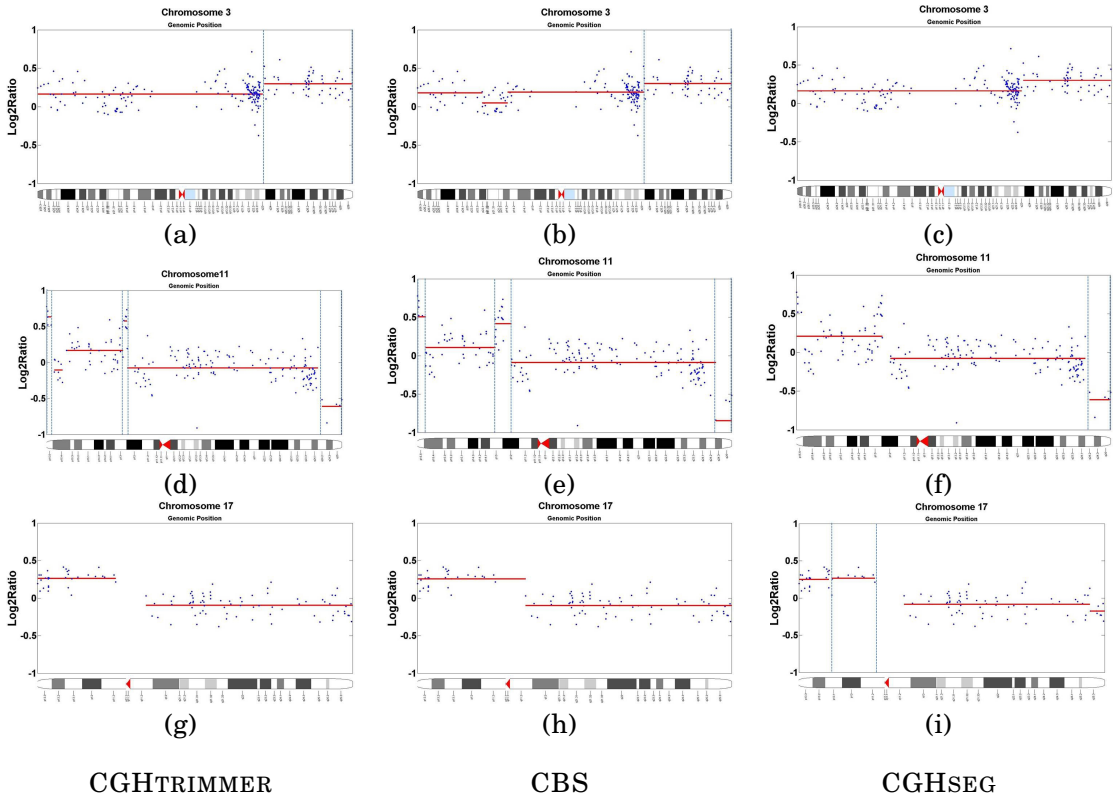


Fig. 7. Visualization of the segmentation output of CGHTRIMMER, CBS, and CGHSEG for the cell line HS578T on chromosomes 3 (a,b,c), 11 (d,e,f), and 17 (g,h,i). (a,d,g) CGHTRIMMER output. (b,e,h) CBS output. (c,f,i) CGHSEG output. Segments exceeding the $\pm$ 0.3 threshold [Bejjani et al. 2005] are highlighted.

breast cancer markers, including MRPL36 (5p33), ADAMTS16 (5p15.32), POLS (5p15.31), ADCY2 (5p15.31), CCT5 (5p15.2), TAS2R1 (5p15.31), ROPN1L (5p15.2), DAP (5p15.2), ANKH (5p15.2), FBXL7 (5p15.1), BASP1 (5p15.1), CDH18 (5p14.3), CDH12 (5p14.3), CDH10 (5p14.2 - 5p14.1), CDH9 (5p14.1) PDZD2 (5p13.3), GOLPH3 (5p13.3), MTMR12 (5p13.3), ADAMTS12 (5p13.3 - 5p13.2), SLC45A2 (5p13.2), TARS (5p13.3), RAD1 (5p13.2), AGXT2 (5p13.2), SKP2 (5p13.2), NIPBL (5p13.2), NUP155 (5p13.2), KRT18P31 (5p13.2), LIFR (5p13.1) and GDNF (5p13.2) [Viti et al. 2009]. The only difference in the assignments is that CBS fits one more probe to this amplified segment.

Finally, for chromosome 17 (Figures 6(g,h,i)), like chromosome 1, all methods detect amplification but CGHTRIMMER predicts a finer breakdown of the amplified region into independently amplified segments. All three methods detect amplification of a region which includes the major breast cancer biomarkers HER2 (17q21.1) and BRCA1 (17q21) as also the additional markers MSI2 (17q23.2) and TRIM37 (17q23.2) [Viti et al. 2009]. While the more discontiguous picture produced by CGHTRIMMER may appear to be a less parsimonious explanation of the data, a complex combination of fine-scale gains and losses in 17q is in fact well supported by the literature [Orsetti et al. 2004].

Cell Line HS578T:. Figure 7 compares the methods on cell line HS578T for chromosomes 3, 11 and 17. Chromosome 3 (Figures 7(a,b,c)) shows identical prediction

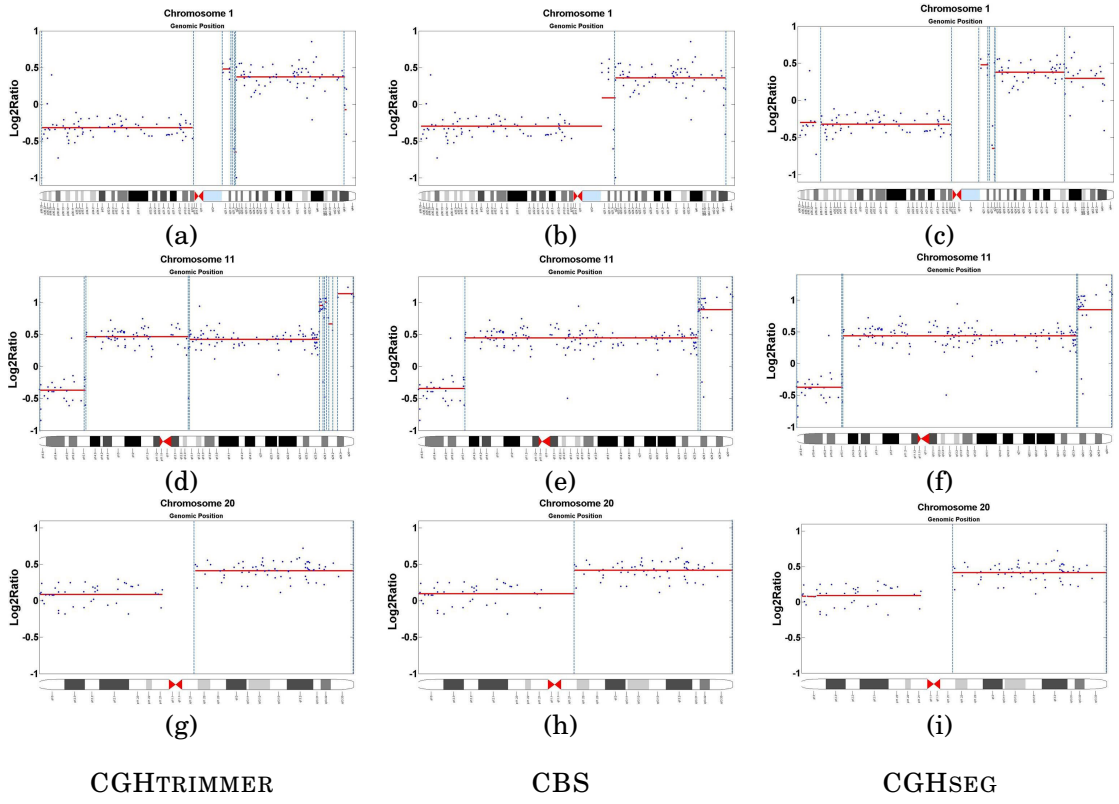


Fig. 8. Visualization of the segmentation output of CGHTRIMMER, CBS, and CGHSEG for the cell line T47D on chromosomes 1 (a,b,c), 11 (d,e,f), and 20 (g,h,i). (a,d,g) CGHTRIMMER output. (b,e,h) CBS output. (c,f,i) CGHSEG output. Segments exceeding the $\pm$ 0.3 threshold [Bejjani et al. 2005] are highlighted.

of an amplification of 3q24-3qter for all three methods. This region includes the key breast cancer markers PIK3CA (3q26.32) [Lee et al. 2005], and additional breast-cancer-associated genes TIG1 (3q25.32), MME (3q25.2), TNFSF10 (3q26), MUC4 (3q29), TFRC (3q29), DLG1 (3q29) [Viti et al. 2009]. CGHTRIMMER and CGHSEG also make identical predictions of normal copy number in the p-arm, while CBS reports an additional loss between 3p21 and 3p14.3. We are unaware of any known gain or loss in this region associated with breast cancer.

For chromosome 11 (Figures 7(d,e,f)), the methods again present an identical picture of loss at the q-terminus (11q24.2-11qter) but detect amplifications of the p-arm at different levels of resolution. CGHTRIMMER and CBS detect gain in the region 11p15.5, which is the site of the HRAS breast cancer metastasis marker [Viti et al. 2009]. In contrast to CBS, CGHTRIMMER detects an adjacent loss region. While we have no direct evidence this loss is a true finding, the region of predicted loss does contain EIF3F (11p15.4), identified as a possible tumor suppressor whose expression is decreased in most pancreatic cancers and melanomas [Viti et al. 2009]. Thus, we conjecture that EIF3F is a tumor suppressor in breast cancer.

On chromosome 17 (Figures 7(g,h,i)), the three methods behave similarly, with all three predicting amplification of the p-arm. CBS places one more marker in the amplified region causing it to cross the centromere while CGHSEG breaks the amplified region into three segments by predicting additional amplification at a single marker.

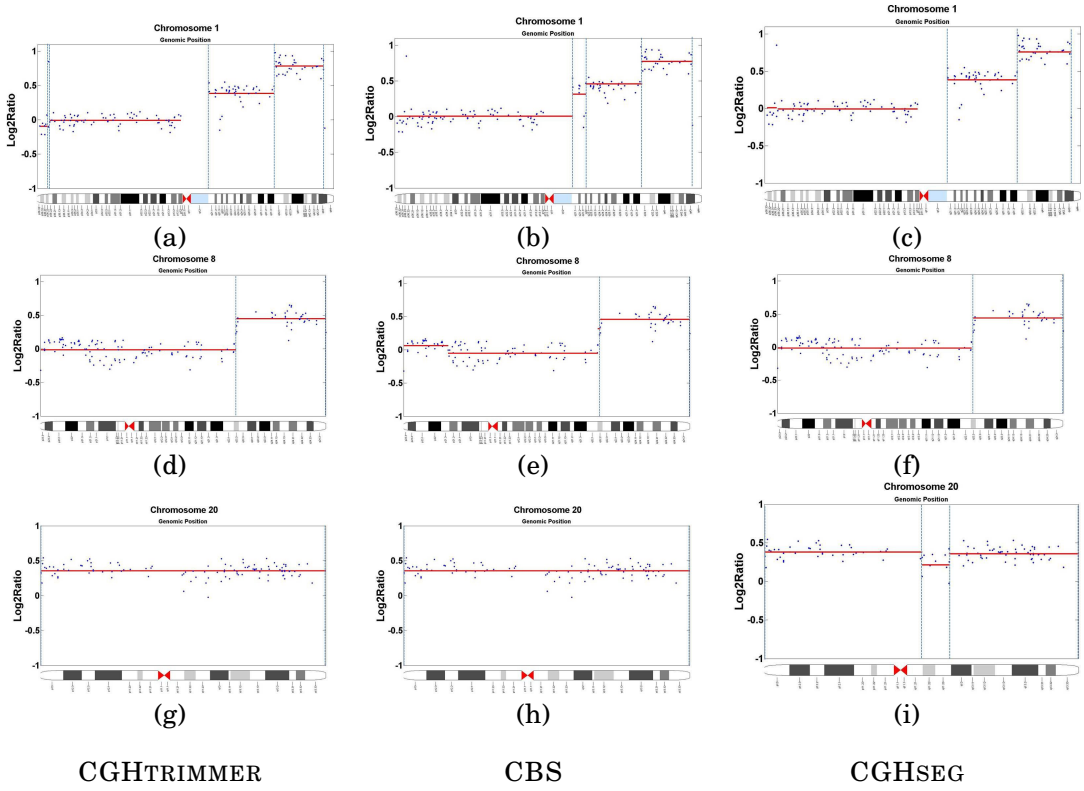


Fig. 9. Visualization of the segmentation output of CGHTRIMMER, CBS, and CGHSEGfor the cell line MCF10A on chromosomes 1 (a,b,c), 8 (d,e,f), and 20 (g,h,i). (a,d,g) CGHTRIMMER output. (b,e,h) CBS output. (c,f,i) CGHSEG output. Segments exceeding the $\pm$ 0.3 threshold [Bejjani et al. 2005] are highlighted.

Cell Line T47D:. Figure 8 compares the methods on chromosomes 1, 8, and 20 of the cell line T47D. On chromosome 1 (Figure 8(a,b,c)), all three methods detect loss of the p-arm and a predominant amplification of the q-arm. CBS infers a presumably spurious extension of the p-arm loss across the centromere into the q-arm, while the other methods do not. The main differences between the three methods appear on the q-arm of chromosome 1. CGHTRIMMER and CGHSEG detect a small region of gain proximal to the centromere at 1q21.1-1q21.2, followed by a short region of loss spanning 1q21.3-1q22. CBS merges these into a single longer region of normal copy number. The existence of a small region of loss at this location in breast cancers is supported by prior literature [Chunder et al. 2003].

The three methods provide comparable segmentations of chromosome 11 (Figure 8(d,e,f)). All predict loss near the p-terminus, a long segment of amplification stretching across much of the p- and q-arms, and additional amplification near the q-terminus. CGHTRIMMER, however, breaks this q-terminal amplification into several sub-segments at different levels of amplification while CBS and CGHSEG both fit a single segment to that region. We have no empirical basis to determine which segmentation is correct here. CGHTRIMMER does appear to provide a spurious break in the long amplified segment that is not predicted by the others.

Finally, along chromosome 20 (Figure 8(g,h,i)), the output of the methods is similar, with all three methods suggesting that the q-arm has an aberrant copy number, an observation consistent with prior studies [Hodgson et al. 2003]. The only exception is

again that CBS fits one point more than the other two methods along the first segment, causing a likely spurious extension of the p-arm's normal copy number into the q-arm.

Cell Line MCF10A. Figure 9 shows the output of each of the three methods on chromosomes 1, 8, and 20 of the cell line MCF10A. On this cell line, the methods all yield similar predictions although from slightly different segmentations. All three show nearly identical behavior on chromosome 1 (Figure 9(a,b,c)), with normal copy number on the p-arm and at least two regions of independent amplification of the q-arm. Specifically, the regions noted as gain regions host significant genes such as PDE4DIP a gene associated with breast metastatic to bone (1q22), ECM1 (1q21.2), ARNT (1q21), MLLT11 (1q21), S100A10 (1q21.3), S100A13 (1q21.3), TPM3 (1q25) which also plays a role in breast cancer metastasis, SHC1 (1q21.3) and CKS1B (1q21.3). CBS provides a slightly different segmentation of the q-arm near the centromere, suggesting that the non-amplified region spans the centromere and that a region of lower amplification exists near the centromere. On chromosome 8 (Figure 9(d,e,f)) the three algorithms lead to identical copy number predictions after thresholding, although CBS inserts an additional breakpoint at 8q21.3 and a short additional segment at 8q22.2 that do not correspond to copy number changes. All three show significant amplification across chromosome 20 (Figure 9(g,h,i)), although in this case CGHSEG distinguishes an additional segment from 20q11.22-20q11.23 that is near the amplification threshold. It is worth mentioning that chromosome 20 hosts significant breast cancer related genes such as CYP24 and ZNF217.

8. CONCLUSIONS

In this paper, we present a new formulation for the problem of denoising aCGH data. Our formulation has already proved to be valuable in numerous settings [Ding and Shah 2010]. We show a basic exact dynamic programming algorithm which runs in $O(n^2)$ time and performs excellently on both synthetic and real data. More interestingly from a theoretical perspective, we develop two techniques for performing approximate dynamic programming in both the additive and the multiplicative norm. Our first algorithm reduces the optimization problem into a well studied geometric problem, namely halfspace emptiness queries. The second technique carefully breaks the problem into a small number of Monge subproblems, which are solved efficiently using existing techniques. Our results strongly indicate that the $O(n^2)$ algorithm, which is — to the best of our knowledge — the fastest exact algorithm, is not tight. There is inherent structure in the optimization problem. Lemma 8.1 is such an example.

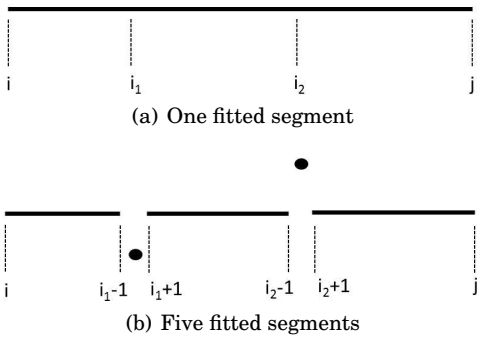


Fig. 10. Lemma 8.1: if $|P_{i_1} - P_{i_2}| > 2\sqrt{2C}$ then Segmentation (b) which consists of five segments (two of which are single points) is superior to Segmentation (a) which consists of a single segment.

LEMMA 8.1. *If* $|P_{i_1} - P_{i_2}| > 2\sqrt{2C}$*, then in the optimal solution of the dynamic programming using* L_2 *norm,* i_1 *and* i_2 *are in different segments.*

PROOF. The proof is by contradiction, see also Figure 10. Suppose the optimal solution has a segment $[i, j]$ where $i \leq i_1 < i_2 \leq j$, and its optimal x value is x^*. Then consider splitting it into 5 intervals $[i, i_1 - 1]$, $[i_1, i_1]$, $[i_1 + 1, i_2 - 1]$, $[i_2, i_2]$, $[i_2 + 1, r]$. We let $x = x^*$ be the fitted value to the three intervals not containing i_1 and i_2. Also, as $|P_{i_1} - P_{i_2}| > 2\sqrt{2C}$, $(P_{i_1} - x)^2 + (P_{i_2} - x)^2 > 2\sqrt{2C}^2 = 4C$. So by letting $x = P_{i_1}$ in $[i_1, i_1]$ and $x = P_{i_2}$ in $[i_2, i_2]$, the total decreases by more than $4C$. This is more than the added penalty of having 4 more segments, a contradiction with the optimality of the segmentation. □

Uncovering and taking advantage of the inherent structure in a principled way should result in a faster exact algorithm. This is an interesting research direction which we leave as future work. Another research direction is to find more applications (e.g., histogram construction [Guha et al. 2006]) to which our methods are applicable.

ACKNOWLEDGMENTS

The authors would like to thank Frank Picard for making his MATLAB code available to us. We would like to thank Pawel Gawrychowski for pointing us reference [Agarwal et al. 1992] and Dimitris Achlioptas and Yuan Zhou for helpful discussions on optimization techniques. We would also like to thank the reviewers for their valuable comments.

REFERENCES

AGARWAL, P. K., EPPSTEIN, D., AND MATOUSEK, J. 1992. Dynamic half-space reporting, geometric optimization, and minimum spanning trees. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 80–89.

AGARWAL, P. K. AND ERICKSON, J. 1999. *Geometric Range Searching and Its Relatives*.

AGGARWAL, A., HANSEN, M., AND LEIGHTON, T. 1990. Solving query-retrieval problems by compacting voronoi diagrams. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. STOC '90. ACM, New York, NY, USA, 331–340.

AGGARWAL, A., KLAWE, M., MORAN, S., SHOR, P., AND WILBER, R. 1986. Geometric applications of a matrix searching algorithm. In *Proceedings of the second annual symposium on Computational geometry*. SCG '86. ACM, New York, NY, USA, 285–292.

ALBERTSON, D. AND PINKEL, D. 2003. Genomic microarrays in human genetic disease and cancer. *Human Molecular Genetics 12*, 145–152.

BARRY, D. AND HARTIGAN, J. 1993. A bayesian analysis for change point problems. *Journal of the American Statistical Association 88,* 421, 309–319.

BEIN, W., GOLIN, M., LARMORE, L., AND ZHANG, Y. 2009. The knuth-yao quadrangle-inequality speedup is a consequence of total monotonicity. *ACM Trans. Algorithms 6*, 1–22.

BEJJANI, B., SALEKI, R., BALLIF, B., OREM, E., SUNDIN, K., THEISEN, A., KASHORK, C., AND SHAFFER, L. 2005. Use of targeted array-based cgh for the clinical diagnosis of chromosomal imbalance: is less more? *American Journal of Medical Genetics A*, 259–67.

BELLMAN, R. 2003. *Dynamic Programming*. Dover Publications.

BERTSEKAS, D. 2000. *Dynamic Programming and Optimal Control*. Athena Scientific.

BIGNELL, G., HUANG, J., GRESHOCK, J., WATT, S., BUTLER, A., WEST, S., GRIGOROVA, M., JONES, K., WEI, W., STRATTON, M., FUTREAL, A., WEBER, B., SHAPERO, M., AND WOOSTER, R. 2004. High-resolution analysis of dna copy number using oligonucleotide microarrays. *Genome Research*, 287–295.

BURKARD, R. E., KLINZ, B., AND RÜDIGER, R. 1996. Perspectives of monge properties in optimization. *Discrete Appl. Math. 70*, 95–161.

BUTLER, M., MEANEY, F., AND PALMER, C. 1986. Clinical and cytogenetic survey of 39 individuals with prader-labhart-willi syndrome. *American Journal of Medical Genetics 23*, 793–809.

CHAZELLE, B., GUIBAS, L., AND LEE, D. T. 1985. The power of geometric duality. *BIT 25*, 76–90.

CHAZELLE, B. AND PREPARATA, F. 1985. Halfspace range search: an algorithmic application of k-sets. In *Proceedings of the first annual symposium on Computational geometry*. SCG '85. ACM, New York, NY, USA, 107–115.

CHUNDER, N., MANDAL, S., BASU, D., ROY, A., ROYCHOUDHURY, S., AND PANDA, C. K. 2003. Deletion mapping of chromosome 1 in early onset and late onset breast tumors–a comparative study in eastern india. *Pathol Res Pract 199*, 313–321.

CLARKSON, K. L. AND SHOR., P. W. 1989. Applications of random sampling in computational geometry, ii. *Discrete Comput. Geom. 4*, 387–421.

CORSON, T., HUANG, A., TSAO, M., AND GALLIE, B. 2005. Kif14 is a candidate oncogene in the 1q minimal region of genomic gain in multiple cancers. *Oncogene 24*, 47414753.

DESPER, R., JIANG, F., KALLIONIEMI, O.-P., MOCH, H., PAPADIMITRIOU, C. H., AND SCHÄFFER, A. A. 1999. Inferring tree models for oncogenesis from comparative genome hybridization data. *Journal of Computational Biology 6,* 1, 37–52.

DESPER, R., JIANG, F., KALLIONIEMI, O.-P., MOCH, H., PAPADIMITRIOU, C. H., AND SCHÄFFER, A. A. 2000. Distance-based reconstruction of tree models for oncogenesis. *Journal of Computational Biology 7,* 6, 789–803.

DING, J. AND SHAH, S. P. 2010. Robust hidden semi-markov modeling of array cgh data. In *IEEE International Conference on Bioinformatics & Biomedicine*. 603–608.

EILERS, P. H. AND DE MENEZES, R. X. 2005. Quantile smoothing of array cgh data. *Bioinformatics 21,* 7, 1146–1153.

EPPSTEIN, D., GALIL, Z., AND GIANCARLO, R. 1988. Speeding up dynamic programming. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 488–496.

EPPSTEIN, D., GALIL, Z., GIANCARLO, R., AND ITALIANO, G. 1992a. Sparse dynamic programming i: linear cost functions. *J. ACM 39*, 519–545.

EPPSTEIN, D., GALIL, Z., GIANCARLO, R., AND ITALIANO, G. 1992b. Sparse dynamic programming ii: convex and concave cost functions. *J. ACM 39*, 546–567.

FRIDLYAND, J., SNIJDERS, A. M., PINKEL, D., ALBERTSON, D. G., AND JAIN, A. N. 2004. Hidden markov models approach to the analysis of array cgh data. *J. Multivar. Anal. 90,* 1, 132–153.

GILBERT, E. AND MOORE, E. 1959. Variable-length binary encodings. *Bell System Tech. 38*, 933–966.

GUHA, S., KOUDAS, N., AND SHIM, K. 2006. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst. 31*, 396–438.

GUHA, S., LI, Y., AND NEUBERG, D. 2006. Bayesian hidden markov modeling of array cgh data. *Harvard University, Paper 24*.

HIRSCHBERG, D. 1975. A linear space algorithm for computing maximal common subsequences. *Commun. ACM 18*, 341–343.

HODGSON, J., CHIN, K., COLLINS, C., AND GRAY, J. 2003. Genome amplification of chromosome 20 in breast cancer. *Breast Cancer Res Treat.*.

HSU, L., SELF, S., GROVE, D., WANG, K., DELROW, J., LOO, L., AND PORTER, P. 2005. Denoising array based comparative genomic hybridization data using wavelets. *Biostatistics 6*, 211–226.

HUANG, T., WU, B., LIZARDI, P., AND ZHAO, H. 2005. Detection of DNA copy number alterations using penalized least squares regression. *Bioinformatics 21,* 20, 3811–3817.

HUPÉ, P., STRANSKY, N., THIERY, J.-P., RADVANYI, F., AND BARILLOT, E. 2004. Analysis of array cgh data: from signal ratio to gain and loss of dna regions. *Bioinformatics 20,* 18, 3413–3422.

JAGADISH, H. V., KOUDAS, N., MUTHUKRISHNAN, S., POOSALA, V., SEVCIK, K. C., AND TORSTEN, S. 1998. Optimal histograms with quality guarantees. In *Proceedings of the 24rd International Conference on Very Large Data Bases*. VLDB '98. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 275–286.

JONG, K., MARCHIORI, E., MEIJER, G., VAN DER VAART, A., AND YLSTRA, B. 2004. Breakpoint identification and smoothing of array comparative genomic hybridization data. *Bioinformatics 20,* 18, 3636–3637.

KALLIONIEMI, A., KALLIONIEMI, O., SUDAR, D., RUTOVITZ, D., GRAY, J., WALDMAN, F., AND PINKEL, D. 1992. Comparative genomic hybridization for molecular cytogenetic analysis of solid tumors. *Science 258,* 5083, 818–821.

KIMURA, M. AND OKANO, Y. 2001. Identification and assignment of the human nima-related protein kinase 7 gene (nek7) to human chromosome 1q31.3. *Cytogenet. Cell Genet.*.

KLAWE, M. AND KLEITMAN, D. 1990. An almost linear time algorithm for generalized matrix searching. *SIAM J. Discret. Math. 3*, 81–97.

KNUTH, D. E. 1971. Optimum binary search trees. *Acta Inf. 1*, 14–25.

KNUTH, D. E. 1981. *Seminumerical Algorithms*. Addison-Wesley.

LAI, W. R., JOHNSON, M. D., KUCHERLAPATI, R., AND PARK, P. J. 2005. Comparative analysis of algorithms for identifying amplifications and deletions in array cgh data. *Bioinformatics 21,* 19, 3763–3770.

LARMORE, L. L. AND SCHIEBER, B. 1991. On-line dynamic programming with applications to the prediction of rna secondary structure. *J. Algorithms 12,* 3, 490–515.

LAVIELLE, M. 2005. Using penalized contrasts for the change-point problem. *Signal Processing 85,* 8, 1501–1510.

LEE, J. W., SOUNG, Y. H., KIM, S. Y., LEE, H. W., PARK, W. S., NAM, S. W., KIM, S. H., LEE, J. Y., YOO, N. J., , AND LEE, S. H. 2005. Pik3ca gene is frequently mutated in breast carcinomas and hepatocellular carcinomas. *Oncogene*.

LEJEUNE, J., GAUTIER, M., AND TURPIN, R. 1959. Etude des chromosomes somatiques de neuf enfants mongoliens. *Comptes Rendus Hebdomadaires des Seances de l'Academie des Sciences (Paris) 248(11)*, 17211722.

LEJEUNE, J., LAFOURCADE, J., BERGER, R., VIALATTA, J., BOESWILLWALD, M., SERINGE, P., AND TURPIN, R. 1963. Trois ca de deletion partielle du bras court d'un chromosome 5. *Compte Rendus de l'Academie des Sciences (Paris) 257*, 3098.

LIPSON, D., AUMANN, Y., BEN-DOR, A., LINIAL, N., AND YAKHINI, Z. 2005. Efficient calculation of interval scores for dna copy number data analysis. In *RECOMB*. 83–100.

MATOUSEK, J. 1991. Reporting points in halfspaces. In *FOCS*. 207–215.

MONGE, G. 1781. Memoire sue la theorie des deblais et de remblais. *Histoire de lAcademie Royale des Sciences de Paris*, 666–704.

MYERS, C. L., DUNHAM, M. J., KUNG, S. Y., AND TROYANSKAYA, O. G. 2004. Accurate detection of aneuploidies in array cgh and gene expression microarray data. *Bioinformatics 20,* 18, 3533–3543.

NEVE, R., CHIN, K., FRIDLYAND, J., YEH, J., BAEHNER, F., FEVR, T., CLARK, N., BAYANI, N., COPPE, J., AND TONG, F. 2006. A collection of breast cancer cell lines for the study of functionally distinct cancer subtypes. *Cancer Cell 10*, 515–527.

NG, G., HUANG, J., ROBERTS, I., AND COLEMAN, N. 2006. Defining ploidy-specific thresholds in array comparative genomic hybridization to improve the sensitivity of detection of single copy alterations in cell lines. *Journal of Molecular Diagnostics*.

OLSHEN, A. B., VENKATRAMAN, E., LUCITO, R., AND M.WIGLER. 2004. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics 5,* 4, 557–572.

ORSETTI, B., NUGOLI, M., CERVERA, N., LASORSA, L., CHUCHANA, P., URSULE, L., NGUYEN, C., REDON, R., DU MANOIR, S., RODRIGUEZ, C., AND THEILLET, C. 2004. Genomic and expression profiling of chromosome 17 in breast cancer reveals complex patterns of alterations and novel candidate genes. *Cancer Research*.

PICARD, F., ROBIN, S., LAVIELLE, M., VAISSE, C., AND DAUDIN, J. J. 2005. A statistical approach for array cgh data analysis. *BMC Bioinformatics 6*.

PICARD, F., ROBIN, S., LEBARBIER, E., AND DAUDIN, J. 2007. A segmentation/clustering model for the analysis of array cgh data. *Biometrics 63*.

PINKEL, D., SEGRAVES, R., SUDAR, D., CLARK, S., POOLE, I., KOWBEL, D., COLLINS, C., KUO, W.-L., CHEN, C., AND ZHA, Y. 1998. High resolution analysis of DNA copy number variation using comparative genomic hybridization to microarrays. *Nature Genetics 25*, 207 – 211.

POLLACK, J. R., PEROU, C. M., ALIZADEH, A. A., EISEN, M. B., PERGAMENSCHIKOV, A., WILLIAMS, C. F., JEFFREY, S. S., BOTSTEIN, D., AND BROWN, P. O. 1999. Genome-wide analysis of dna copy-number changes using cdna microarrays. *Nature Genetics 23*, 41–46.

SEN, A. AND SRIVASTAVA, M. 1975. On tests for detecting change in mean. *Annals of Statistics 3*, 98–108.

SHAH, S. P., XUAN, X., DELEEUW, R. J., KHOJASTEH, M., LAM, W. L., NG, R., AND MURPHY, K. P. 2006. Integrating copy number polymorphisms into array cgh analysis using a robust hmm. *Bioinformatics 22,* 14.

SHI, Y., GUO, F., WU, W., AND XING, E. P. 2007. Gimscan: A new statistical method for analyzing whole-genome array cgh data. In *RECOMB*. 151–165.

SNIJDERS, A., NOWAK, N., SEGRAVES, R., BLACKWOOD, S., BROWN, N., CONROY, J., HAMILTON, G., HINDLE, A., HUEY, B., KIMURA, K., LAW, S., MYAMBO, K., PALMER, J., YLSTRA, B., YUE, J., GRAY, J., JAIN, A., PINKEL, D., AND ALBERTSON, D. 2001. Assembly of microarrays for genome-wide measurement of DNA copy number. *Nature Genetics 29*, 263–264.

TIBSHIRANI, R. AND WANG, P. 2008. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics (Oxford, England) 9,* 1, 18–29.

VITI, F., MOSCA, E., MERELLI, I., CALABRIA, A., ALFIERI, R., AND MILANESI, L. 2009. Ontological enrichment of the Genes-to-Systems Breast Cancer Database. *Communications in Computer and Information Science 46*, 178–182.

WANG, P., KIM, Y., POLLACK, J., NARASIMHAN, B., AND TIBSHIRANI, R. 2005. A method for calling gains and losses in array cgh data. *Biostatistics 6,* 1, 45–58.

WATERMAN, M. S. AND SMITH, T. 1986. Rapid dynamic programming algorithms for rna secondary structure. *Adv. Appl. Math. 7*, 455–464.

WILLENBROCK, H. AND FRIDLYAND, J. 2005. A comparison study: applying segmentation to array cgh data for downstream analyses. *Bioinformatics 21,* 22, 4084–4091.

XING, B., GREENWOOD, C., AND BULL, S. 2007. A hierarchical clustering method for estimating copy number variation. *Biostatistics 8*, 632–653.

YAO, F. F. 1980. Efficient dynamic programming using quadrangle inequalities. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*. STOC '80. ACM, New York, NY, USA, 429–435.

YAO, F. F. 1982. Speed-up in dynamic programming. *SIAM Journal on Algebraic and Discrete Methods 3,* 4, 532–540.

ZHANG, F., GU, W., HURLES, M. E., AND LUPSKI, J. R. 2009. Copy number variation in human health, disease, and evolution. *Annual Review of Genomics and Human Genetics 10*, 451–481.

Received February 2011; revised March 2011; accepted June 2011