A Linear-Time Algorithm for Analyzing Array CGH Data Using Log Ratio Triangulation

Matthew Hayes and Jing Li*

Case Western Reserve University, Cleveland, OH 44106 USA
jingli@case.edu

Abstract. DNA copy number is the number of replicates of a contiguous segment of DNA on the genome. Copy number alteration (CNA) is a genetic abnormality in which the number of these segments differs from the normal copy number, which is two for human chromosomal DNA. The association of CNA with cancer has led to a proliferation of research into algorithmic methods for detecting these regions of genetic abnormality. We propose a linear-time algorithm to identify chromosomal change points using array comparative genomic hybridization (aCGH) data. This method treats log-2 ratio values as points in a triangle and segments the genome into regions of equal copy number by exploiting the properties of log-2 ratio values often seen at segment boundaries. Applying our method to real and simulated aCGH datasets shows that the triangulation method is fast and is robust for data with low to moderate noise levels.

1 Introduction

For a given contiguous segment of chromosomal DNA, the copy number of the segment is the number of its replicates on the chromosome. In humans, the normal copy number is two, with one copy inherited from each parent. However, the phenomenon known as copy number alteration (CNA) can occur which leads to abnormal copy numbers (i.e. where the number of segment replicates is not 2). Copy number alteration can occur because of genetic inheritance, but can also be the result of cancer progression. The studies performed in [1,2,3] provide clear pictures into the relationship between copy number aberrations and different forms of cancer.

The correlation between CNA and cancer necessitates the ability to locate CNA regions on the genome. Accurately locating these regions would aid the researcher in locating relevant genes associated with a particular type of cancer, and it would also help in classifying tumor cells [4,5].

1.1 Locating CNA Regions: The aCGH Platform

Array comparative genomic hybridization (aCGH) is a microarray technology designed to provide genome-wide screening of DNA copy number profiles. Using

* Correspondence author.

I. Măndoiu, G. Narasimhan, and Y. Zhang (Eds.): ISBRA 2009, LNBI 5542, pp. 248–259, 2009.

aCGH, it is possible to measure DNA copy number for thousands of DNA clones along the chromosome [6]. Using data from the aCGH platform, regions of equal copy number can be computationally determined through segmentation, in which the genome is partitioned into regions of distinct copy number. The problem of segmentation is then to find copy number transition points or "breakpoints" - locations that define the boundaries between regions of different copy number.

For a given clone in an aCGH experiment, its copy number in a test sample is compared to its copy number in a reference sample. The samples here are cells, and the normalized log-2 ratio of the copy numbers for the test and reference samples are recorded for each clone. An example graph of log-2 ratio values is shown in Fig. 1. However, a significant problem with aCGH experiments is made clear in this figure: the data is very noisy. There are various experimental factors that contribute to noisy log-2 ratio measurements, so it can be expected that this aspect of aCGH data will continue to be problematic.

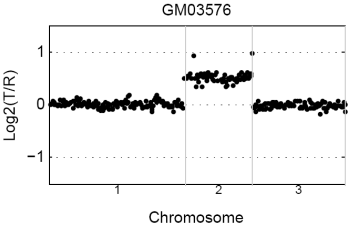

Fig. 1. Subset of aCGH data from Coriell cell line [7]. Each point represents the log-2 ratio value of one clone.

1.2 Our Proposed Method: Log-2 Ratio Triangulation

We propose a linear-time algorithm to locate chromosomal breakpoints in array CGH data by triangulating[1] log-2 ratio values, which enables us to exploit the properties of triangles created from those values. Our results show that this method is not only fast, but is robust against aCGH datasets with low to moderate noise levels.

2 Related Work

Locating regions of distinct copy number can be generalized to the problem of detecting *change points*, which are specific locations in a numeric set where the local distribution changes. The Circular Binary Segmentation (CBS) algorithm of Olshen *et al.* finds change points in array CGH data by partitioning a chromosome until a partition is found that maximizes a t-statistic [8]. In two comparison

[1] Note that we define triangulation as the task of forming a triangle from 3 points in two dimensional space.

papers that evaluate aCGH analysis algrithms [9,10], the CBS algorithm consistently returned good results, but was computationally slow. A change to the CBS algorithm [11] sought to address this issue and while the speed improvements were sound, the algorithm still produced a significant amount of overhead for certain instances of their datasets.

The permutation test performed by the CBS algorithm results in a running time of $O(n^2)$, where n is the number of clones. Other methods, such as the hidden Markov model (HMM) methods of Guha [12] and Fridlyand [13] could require quadratic time because the algorithms associated with HMMs (i.e. Viterbi, Baum-Welch, forward-backward) all require at *least* $\Theta(n)$ time in the number of observations [14], though the running times of these methods may depend on the number and connectivity of the hidden states in the model. The HMM-based methods are different than segmentation algorithms such as CBS because the HMM methods generally assign clones to states, where the states are "gain" or "loss" of a specific number of copies. Dynamic programming based approaches such as that proposed by Daruwala *et al.* [15] could potentially have high asymptotic running times because of the time required to fill in the tabular information used by the algorithm.

3 Log-2 Ratio Triangulation

The idea of the triangulation method is to exploit the behavior of log-2 ratio values that are often seen at change point locations. By forming triangles from every three consecutive ratio values on the chromosome, we seek to computationally compare the triangles formed in segments of constant copy number to triangles formed at the segment boundaries. To distinguish between these two types of triangles, we define two score functions S and G that assign values to each triangle. The S-score for a triangle τ is defined by the following equation:

$$S(\tau) := F^{cHT^\circ} \ . \tag{1}$$

where F is the value in radians of the largest interior angle of τ, T is the triangle's tilt off the x-axis in degrees, H is the height of τ, and c is a user specified parameter whose effects are explored in Section 4.[2] Figure 2 provides an illustrative example of the T, F, and H dimensions and how they are determined from log-2 ratio values.

After calculating the value in (1), $S(\tau)$ is then given as input to the following function $G(S)$, which we refer to as the G-score.

$$G(S(\tau)) := arctan(\gamma(S(\tau) - d)) \ . \tag{2}$$

In (2), the variable γ is the estimated noise dispersion from the segment means of all log-2 ratios for the given input set of clones. The variable d is a user-specified constant that specifies the x-intercept of the arctangent function.

[2] We specify the tilt dimension in degrees so that it produces a larger value to aid in separating triangles at breakpoints from triangles in constant copy number segments.

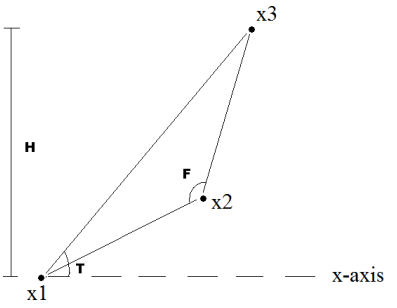

Fig. 2. Illustrative example of T,F, and H for a given triangle. The points x1, x2, and x3 are log-2 ratio values in the aCGH data, and triangles are created for every three consecutive log-2 ratio values.

The effects of this parameter are studied further in Section 4. We calculate γ by first performing median filtering on the input log-2 ratios (with a window size of 20) and for each original ratio value, we take the difference between that value and its corresponding filtered value. We then sum the differences (i.e., their absolute values) of all clones to their filtered values and we divide the resulting sum by the number of clones in the input set. The result is a value that serves as a measure of noise level for the input data and such a value is important because array CGH datasets vary in the levels of noise they contain.[3] The γ parameter also affects the curvature of the sigmoid and the rate at which the upper and lower asymptotes are reached. A larger γ value will give a steeper curve and will have a larger rate of change to reach the lower and upper asymptotes. This effect is desirable because γ varies directly with noise, and noise varies directly with S-score values. Thus, a noisy dataset will have triangles with larger S-scores, and we want our sigmoid G function to account for these larger values.

The idea of the score functions is not only to emphasize the importance of the T,F, and H dimensions, but to exploit the properties of triangles that are formed at the breakpoint locations. These triangles tend to have larger heights, greater tilt, and larger maximum interior angles than other triangles. Since these triangles are formed from log-2 ratio values in the data, we are essentially exploiting the noisiness of the data and projecting the related properties onto properties of the associated triangles. Equations (1) and (2) also allow our method to numerically separate triangles seen at copy number transition points from triangles obtained from clones with the same copy number. For our purposes, we want to place smaller triangles with low S-score values in the lower asymptote region of the G-score curve, while triangles with higher S-scores should be placed in

[3] We do not refer to this value as the estimated noise standard deviation because we do not take the mean of the values in the window, nor do we square the difference between the observed and expected values. We observed that such a method significantly misestimates the true noise standard deviation.

the upper asymptote region. We chose a sigmoid as our G function because at a certain point, we do not want scores to increase because a large score value for a single triangle would hinder our algorithm's ability to locate change points indexed by smaller triangles.

4 The Algorithm

The Triangulator algorithm is shown in Fig. 3. The algorithm clearly runs in $O(n)$ time, which is an improvement over many methods to address this problem. It first reads in an array of clones A ordered by genomic position, and each element in A contains information about the clone's genomic location and it's log-2 ratio. The genomic locations for the clones are temporarily stored to an array P so that we can "normalize" the distances between the clones to ensure that all triangles formed are of the same width. The original genomic position values are indexed by the value of the new position, so at termination, the original positions are used in estimating change point locations.

For every three consecutive clones, the log-2 ratios for the clones are used to create a triangle. This step is shown on line 8. The *Triangulate* function simply creates a triangle from the three clones given as input where the first, second, and third parameters correspond to the first, second, and third points in the triangle, respectively ordered by their position on the x-axis. The G-score of each triangle is stored to the array M, and on line 11, the actual triangles are stored to the array T. Moreover, each triangle is indexed by its first point on the x-axis, as shown on line 9. This indexing facilitates the process of associating triangles with genomic locations.

Lines 14 through 17 identify an initial set of triangles that are candidates for breakpoint locations. For each triangle in the array, its G-score is compared to the mean G-score of all triangles in T. If the G-score for a given triangle is not within one standard deviation of the mean of all G-scores, then the triangle is a candidate for a true breakpoint. All such triangles are stored to the array T_a in increasing order by genomic position.

Lines 18 through 33 are the part of the algorithm that determines if the aberrant triangles in T_a are indexed at true change points or if they are simply indexed at noisy points. The index of the kth aberrant triangle is stored to e at line 20. The inner for-loop at line 25 loops between the index values for consecutive aberrant triangles. In this step, the segment of the chromosome between the indexed aberrant triangles is analyzed by triangulating the 1st point of the triangle at $T[e]$ with the 2nd and 3rd points of the triangle indexed by p. The score value from the resulting triangle is stored to the set R and this process is repeated for all points in the current segment. Referring to lines 31 through 33, if the aberrant triangle $T_a[k]$ is within 1 standard deviation of the mean of the score values in R, the clone indexed by $T_a[k]$ is identified as a change point and is added to the set C.

input : An array A of n DNA clones along a chromosome.

output: A set C of genomic locations (in kilobases) where the locations are predicted copy number break points.

```
1  M ← ∅;
2  P ← ∅;
3  T ← ∅;
4  for h ← 1 to n − 2 do
5  |  P[h] ← A_GenomicPosition[h];
6  |  A_GenomicPosition[h] ← h;
7  for i ← 1 to n − 2 do
8  |  τ ← Triangulate(A[i], A[i + 1], A[i + 2]);
9  |  τ_index ← i;
10 |  M[i] ← G(S(τ));
11 |  T[i] ← τ;
12 T_a ← ∅;
13 numAberrant ← 1;
14 for j ← 1 to n − 2 do
15 |  if M[j] > μ(M) + σ(M) or M[j] < μ(M) − σ(M) then
16 |  |  T_a[numAberrant] ← T[j];
17 |  |  numAberrant ← numAberrant + 1;
18 for k ← 1 to length(T_a) do
19 |  R ← ∅;
20 |  e ← T_{a_index}[k];
21 |  if e > 0 then
22 |  |  e ← T_{a_index}[k] − 1;
23 |  a ← T_point1[e];
24 |  a_GenomicPosition ← T_{a_index}[k] + 1;
25 |  for p ← T_{a_index}[k] + 1 to T_{a_index}[k + 1] − 1 do
26 |  |  b ← T_point2[p];
27 |  |  c ← T_point3[p];
28 |  |  τ ← Triangulate(a, b, c);
29 |  |  R ← R ∪ G(S(τ));
30 |  |  a_GenomicPosition ← a_GenomicPosition + 1;
31 |  if G(S(T_a[k])) ≤ μ(R) + σ(R) and G(S(T_a[k])) ≥ μ(R) − σ(R)
   |  then
32 |  |  pos ← T_{a_index}[k];
33 |  |  C ← C ∪ P[pos];
34 return C;
```

Fig. 3. CGH Triangulator Algorithm

5 Experiments

5.1 Study on Real Dataset: Coriell Cell Line

We first tested our algorithm on an array CGH dataset from the Coriell Institute that was first published in [7]. This dataset is annotated with true copy number state information which allowed for quantitative evaluation of the accuracy of our segmentation method. Moreover, it is important to have this information for real data so that we can make predictions as to the performance of our method on real datasets.

Design and Methods. To measure accuracy of predicted breakpoints, we measured the sensitivity (SE) as the proportion of true breakpoints identified to all real breakpoints, and we also measured the false discovery rate (FDR) as the proportion of falsely-predicted breakpoints to all predicted breakpoints. We also ran a Matlab implementation of the CBS algorithm and we compared the performance of the two programs in regards to accuracy and computation time. For the user parameters c and d, referenced in Section 3, we ran the experiment for $c = \{2, 3\}$ and for each value of c, we chose $d = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$. For $c > 3$, we noticed that the method returned a high number of false positives, and for $c = 1$, a low number number of true positives were returned. Thus, we focused our attention only on two values of c.

Results and Discussion. The results from this study are provided in Tables 1 and 2, and the results of the CBS algorithm on this dataset are given in Table 3. The tables suggest that the results are generally robust against different choices of parameters for c and d, though the proportion of false positives is higher for the $c = 3$ experiment. This is especially true for lower values of d when $c = 3$ because the S-score values are higher. This causes the G-score function to lose effectiveness in separating normal and aberrant triangles. We note, however, that the number of false positives decreased as d grew larger. In both sets of results, we observed that as d grows larger, the FDR will decrease before the method begins to lose sensitivity. Although not reported in Table 2, we observed that for $c = 3$, the value of d can increase to approximately 10000 before the method begins to lose sensitivity.[4] For $c = 2$, the drop in sensitivity occurs much sooner.

As suggested in [9], we used a variable w to represent the maximum allowed localization error of clones, so a breakpoint was correctly identified if it was within w clones of the true breakpoint. For both values of c, the triangulation method generally outperforms CBS with regard to FDR, while SE is never less than 0.80 for $w = 1$. For $w = 0$, the Triangulator algorithm is clearly more precise than the CBS algorithm on this dataset.

The sensitivity of both algorithms improved when we increased w from 0 to 1. This is mainly because the data does not have sharp breakpoints in some cases. For our algorithm, reasons for loss of sensitivity include lack of breakpoint

[4] For $c = 3$ and $d = 10000$, SE = 0.86, FDR = 0.22.

"sharpness" and segments that were of length 1. Our method relies on breakpoints that are defined by large, abnormal triangles. But on occasion, the breakpoints are defined by smaller triangles whose points are gradually increasing. There is also a loss of sensitivity with segments of length 1 because the method treats them as single, noisy data points. We observed that CBS will also miss segments of length 1. As previously stated, our reported FDR is generally smaller than that of CBS, which is a desirable feature.

In regards to time, our study confirmed our assumption that the Triangulator algorithm would be much faster than the CBS algorithm. In each case, our algorithm segmented the 15 cell lines in under 11 seconds, while the CBS algorithm needed over 25 minutes to segment these data.

Table 1. For $c = 2$. Results of Triangulator algorithm on Coriell data. Each test run required approximately 11 seconds.

d	SE, FDR ($w = 0$)	SE, FDR ($w = 1$)
100	0.77, 0.58	0.82, 0.55
200	0.80, 0.375	0.86, 0.32
300	0.80, 0.31	0.86, 0.25
400	0.77, 0.32	0.84, 0.26
500	0.72, 0.36	0.81, 0.26
600	0.70, 0.34	0.80, 0.25
700	0.70, 0.30	0.80, 0.20
800	0.70, 0.32	0.80, 0.24
900	0.70, 0.26	0.80, 0.16
1000	0.70, 0.32	0.80, 0.16

Table 2. For $c = 3$. Results of Triangulator algorithm on Coriell data. Each test run required approximately 11 seconds.

d	SE, FDR ($w = 0$)	SE, FDR ($w = 1$)
100	0.77, 0.80	0.82, 0.55
200	0.82, 0.71	0.89, 0.68
300	0.80, 0.65	0.86, 0.65
400	0.80, 0.60	0.86, 0.57
500	0.80, 0.59	0.86, 0.55
600	0.80, 0.57	0.80, 0.54
700	0.80, 0.51	0.86, 0.46
800	0.80, 0.49	0.86, 0.44
900	0.82, 0.47	0.86, 0.44
1000	0.82, 0.45	0.86, 0.41

Table 3. Results of the CBS algorithm on Coriell dataset. Results are consistent with those in [8].

Programs	SE, FDR ($w = 0$)	SE, FDR ($w = 1$)	Time
CBS	0.600, 0.769	0.900, 0.654	> 25m0s

5.2 Study on Simulated Dataset

The second part of our study involved the application of our algorithm to synthetic aCGH data. The simulation model used was that of Willenbrock *et al.* [9]. In this paper, they assume that not all cells in a sampled culture will be affected by copy number changes, but that only a proportion of cells will be diseased. Thus, they assume that the expected log-2 ratio of a clone is given by the following equation:

$$ratio := log_2[(c * P_t + 2 * (1 - P_t))/2]. \quad (3)$$

where c is the copy number state (0,1,2,3,4,5) and P_t is a proportion of tumorous cells typically seen in biopsies, sampled from a uniform distribution on [0.3,0.7]. After adding Gaussian noise sampled from $Normal(0, \sigma)$, where σ is sampled from $Uniform(0.1, 0.2)$, the final value of the clone's log ratio is determined.

Design and Methods. As in the study involving real aCGH data, we measured the sensitivity and false discovery rates of breakpoint prediction accuracy using the simulated data created from the model in [9]. However, we not only wanted to measure the accuracy of predicted breakpoints, but we wanted to determine the level of noise that would cause our algorithm to give undesirable results. We therefore altered the simulation model by assigning different values to the parameters of the uniform distribution used for the σ variable. The σ parameter is the standard deviation for the Gaussian noise added to the log-2 ratios. By restricting the values of these parameters at each step, we determine the suboptimal noise level by measuring sensitivity and false discovery rate of predicted breakpoints.

For this part of the experiment, we measured sensitivity and FDR for parameter (σ) values of [0.0,0.05], [0.0,0.1], [0.05,0.1], and [0.1,0.15]. For each of these parameter settings, we generated 500 samples with 20 chromosomes, each with 100 clones. This is similar to the experiment performed in [9]. As in the study on the Coriell data, we calculated SE and FDR for w = 0,1. We did not redo this experiment using CBS because the comparison paper [9] had done an extensive study of CBS on the simulated data. For this part of the study, we chose values of $c = \{2,3\}$ and $d = \{100,200,300,400,500\}$.

Results and Discussion. The results of this study are given in Tables 4 and 5. As seen in the tables, the method performs well on data with low to moderate noise levels, but loses accuracy as the amount of noise increases. Increasing w from 0 to 1 causes a slight increase in sensitivity, but a sharper decline in the number of false positives. This improvement from $w = 0$ to $w = 1$ is consistent with the results from the Coriell data, in which the results also improved when the offset value increased. It should be noted that the CBS algorithm gave more consistent results even with a noise level higher than those specified in our study (i.e., ($\sigma \in [0.1, 0.2]$). Our method would likely be best suited for microarray experiments in which the researcher can expect low to moderate noise levels. Examples of real data with such noise levels are the Coriell data, that we observed had a noise variance of around 0.1. Furthermore, we used a simulation model that assumed that a sample from a cell line contained only a portion of cells affected by copy number. As a result, the log-2 ratio values calculated for each clone were smaller than they would be if the assumption was that *all* cells in a culture were affected by copy number changes. Ultimately, if the data is very noisy, then triangles within noisy segments will be very large. Moreover, breakpoints will not be sharply defined by single, oblong triangles, but may instead be defined by multiple triangles that are indistiguishable from other triangles in the data. This lack of sharply defined breakpoints will certainly cause the method to lose sensitivity.

Table 4. For $c = 2$. Results of Study on Simulated Data. Each test run required approximately 5m30s.

$c = 2$			
d	Noise	SE,FDR (w=0)	SE,FDR (w=1)
100	0.0, 0.05	0.80, 0.02	0.82, 0.005
	0.0,0.1	0.77, 0.07	0.80, 0.03
	0.05,0.1	0.70, 0.17	0.76, 0.09
	0.1,0.15	0.55, 0.72	0.70, 0.64
200	0.0,0.05	0.75, 0.03	0.77, 0.006
	0.0,0.1	0.72, 0.07	0.75, 0.02
	0.05,0.1	0.65, 0.13	0.72, 0.05
	0.1,0.15	0.54, 0.64	0.67, 0.54
300	0.0,0.05	0.73, 0.03	0.74, 0.005
	0.0,0.1	0.69, 0.06	0.72, 0.02
	0.05,0.1	0.65, 0.12	0.69, 0.03
	0.1,0.15	0.52, 0.59	0.65, 0.49
400	0.0,0.05	0.71, 0.02	0.73, 0.006
	0.0,0.1	0.67, 0.07	0.71, 0.02
	0.05,0.1	0.61, 0.12	0.67, 0.03
	0.1,0.15	0.52, 0.56	0.64, 0.45
500	0.0,0.05	0.70, 0.03	0.715, 0.005
	0.0,0.1	0.65, 0.07	0.69, 0.01
	0.05,0.1	0.60, 0.11	0.66, 0.02
	0.1,0.15	0.51 0.53	0.63, 0.42

Table 5. For $c = 3$. Results of Study on Simulated Data. Each test run required approximately 5m30s.

$c = 3$			
d	Noise	SE,FDR (w=0)	SE,FDR (w=1)
100	0.0, 0.05	0.91, 0.02	0.92, 0.006
	0.0,0.1	0.86, 0.16	0.90, 0.12
	0.05,0.1	0.78, 0.35	0.86, 0.28
	0.1,0.15	0.56, 0.86	0.76, 0.80
200	0.0,0.05	0.88, 0.02	0.90, 0.005
	0.0,0.1	0.84, 0.12	0.88, 0.08
	0.05,0.1	0.76, 0.26	0.83, 0.18
	0.1,0.15	0.56, 0.86	0.76, 0.80
300	0.0,0.05	0.86, 0.02	0.88, 0.005
	0.0,0.1	0.82, 0.1	0.85, 0.06
	0.05,0.1	0.74, 0.21	0.82, 0.13
	0.1,0.15	0.56, 0.80	0.74, 0.73
400	0.0,0.05	0.85, 0.02	0.86, 0.005
	0.0,0.1	0.82, 0.09	0.85, 0.05
	0.05,0.1	0.74, 0.20	0.81, 0.11
	0.1,0.15	0.56, 0.78	0.73, 0.71
500	0.0,0.05	0.83, 0.02	0.85, 0.005
	0.0,0.1	0.80, 0.08	0.84, 0.04
	0.05,0.1	0.73, 0.18	0.80, 0.1
	0.1,0.15	0.56 0.76	0.72, 0.70

The simulated dataset appears to be more sensitive to choices of c than the Coriell dataset, though for both values of c, the method generally has a low FDR. Regarding sensitivity, lower values of d increase the number of true positives, though the number of true positives increases significantly for $c = 3$. For both values of c, the proportion of false positives was clearly higher in the Coriell dataset than in the simulated dataset. This difference can be attributed to differences between the two datasets, such as the number of copy number states, noise standard deviation, copy number state distribution, and the assumption of a proportion of affected cells.[5] These differences between the datasets likely account for the differences seen in the results, so it is important to know how these properties may affect results for microarray experiments.

Our implementation of the Triangulator algorithm required approximately 5m30s to process 500 simulated cell lines on a 3 GHz, 15 GB RAM server. Conversely, the Matlab implementation of CBS required approximately 6h40min on a 2.4 GHz, 3 GB machine. Although the Triangulator algorithm was implemented on a more powerful machine, the difference in computation time is still vast. More importantly, to identify copy number alterations with finer resolutions, researchers have investigated approaches using newer technologies such as SNP genotyping platforms [16,17] and next-generation sequencing techniques [18]. The number of SNPs from SNP chips and the number of reads generated from

[5] The simulation model used in our experiments does not assume that all cells in a culture are affected by copy number changes.

massive parallel sequencing machines are of a magnitude several orders greater than the number of clones based on aCGH techniques. To develop approaches for such datasets, it is important that they are fast and effective. Our linear algorithm has the potential to handle large datasets from these newer technologies.

6 Conclusion and Future Work

We have presented a linear-time algorithm for locating copy number breakpoints in array CGH data. Our results show that the method works best on data where the noise levels are not high. As a result, a researcher must have an idea of what kinds of noise levels to expect for her microarray experiments before employing such a method.

Future work will focus on increasing the robustness of the method against large amounts of noise. Possible ways of addressing this issue would be to triangulate every two or three log-2 ratios in addition to triangulating consecutive values. By taking the union of the results of each triangulation, the result should be more robust against breakpoints that are not sharply defined by a single triangle. Another direction is to investigate new score functions, or to explore alternative parameters for our existing score functions. More detailed analysis of false positives and false negatives may guide us to define new score functions that better distinguish true breakpoints from false ones. We will also investigate the applicability of our approach on datasets that are based on newer technologies such as those generated by massive parallel sequencing machines.

Acknowledgments

We would like to thank Xiaolin Yin for his help in performing our experiments. This work is supported in part by NIH/NLM (grant LM008991), NIH/NCRR (grant RR03655), NSF (grant CRI0551603) and a start-up fund from Case Western Reserve University.

References

1. Veltman, J.A., Fridlyand, J., Pejavar, S., Olshen, A., Korkola, J., DeVries, S., Carroll, P., Kuo, W., Pinkel, D., Albertson, D., Cordon-Cardo, C., Jain, A., Waldman, F.: Array-based comparative genomic hybridization for genome-wide screening of DNA copy number in bladder tumors. Cancer Res. 63, 2872–2880 (2003)
2. Whang-Peng, J., Kao-Shan, C., Lee, E., Bunn, P., Carney, D., Gadzar, A., Minna, J.: Specific chromosome defect associated with human small cell lung cancer; deletion 3p(14-23). Science 215, 181–182 (1982)
3. de Leeuw, R., Davies, J., Rosenwald, A., Bebb, G., Gascoyne, D., Dyer, M., Staudt, L., Martinez-Climent, J., Lam, W.: Comprehensive whole genome array cgh profiling of mantle cell lymphoma model genomes. Hum. Mol. Genet. 13(17), 1827–1837 (2004)

4. Fridlyand, J., Snijders, A., Ylstra, B., Li, H., Olshen, A., Segraves, R., Dairkee, Shanaz, Tokuasu, T., Ljung, B., Jain, A., McLenna, J., Ziegler, J., Chin, K., DeVries, S., Feiler, H., Gray, J., Waldman, F., Pinkel, D., Albertson, D.: Breast tumor copy number aberration phenotypes and genomic instability. BMC Cancer 6, 96 (2006)
5. Wang, Y., Makedon, F., Pearlman, J.: Tumor classification based on DNA copy number aberrations determined using SNP arrays. Oncology Reports 15, 1057–1061 (2006)
6. Pinkel, D., Albertson, D.G.: Array comparative genomic hybridization and its applications in cancer. Nat. Genet. 37, suppl. S11–S17 (2005)
7. 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., Albertson, D.: Assembly of microarrays for genome-wide measurement of DNA copy number. Nat. Genet. 3, 263–264 (2001)
8. Olshen, A., Venkatraman, E.: Circular binary segmentation for the analysis of array-based DNA copy number data. Biostatistics 5, 557–572 (2004)
9. Willenbrock, H., Fridlyand, J.: A comparison study: applying segmentation to array CGH data for downstream analyses. Bioinformatics 21(22), 4084–4091 (2005)
10. Lai, W., Johnson, M., Kucherlapati, R., Park, P.: Comparative analysis of algorithms for identifying amplifications and deletions in array CGH data. Bioinformatics 21, 3763–3770 (2005)
11. Venkatraman, E., Olshen, A.: A Faster Circular Binary Segmentation Algorithm for the Analysis of Array CGH Data. Bioinformatics 23(6), 657–663 (2007)
12. Guha, S., Li, Y., Neuberg, D.: Bayesian Hidden Markov Modeling of Array CGH Data. Harvard University Biostatistics Working Paper Series. Working Paper 24 (2006)
13. Fridlyand, J., Snijders, A., Pinkel, D., Albertson, D., Jain, A.: Hidden Markov models approach to the analysis of array CGH data. J. Multivar. Anal. 90, 132
14. Durbin, R., et al.: Biological Sequence Analysis. Cambridge University Press, Cambridge (1999)
15. Daruwala, R., Rudra, A., Ostrer, H., Lucito, R., Wigler, M., Mishra, B.: A versatile statistical analysis algorithm to detect genome copy number variation. Proc. Natl. Acad. Sci. 101(46), 16292–16297 (2004)
16. Redon, R., Ishikawa, S., Fitch, K.R., Feuk, L., Perry, G.H., Andrews, T.D., Fiegler, H., Shapero, M.H., Carson, A.R., Chen, W., Cho, E.K., Dallaire, S., Freeman, J.L., Gonzalez, J.R., Gratacos, M., Huang, J., Kalaitzopoulos, D., Komura, D., MacDonald, J.R., Marshall, C.R., Mei, R., Montgomery, L., Nishimura, K., Okamura, K., Shen, F., Somerville, M.J., Tchinda, J., Valsesia, A., Woodwark, C., Yang, F., Zhang, J., Zerjal, T., Armengol, L., Conrad, D.F., Estivill, X., Tyler-Smith, C., Carter, N.P., Aburatani, H., Lee, C., Jones, K.W., Scherer, S.W., Hurles, M.E.: Global variation in copy number in the human genome. Nature 444(7118), 444–454 (2006)
17. Carter, N.P.: Methods and strategies for analyzing copy number variation using dna microarrays. Nat. Genet. 39(suppl. 7), S16–S21 (2007)
18. Chiang, D.Y., Getz, G., Jaffe, D.B., O'Kelly, M.J., Zhao, X., Carter, S.L., Russ, C., Nusbaum, C., Meyerson, M., Lander, E.S.: High-resolution mapping of copy-number alterations with massively parallel sequencing. Nat. Methods 6(1), 99–103 (2009)