# Reduction Algorithm for the MLE for the Distribution Function of Bivariate Interval Censored Data

**Marloes H. Maathuis**[1]

**Department of Statistics, University of Washington, Seattle, WA 98195**

**Abstract**

We study computational aspects of the nonparametric maximum likelihood estimator (MLE) for the distribution function of bivariate interval censored data. The computation of the MLE consists of two steps: a parameter reduction step and an optimization step. In this paper we focus on the reduction step. We introduce two new reduction algorithms: the Tree algorithm and the HeightMap algorithm. The Tree algorithm is only mentioned briefly. The HeightMap algorithm is discussed in detail and also given in pseudo code. It is a very fast and simple algorithm of time complexity $O(n^2)$. This is an order faster than the best known algorithm thus far, the $O(n^3)$ algorithm of Bogaerts and Lesaffre (2003). We compare our algorithms with the algorithms of Gentleman and Vandal (2001), Song (2001) and Bogaerts and Lesaffre (2003), using simulated data. We show that our algorithms, and especially the HeightMap algorithm, are significantly faster. An R-package of the HeightMap algorithm is available from the author.

**Key words**: Computational Geometry; Maximal Clique; Maximal Intersection; Parameter Reduction.

# 1 INTRODUCTION

We consider the nonparametric maximum likelihood estimator (MLE) for the distribution function of bivariate interval censored data. Let $(X, Y)$ be the variables of interest and let $F$ be their joint distribution function. Suppose that there is a censoring mechanism, independent of $(X, Y)$, so that $(X, Y)$ cannot be observed directly. Thus, instead of a realization $(x, y)$, we observe a rectangular region $R \subset \mathbb{R}^2$ that is known to contain $(x, y)$. We call $R$ an *observation rectangle*. Our data consists of $n$ observation rectangles $R_1, \ldots, R_n$, and our goal is to compute the MLE $\hat{F}_n$ of $F$.

Let $P_F(R_i)$ denote the probability that the pair of random variables $(X, Y)$ is in observation rectangle $R_i$. Then, omitting the part that does not depend on $F$, we can write the log likelihood as

$$l_n(F) = \sum_{i=1}^{n} \log(P_F(R_i)), \tag{1}$$

and an MLE $\hat{F}_n$ is defined as

$$\hat{F}_n = \text{argmax}_{F \in \mathcal{F}} \;\; l_n(F),$$

[1]Marloes H. Maathuis is a Ph.D. student in the Department of Statistics, University of Washington, Seattle, WA 98195 (email: marloes@stat.washington.edu).

where $\mathcal{F}$ denotes the class of all bivariate distribution functions. As stated here, this is an infinite dimensional optimization problem. However, the number of parameters can be reduced by generalizing the reasoning of Turnbull (1976) for univariate censored data. By doing this (see e.g. Betensky and Finkelstein (1999), Wong and Yu (1999), Gentleman and Vandal (2002)) one can easily derive that:

- The MLE can only assign mass to a finite number of disjoint rectangles. We denote these rectangles by $A_1, \ldots, A_m$ and call them *maximal intersections*, following Wong and Yu (1999).
- The MLE is indifferent to the distribution of mass within the maximal intersections.

The second property implies that the MLE is non-unique, in the sense that we cannot determine the distribution of mass within the maximal intersections. Gentleman and Vandal (2002) call this *representational non-uniqueness.* Hence, we can at best hope to determine the amount of mass assigned to each maximal intersection. Let $\alpha_j$ be the mass assigned to maximal intersection $A_j$, and let $\alpha = (\alpha_1, \ldots, \alpha_m)$. Then $P_F(R_i)$ in equation (1) is simply the sum of the probability masses of the maximal intersections that are subsets of $R_i$:

$$P_F(R_i) = P_\alpha(R_i) = \sum_{j=1}^{m} \alpha_j 1_{\{A_j \subset R_i\}}.$$

We can then express the log likelihood in terms of $\alpha$:

$$l_n(\alpha) = \sum_{i=1}^{n} \log\Big(\sum_{j=1}^{m} \alpha_j 1_{\{A_j \subset R_i\}}\Big), \tag{2}$$

and an MLE $\hat{\alpha}$ is defined as

$$\hat{\alpha} = \text{argmax}_{\mathcal{K} \cap \mathcal{A}} l_n(\alpha), \tag{3}$$

where $\mathcal{K} = \{\alpha \in \mathbb{R}^m : \alpha_j \geq 0, j = 1, \ldots, m\}$, $\mathcal{A} = \{\alpha \in \mathbb{R}^m : \mathbf{1}^T \alpha = 1\}$ and $\mathbf{1}$ is the all-one vector in $\mathbb{R}^m$. This is an $m$-dimensional convex constrained optimization problem that does not need to have a unique solution in $\alpha$. This forms a second source of non-uniqueness in the MLE. Gentleman and Vandal (2002) call this *mixture non-uniqueness.*

Asymptotic properties of the MLE for univariate interval censored data have been studied by Groeneboom and Wellner (1992). In contrast to the consistency problems of the MLE for bivariate right censored data, the MLE for bivariate interval censored data has been shown to be consistent (see e.g. Van der Vaart and Wellner (2000), Song (2001)). This implies that the MLE can be used in practical applications to estimate the distribution function of bivariate interval censored data, for example to study mother-to-child transmission of HIV.

From the discussion above, it follows that the computation of the MLE consists of two steps. First, in the *reduction step*, we need to find the maximal intersections $A_1, \ldots, A_m$. This reduces the dimensionality of the problem. Then, in the *optimization step*, we need to solve the optimization problem defined in (3).

In this paper we focus on the reduction step. We distinguish between two types of reduction algorithms that reflect a trade-off between computation time and space:

- type 1: The reduction algorithm computes the maximal intersections $A_1, \ldots, A_m$.
- type 2: The reduction algorithm computes the *clique matrix*, an $m \times n$ matrix $C$ with elements $C_{ji} = 1_{\{A_j \subset R_i\}}$.

For $n$ observation rectangles, the number of maximal intersections is $O(n^2)$. Hence, given the observation rectangles, one can compute the clique matrix from the maximal intersections and vice versa in $O(n^3)$ time.

We need $O(n^2)$ space to store the maximal intersections, while we need $O(n^3)$ space to store the clique matrix. Thus, type 1 algorithms require an order of magnitude less space to store the output. On the other hand, the choice of reduction algorithm determines the amount of computational overhead in the optimization step, where the values of the indicator functions $1_{\{A_j \subset R_i\}}$ are needed repeatedly. Namely, using a type 1 algorithm requires repeated computation of these indicator functions, while such computations are avoided with a type 2 algorithm. Thus, if we use a type 1 reduction algorithm, the computational overhead in the optimization step is increased by a constant factor.

Finally, it should be noted that the clique matrix provides useful information about mixture uniqueness of the MLE. For example, properties of the clique matrix are used to derive sufficient conditions for mixture uniqueness by Gentleman and Geyer (1994) and Gentleman and Vandal (2002). We can also use the clique matrix to describe the equivalence class of solutions to (3). Let $\hat{\alpha}$ be a solution, and consider $\left(C^T \hat{\alpha}\right)_i = P_{\hat{\alpha}}(R_i)$, $i = 1, \ldots, n$. Since the log likelihood (1) is strictly concave in $P_F(R_i)$, the vector $C^T \hat{\alpha}$ is unique. Thus, the equivalence class of MLEs is exactly the set of all nonnegative vectors $\alpha$ for which $C^T \alpha = C^T \hat{\alpha}$, since these all yield the same likelihood value. In other words, it is the set $\left\{\alpha \geq 0 : \alpha = \hat{\alpha} + x, x \in \text{Null}\big(C^T\big)\right\}$. Note that we do not require explicitly that $\mathbf{1}^T \alpha = 1$, or equivalently, that $\mathbf{1}^T x = 0$. Namely, by induction it can be shown that $\mathbf{1}^T$ is contained in the row space of $C^T$. Hence, the fact that $x \in \text{Null}\big(C^T\big)$ implies $\mathbf{1}^T x = 0$ automatically.

We now give a brief overview of existing reduction algorithms. Betensky and Finkelstein (1999) provide a simple, but not very efficient, type 1 algorithm. Gentleman and Vandal (2002) introduce a type 2 algorithm of time complexity $O(n^5)$. Song (2001) proposes a type 1 algorithm that is of comparable speed. The algorithm with the best time complexity so far is the $O(n^3)$ type 1 algorithm of Bogaerts and Lesaffre (2003). Finally, Lee (1983) gives an $O(n \log n)$ algorithm for a somewhat different but related problem, namely that of finding the largest number of rectangles having a non-empty intersection.

In this paper, we introduce two new reduction algorithms. The algorithm we initially developed, the *Tree* algorithm, is only mentioned briefly. It is based on the algorithm of Lee (1983), and is a fast but complex type 2 algorithm. Later, we realized the reduction problem could be solved in a much simpler way if one is only interested in finding the maximal intersections. This resulted in the *HeightMap* algorithm, a very fast and simple type 1 algorithm of time complexity $O(n^2)$. We discuss this algorithm in detail and also give it in pseudo code. Finally, we compare the performance of our algorithms with the algorithms of Gentleman and Vandal (2001), Song (2001) and Bogaerts and Lesaffre (2003), using simulated data. We show that our algorithms, and especially the HeightMap algorithm, are significantly faster.

# 2 HEIGHT MAP ALGORITHM

Recall that we want to find the maximal intersections $A_1, \ldots, A_m$ of a set of observation rectangles $R_1, \ldots, R_n$. There exist several equivalent definitions for the concept of maximal intersection in the literature. Wong and Yu (1999) use the following: $A_j \neq \emptyset$ is a maximal intersection if and only if it is a finite intersection of the $R_i$'s such that for each $i$ $A_j \cap R_i = \emptyset$ or $A_j \cap R_i = A_j$. Gentleman and Vandal (2002) use a graph theoretic perspective: maximal intersections are the real representations of maximal cliques in the intersection graph of the observation rectangles.

We view the maximal intersections in yet another way. We define a *height map* of the observation rectangles. This height map is a function $h : \mathbb{R}^2 \to \mathbb{N}$, where $h(x, y)$ is defined to be the number of observation rectangles that overlap at the point $(x, y)$. The concept of the height map is illustrated in Figure 1. It is easily seen that the maximal intersections are exactly the local maxima of the height map. This observation forms the basis of our algorithm.

## 2.1 Canonical rectangles

We represent each rectangle $R_i$ as $(x_{1,i}, x_{2,i}, y_{1,i}, y_{2,i})$. The point $(x_{1,i}, y_{1,i})$ is the lower left corner and $(x_{2,i}, y_{2,i})$ is the upper right corner of the rectangle. We call $(x_{1,i}, x_{2,i}]$ the $x$-interval, and $(y_{1,i}, y_{2,i}]$ the $y$-interval of $R_i$. Furthermore, we use boolean variables $(c^x_{1,i}, c^x_{2,i}, c^y_{1,i}, c^y_{2,i})$ to indicate whether an endpoint is closed. As default we assume that left endpoints are open and right endpoints are closed, so that $(c^x_{1,i}, c^x_{2,i}, c^y_{1,i}, c^y_{2,i}) = (0, 1, 0, 1)$.

We now transform the observation rectangles $R_1, \ldots, R_n$ into *canonical rectangles* with the same intersection structure. We call a set of $n$ rectangles canonical if all $x$-coordinates are different and all $y$-coordinates are different, and if they take on values in the set $\{1, 2, \ldots, 2n\}$. An example of a set of canonical rectangles is given in Figure 1.

We perform this transformation as follows. We consider the $x$-coordinates and $y$-coordinates separately and replace them by their order statistics. The only complication lies in the fact that there might be ties in the data. Hence, we need to define how to break ties. We explain the basic idea using the examples given in Figure 2. In (a) we have an open left endpoint $x_{1,i}$ and a closed right endpoint $x_{2,j}$ with $x_{1,i} = x_{2,j}$ and $i \neq j$. Then the $x$-intervals of $R_i$ and $R_j$ have no overlap. Therefore, we sort the endpoints so that the corresponding canonical intervals have no overlap, i.e. we let $x_{2,j}$ be smaller. In (b) we have a closed left endpoint $x_{1,i}$ and a closed right endpoint $x_{2,j}$ with $x_{1,i} = x_{2,j}$ and $i \neq j$. Now the $x$-intervals of $R_i$ and $R_j$ do have overlap. Therefore, we sort the endpoints so that the corresponding

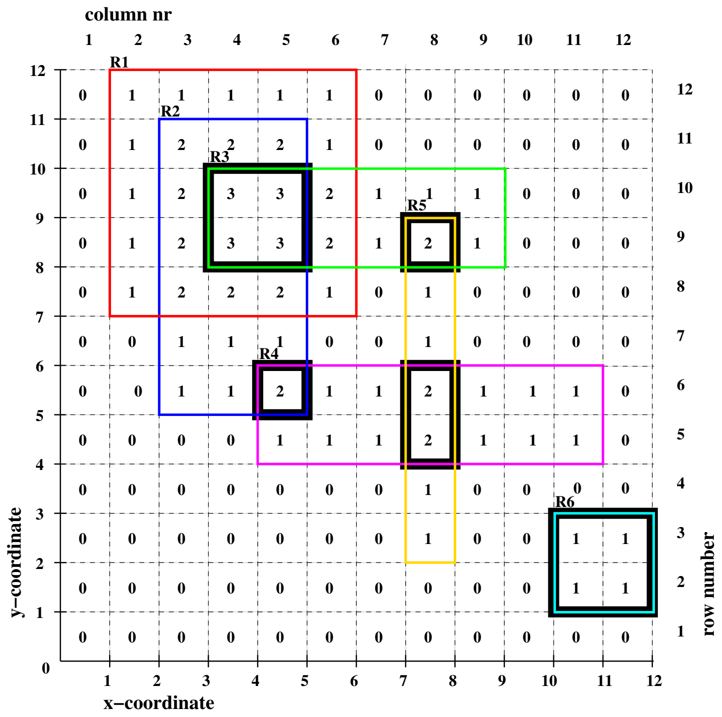


Figure 1: An example of six observation rectangles and their height map. The rectangles with the black fat borders are the maximal intersections. Note that they correspond exactly to the local maxima of the height map.

canonical intervals overlap, i.e. we let $x_{1,i}$ be smaller. In this way, we can consider all possible combinations of endpoints. By listing the results in a table, we found a compact way to code an algorithm for comparing endpoints. It is given in pseudo code (Algorithm 1).

The reason for transforming the observation rectangles into canonical rectangles is twofold. First, it forces us in the very beginning to deal with ties and with the fact whether endpoints are open or closed. As a consequence, we do not have to account for ties and open or closed endpoints in the actual algorithm. Second, it simplifies the reduction algorithm, since the column and row numbers in the height map directly correspond to the $x$- and $y$-coordinates of the canonical rectangles.

## 2.2 Building the height map

After transforming the rectangles, we build up the height map. To this end, we use a sweeping technique commonly used in the field of computational geometry (Lee 1983). By using this technique, we do not need to store the entire height map. Instead, we store one column at a time, in an array $h_1, \ldots, h_{2n}$. To build up the height map, we start with $h_1, \ldots, h_{2n} = 0$. This is column 1 of the height map. We then sweep through the plane, column by column, from left to right. Every time we move to a new column, we either enter or leave one observation rectangle. Thus, to compute the values of the height map in the next column, we respectively increment or decrement the values in the corresponding cells by 1. For example, when we move from the first to the second column of the height map in Figure 1, we enter

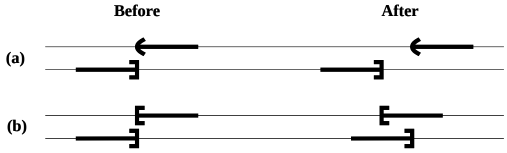


Figure 2: Breaking ties during the transformation of observation rectangles into canonical rectangles.

rectangle $R_1$. $R_1$ has $y$-interval $(7, 12]$ which corresponds to rows 8 to 12 in the height map. Hence, we increment $h_8, \ldots, h_{12}$ by 1.

## 2.3 Finding local maxima

During the process of building up the height map, we can find its local maxima, or equivalently, the maximal intersections. We denote the maximal intersections in the same way as the observation rectangles: $A_j = (x_{1,j}, x_{2,j}, y_{1,j}, y_{2,j})$. Now suppose we apply the sweeping technique, and currently are in column 5. We then are about to leave rectangle $R_2$. The $y$-interval of $R_2$ is $(5, 11]$ and corresponds to rows 6 to 11 in the height map. Hence, the values of the height map are going to decrease by 1 in rows 6 to 11, and will not change in the remaining rows. Since the values of the height map are going to decrease, we may leave areas of local maxima. Therefore, we need to look for local maxima in rows 6 to 11 of column 5. We find two local maxima: the cell in row 6, and the cells in rows 9 and 10. These local maxima in column 5 correspond to local maxima in the height map, say $A_1$ and $A_2$ respectively. For $A_1$, we know that $(y_{1,1}, y_{2,1}) = (5, 6)$ and for $A_2$ we know that $(y_{1,2}, y_{2,2}) = (8, 10)$. Furthermore, from the fact that we currently are in column 5, we know that $x_{2,1} = x_{2,2} = 5$. Finally, we obtain the values of $x_{1,1}$ and $x_{1,2}$ from the left boundaries of the rectangles that were last entered. For the cell in row 6 this is $R_4$ with left boundary 4. Hence, $A_1 = (4, 5, 5, 6)$. For the cells in rows 9 and 10, we last entered rectangle $R_3$ with left boundary 3. Hence, $A_2 = (3, 5, 8, 10)$. From this example we see that we need an additional array, $e_1, \ldots, e_{2n}$, where $e_k$ contains the index of the rectangle that was last entered in row $k$ of the height map.

After finding the first local maxima we can continue the above procedure. However, we first need to have an increase in the values of the height map before we can encounter any new local maxima. We illustrate this continuing in Figure 1, column 6, when we are about to leave rectangle $R_1$ with $y$-interval $(7, 12]$. Hence, applying the above procedure, we look for local maxima in rows 8 to 12 of column 6, and we find a maximum in rows 9 and 10. However, this does not correspond to a local maximum in the height map. It merely is a remainder from the maximal intersection $A_2$ that we found earlier, because there has not been an increase in the values of $h_9$ and $h_{10}$ since the output of $A_2$. In the algorithm, we block the output of such pseudo local maxima as follows. When we leave a rectangle $R_i$ with $y$-interval $(y_{1,i}, y_{2,i}]$, we set $e_k := 0$ for $k = y_{1,i} + 1, \ldots, y_{2,i}$. However, this is not yet enough. Also within a column, we need to make sure that there has been an increase in the values of the height map before we output a new maximal intersection. Therefore, walking down from $k = y_{1,i}$ to $k = 0$, we set $e_k := 0$ as long as

$h_k$ does not increase. Similarly, walking up from $k = y_{2,i} + 1$ to $2n$, we set $e_k := 0$ as long as $h_k$ does not increase.

Summarizing, we sweep through the plane from left to right, column by column. At each step in the sweeping process we either enter or leave a canonical rectangle. When we enter a rectangle $R_i$ with $y$ interval $(y_{1,i}, y_{2,i}]$, we increment $h_k$ by 1 and set $e_k := i$ for $k = y_{1,i} + 1, \ldots, y_{2,i}$. When we leave a rectangle $R_i$, we first look for local maxima in $h_k$ for $k = y_{1,i} + 1, \ldots, y_{2,i}$. If the corresponding values in $e_k$ are different from 0, we output the maximal intersection(s). Subsequently, walking down from $k = y_{1,i}$ and walking up from $k = y_{2,i} + 1$, we set $e_k := 0$ as long as $h_k$ does not increase. Finally, we decrement $h_k$ by 1 and set $e_k := 0$ for $k = y_{1,i} + 1, \ldots, y_{2,i}$. The algorithm is given in pseudo code (Algorithm 2). An R-package of the algorithm is available at http://www.stat.washington.edu/marloes.

### 2.4 Time and space complexity

We can easily determine the time and space complexity of the algorithm. In order to transform a set of rectangles into canonical rectangles, we need to sort the endpoints of their $x$-intervals and $y$-intervals. This takes $O(n \log n)$ time and $O(n)$ space. At each step in the sweeping process, we need to update the arrays $h$ and $e$ of length $2n$. We may also need to check for local maxima in at most $2n$ cells. Thus the time complexity of one sweeping step is $O(n)$. Combining this with the fact that the number of sweeping steps is $O(n)$ gives a total time complexity of $O(n^2)$. With respect to the space complexity, we need to store the arrays $h$ and $e$. Hence, the space complexity for computing the maximal intersections is $O(n)$. However, storing the maximal intersections takes $O(n^2)$ space.

## 3 EVALUATION OF THE ALGORITHMS

We compared our algorithms with the algorithms of Gentleman and Vandal (2001), Song (2001), and Bogaerts and Lesaffre (2003), using simulated data. We generated bivariate current status data according to a very simple exponential model:

$$X, Y, U, V \sim \exp(1), \tag{4}$$

where $X$ and $Y$ are the variables of interest, $U$ is the observation time for $X$, $V$ is the observation time for $Y$, and $X$, $Y$, $U$ and $V$ are mutually independent. Thus, the observation rectangles were generated as follows:

$$\begin{array}{llcl} X_i \le U_i, & Y_i \le V_i & \Rightarrow & R_i = (0, U_i, 0, V_i) \\ X_i \le U_i, & Y_i > V_i & \Rightarrow & R_i = (0, U_i, V_i, \infty) \\ X_i > U_i, & Y_i \le V_i & \Rightarrow & R_i = (U_i, \infty, 0, V_i) \\ X_i > U_i, & Y_i > V_i & \Rightarrow & R_i = (U_i, \infty, V_i, \infty) \end{array}$$

| $n$ | Gentleman&Vandal mean | Gentleman&Vandal sd | Song mean | Song sd | Bogaerts&Lesaffre mean | Bogaerts&Lesaffre sd | Tree mean | Tree sd | HeightMap mean | HeightMap sd |
|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 0.0004 | 0.0028 | 0.029 | 0.011 | 0.0010 | 0.0042 | 0.0012 | 0.0044 | 0.0002 | 0.0014 |
| 100 | 0.001 | 0.0036 | 0.52 | 0.14 | 0.0052 | 0.0079 | 0.0036 | 0.0072 | 0.0006 | 0.0031 |
| 250 | 0.061 | 0.015 | 26.0 | 47.0 | 0.083 | 0.014 | 0.016 | 0.0053 | 0.0016 | 0.0051 |
| 500 | 1.3 | 0.48 | 540.0 | 100.0 | 0.91 | 0.11 | 0.058 | 0.0087 | 0.0062 | 0.0083 |
| 1,000 | 46.0 | 63.0 | NA | NA | 13.0 | 1.0 | 0.29 | 0.032 | 0.020 | 0.008 |
| 2,500 | NA | NA | NA | NA | 470.0 | 30.0 | 3.1 | 0.10 | 0.11 | 0.0085 |
| 5,000 | NA | NA | NA | NA | NA | NA | 25.0 | 0.37 | 0.40 | 0.013 |
| 10,000 | NA | NA | NA | NA | NA | NA | 180.0 | 2.7 | 1.5 | 0.019 |

Table 1: Mean and standard deviation of the user time in seconds, over 50 simulations per sample size from model (4). Cells with NA indicate that simulations took over 1000 seconds to run and were therefore omitted.

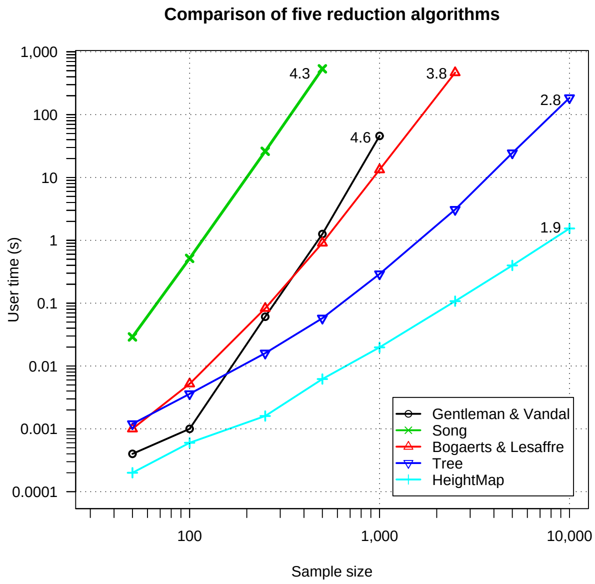


Figure 3: Log-log plot of the mean user time in seconds versus the sample size, over 50 simulations per sample size from model (4). For each algorithm, the estimated slope of its graph is given. These slopes can be used as empirical estimates of the time complexity of the algorithms.

We used sample sizes 50, 100, 250, 500, 1000, 5000 and 10000. For each sample size, we ran 50 simulations on a Pentium IV 2.4GHz computer with 512 MB of RAM and we recorded the user times of the algorithms. For each algorithm, we omitted sample sizes that took over 1000 seconds to run. All algorithms were implemented in C.

The results of the simulation are shown in Table 1. We see that the Tree algorithm, and especially the HeightMap algorithm are significantly faster than the other algorithms. The HeightMap algorithm runs sample sizes of $10,000$ in less than two seconds.

To get an empirical idea of the time complexity of the algorithms, Figure 3 shows a log-log plot of the mean user time versus the sample size. We fitted least squares lines through the last 4 points of each algorithm. The slopes of these lines can be used as empirical estimates of the time complexity of the algorithms. We see that the estimated slope of the HeightMap algorithm is 1.9, which agrees with the theoretical time complexity of $O(n^2)$ that we derived earlier. Furthermore, we see that the HeightMap algorithm is about an order faster than the Tree algorithm, which is about an order faster than the algorithm of Bogaerts and Lesaffre. Finally, note that the empirical time complexity of the algorithm of Bogaerts and Lesaffre is greater than the theoretical complexity of $O(n^3)$ that they derived.

# 4 DISCUSSION

The reduction step used to be a bottleneck in the computation of the MLE for bivariate interval censored data. With the development of the HeightMap algorithm this is no longer the case. Therefore, we think that further research on computational aspects should now be focused on the optimization step. The optimization step can be solved by a variety of algorithms. Gentleman and Vandal (2001) discuss various algorithms, viewing the problem either as a convex optimization problem or as a mixture problem. Song (2001) uses an infeasible interior point method. We have had positive experience with Sequential Quadratic Programming (SQP), using a vertex direction method to solve each quadratic optimization problem (Maathuis 2003). We recommend a comparison of these approaches and further research in this direction, also paying attention to finding good starting points for the optimization algorithms.


# ACKNOWLEDGEMENTS

This research was partly supported by NSF grant DMS-0203320. The author would like to thank Kris Bogaerts, Shuguang Song, and Alain Vandal for providing the code of their algorithms, and Piet Groeneboom, Steven Schimmel and Jon Wellner for their contributions, support and encouragement.


# APPENDIX: PSEUDO CODE

**Algorithm 1:** COMPAREENDPOINTS($A$,$B$):

***Input:*** Two endpoint descriptors $A = (x_{k,i}, c^x_{k,i})$ and $B = (x_{l,j}, c^x_{l,j})$

***Output:*** A boolean value indicating $A < B$

$c_A := (c^x_{k,i} = 1)$ { boolean indicating $A$ is a closed endpoint}
$c_B := (c^x_{l,j} = 1)$ { boolean indicating $B$ is a closed endpoint}
$r_A := (k = 2)$ { boolean indicating $A$ is a right endpoint}
$r_B := (l = 2)$ { boolean indicating $B$ is a right endpoint}
**if** $(x_{k,i} \neq x_{l,j})$ **then** { if the endpoints have different coordinates}
    **return** $(x_{k,i} < x_{l,j})$ { ...then let their coordinates determine their order}
**if** $(r_A = r_B$ **and** $c_A = c_B)$ **then** { if the endpoints are identical}
    **return** $(i < j)$ { ...then let their index determine their order}
**if** $(r_A \neq r_B$ **and** $c_A \neq c_B)$ **then** { if the endpoints are opposites}
    **return** $(r_A)$ { ...then $A < B$ when $A$ is a right endpoint}
**return** $(r_A \neq c_A)$ { otherwise $A < B$ when $A$ is closed left or open right}

**Algorithm 2:** HeightMapAlgorithm($R_1, \ldots, R_n$):
***Input:*** A set of $n$ observation rectangles $R_1, \ldots, R_n$
***Output:*** The corresponding maximal intersections $A_1, \ldots, A_m$

Transform observation rectangles into canonical rectangles $(x_{1,i}, x_{2,i}, y_{1,i}, y_{2,i})$, using CompareEndpoints
Sort $x_{1,i}, x_{2,i}$, $i = 1, \ldots, n$ in ascending order and store their indices $i$ in the list $r_1, \ldots, r_{2n}$
$m := 0$ { counts number of maximal intersections}
$h_1, \ldots, h_{2n} := 0$ { column of height map}
$e_1, \ldots, e_{2n} := 0$ { index of last entered rectangle; 0 blocks output}
**for** $j = 1$ to $2n$ **do** { sweep through height map from column 1 to $2n$}
  **if** ($r_j$ is a left boundary) **then** { we enter a rectangle}
    **for** $k = y_{1,r_j} + 1$ **to** $y_{2,r_j}$ **do** { update $h_k$ and $e_k$ for $k = y_{1,r_j} + 1, \ldots, y_{2,r_j}$}
      $h_k := h_k + 1; \quad e_k := r_j$
  **else** { we leave a rectangle}
    $b := y_{1,r_j}$ { bottom coordinate of local maximum; 0 blocks output}
    **for** $k = y_{1,r_j} + 1$ **to** $y_{2,r_j} - 1$ **do** { look for local maxima in rows $y_{1,r_j} + 1, \ldots, y_{2,r_j} - 1$}
      **if** ($h_{k+1} < h_k$ **and** $b > 0$) **then**
        **if** ($e_k > 0$) **then**
          $m := m + 1; \quad A_m := (x_{1,e_k}, j, b, k)$
        $b := 0$
      **if** ($h_{k+1} > h_k$) **then**
        $b := k$
    $k := y_{2,r_j}$
    **if** ($b > 0$ **and** $e_k > 0$) **then** { look for local maximum in row $y_{2,r_j}$}
      $m := m + 1; \quad A_m := (x_{1,e_k}, j, b, k)$
    $k := y_{1,r_j}$
    **while** ($k \geq 1$ **and** $h_k \leq h_{k+1}$) **do** { update $e_k$ for $k \leq y_{1,r_j}$}
      $e_k := 0; \quad k := k - 1$
    $k := y_{2,r_j} + 1$
    **while** ($k \leq 2n$ **and** $h_k \leq h_{k-1}$) **do** { update $e_k$ for $k > y_{2,r_j}$}
      $e_k := 0; \quad k := k + 1$
    **for** $k = y_{1,r_j} + 1$ **to** $y_{2,r_j}$ **do** { update $h_k$ and $e_k$ for $k = y_{1,r_j} + 1, \ldots, y_{2,r_j}$}
      $h_k := h_k - 1; \quad e_k := 0$
Transform the canonical maximal intersections $A_1, \ldots, A_m$ back to the original coordinates
**return** $A_1, \ldots, A_m$

# References


Betensky, R. A. and Finkelstein, D. M. (1999). "A Nonparametric Maximum Likelihood Estimator for Bivariate Censored Data," *Statistics in Medicine*, 18, 3089–3100.

Bogaerts, K. and Lesaffre, E. (2003). "A New Fast Algorithm to Find the Regions of Possible Mass Support for Bivariate Interval Censored Data," Technical report 0312, IAP Statistics Network.

Gentleman, R. and Geyer, C. J. (1994). "Maximum Likelihood for Interval Censored Data: Consistency and Computation," *Biometrika*, 81, 618–623.

Gentleman, R. and Vandal, A. C. (2001). "Computational Algorithms for Censored-Data Problems using Intersection Graphs," *Journal of Computational and Graphical Statistics*, 10, 403–421.

——— (2002). "Nonparametric Estimation of the Bivariate CDF for Arbitrarily Censored Data," *The Canadian Journal of Statistics*, 30, 557–571.

Groeneboom, P. and Wellner, J. A. (1992). *"Information Bounds and Nonparametric Maximum Likelihood Estimation,"* Birkhäuser, Boston.

Lee, D. T. (1983). "Maximum Clique Problem of Rectangle Graphs," *Advances in Computing Research*, 1, 91–107.

Maathuis, M. H. (2003). "Nonparametric Maximum Likelihood Estimation for Bivariate Censored Data," Master's thesis, Delft University of Technology, The Netherlands.

Song, S. (2001). *"Estimation with Bivariate Interval Censored data,"* Ph. D. thesis, University of Washington.

Turnbull, B. W. (1976). "The Empirical Distribution Function with Arbitrarily Grouped, Censored, and Truncated Data," *Journal of the Royal Statistical Association, Ser. B*, 38, 290–295.

Van der Vaart, A. W. and Wellner, J. A. (2000). "Preservation Theorems for Glivenko-Cantelli and Uniform Glivenko-Cantelli Classes," *High Dimensional Probability II*, 115–133, Birkhäuser, Boston.

Wong, G. Y. and Yu, Q. (1999). "Generalized MLE of a Joint Distribution Function with Multivariate Interval-Censored Data," *Journal of Multivariate Analysis*, 69, 155–166.