Fast Computation of the Minimum Covariance Determinant Estimator

Christoph Pesch
Universität Passau
D-94030 Passau
cpesch@stoch.fmi.uni-passau.de

Abstract

Robust estimation of location and scale in the presence of outliers is an important task in statistics. Rousseeuw introduced two estimators with high breakdown point, namely the minimum-volume-ellipsoid estimator (MVE) and the minimum-covariance-determinant estimator (MCD). While the MCD estimator has better theoretical properties than the MVE, the latter one appears to be used more widely. This may be due to the lack of fast algorithms for computing the MCD, up to now.

In this paper two branch-and-bound algorithms for the exact computation of the MCD estimator are presented. The results of their application to simulated samples are compared with a new heuristic algorithm and the steepest descent method suggested by Hawkins. The results show that multistart iterative trimming is a good and very fast heuristic for the MCD which can be applied to samples of large size.

1 Introduction

The problem of handling outliers is one of the most important of statistical interests. Reliable statements based on a statistical model are inpossible without a reliable estimation of the model parameters. We are concerned here with the estimation of location and scale. Consider the observations x_1, ..., $x_n \in \mathbb{R}^d$ of the random variables X_1, ..., X_n. A typical distributional model might be that the X_i are independent identically distributed with an elliptical symmetrical density. The following two estimators were introduced by Rousseeuw [7] as tools to estimate the mean of the X_i.

1.1 Definition A *k-Minimum-Volume-Ellipsoid estimator* or short *MVE(k)-estimator*, $k \in \{1, \ldots, n\}$, is a mapping $t\colon \mathbb{R}^{d\times n} \to \mathbb{R}^d$ with $t(x) =$ center of a smallest ellipsoid covering at least k points of $x = (x_1, \ldots, x_n) \in \mathbb{R}^{d\times n}$.
A *k-Minimum-Covariance-Determinant estimator* in short *MCD(k)-estimator*, $k \in \{1, \ldots, n\}$, is a mapping $t\colon \mathbb{R}^{d\times n} \to \mathbb{R}^d$ with $t(x) =$ mean of k points of x for which the determinant of the covariance matrix is minimal.
The corresponding k points will be called a *MCD(k) subset*. Note that this set is not necessarily unique.

To obtain a robust estimation of location, the parameter k should be a lower bound for the number of non-outliers. The aim of these estimators is to find a representative subset which includes no outliers. Estimation of the mean can then be based on this subset. To obtain the highest possible breakdown point $k := \lfloor (n+d+1)/2 \rfloor$ can be used. See Rousseeuw and Leroy [8].
To estimate the covariance, we can use the sample covariance of a MCD subset. The entries of this matrix will usually be too small. Rousseeuw and Leroy [8] propose to multiply the matrix with the factor $(\chi^2_{d,0.5})^{-1} m$ where $\chi^2_{d,0.5}$ is the median of the χ^2-distribution with d degrees of freedom and m is the median of the squared sample Mahalanobis distances.

MVE($\lceil n/2 \rceil$)	MCD($\lceil n/2 \rceil$)
permutation invariant	permutation invariant
affine equivariant	affine equivariant
breakdown point $\lfloor \frac{n}{2} - d + 1 \rfloor / n$	breakdown point $\lfloor \frac{n}{2} - d + 1 \rfloor / n$
$n^{1/3}$-consistent	$n^{1/2}$-consistent
	ML-estimator of a $\lfloor n/2 \rfloor$-outlier model
useful fast heuristics	only slow heuristics

Figure 1: Comparison of MVE and MCD

Figure 1 compares the MCD and the MVE estimators. The MCD estimator has a higher order of convergence; see Butler, Davies and Jhun [1] and Davies [2]. It is the maximum likelihood estimator of a $\lfloor n-k \rfloor$-outlier model where the non-outliers are normally distributed. Therefore the MCD estimator has the better theoretical properties. Nonetheless the MVE estimator appears to be more widely used. This may be due to the lack of fast algorithms for the MCD.
The main contributions of this paper are exact and heuristic algorithms for the fast computation of the MCD estimator.

2 Exact computation of the MCD estimator

A naive approach for true global optimization of the MCD is the exhaustive enumeration of all subsets of size k and the computation of their covariance matrix determinants. This approach can be implemented as a depth first search in a tree like the one in figure 2 which shows the case $n = 6$ and $k = 3$. Each leaf of this tree corresponds to a k-element subset of $\{1, \dots, n\}$ hence it will be called *subset tree*. At each leaf the covariance matrix and its determinant have to be computed.

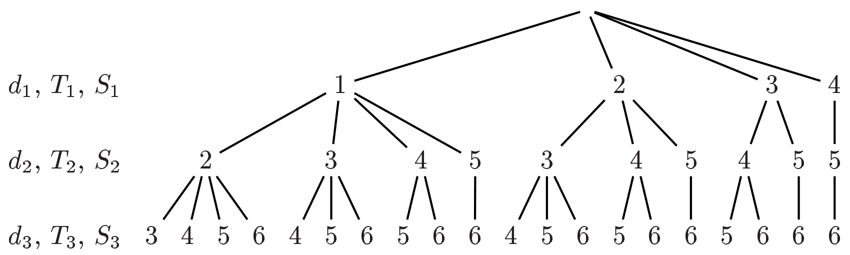

Figure 2: The subset tree

The resulting algorithm, denoted by *MCD-NA*, is shown in figure 3. Instead of the covariance V its multiple $S := kV$ is used. The determinant of S is k^d times the determinant of V, hence a MCD subset will minimize both $\det(V)$ and $\det(S)$.
To keep the number k of for-loops variable, a stack to store the $i_1, \dots, i_k$ can be used. We leave the details of implementation to the interested reader.
A speedup of the naive approach can be obtained using the following update formulas introduced by Youngs and Cramer [9]. Given a subset M of $\{x_1, \dots, x_n\}$ let $\mathrm{mean}(M)$ denote the sample mean, $\mathrm{var}(M) := \frac{1}{n}\sum(x_i - \mathrm{mean}(M))(x_i - \mathrm{mean}(M))^T$ the sample covariance matrix and $\mathrm{convexhull}(M)$ the convex hull of M. Let $T_i := i\,\mathrm{mean}(x_1, \dots, x_i)$, $S_i := i\,\mathrm{var}(x_1, \dots, x_i)$ and $d_i := \det(S_i)$. For each $i \in \{1, \dots, n-1\}$ we have

$$T_{i+1} = T_i + x_{i+1} \tag{1}$$

$$S_{i+1} = S_i + \frac{1}{i(i+1)}(T_i - ix_{i+1})(T_i - ix_{i+1})^T. \tag{2}$$

Using these formulas we can compute T_i and S_i in the i-th level of the subset tree in constant time given T_{i-1} and S_{i-1}. This approach results in a constant

Algorithm MCD-NA

Input: $k \in \{1, \dots, n\},\ x[1], \dots, x[n] \in \mathbb{R}^d$

begin

bestDet $:= \infty$;

for $i_1 := 1$ **to** $n - k + 1$ **do**

  for $i_2 := i_1 + 1$ **to** $n - k + 2$ **do**

    $\vdots$

    for $i_k := i_{k-1} + 1$ **to** n **do**

      $m := \frac{1}{k} \sum_{j=1}^{k} x[i_j]$;

      $S := \sum_{j=1}^{k} (x[i_j] - m)(x[i_j] - m)^T$;

      $d := \det(S)$;

      if $d <$ bestDet **then**

        bestDet $:= d$;

        $\mu := m$;

      fi

    od

    $\vdots$

  od

od

Output: μ;

end

Figure 3: Algorithm MCD-NA

speedup of the naive algorithm. We denote this algorithm by MCD-FA, see figure 4.
To deal with larger sample sizes efficiently, the number of subsets that need to be computed must be reduced. The following lemma shows a simple bounding heuristic for this task.

2.1 Lemma *Let* $x_1, \ldots, x_k \in \mathbb{R}^d$ *and* $d_i := \det(i \operatorname{var}(x_1, \ldots, x_i))$. *Then*

$$\forall i \in \{1, \ldots, k-1\}: \quad d_{i+1} \geq d_i. \tag{3}$$

Proof: Equation (2) gives

$$\begin{aligned} d_{i+1} &= \det(S_{i+1}) \\ &= \det\left(S_i + \frac{1}{i(i+1)}(T_i - i x_{i+1})(T_i - i x_{i+1})^T\right) \end{aligned}$$

Both summands are positive semidefinite, hence

$$\begin{aligned} d_{i+1} &\geq \det(S_i) + \det\left(\frac{1}{i(i+1)}(T_i - i x_{i+1})(T_i - i x_{i+1})^T\right) \\ &\geq \det(S_i) = d_i. \end{aligned}$$

□

In the i-th level of the subset tree the determinant d_i is computed and compared with the minimal determinant d_k found so far. See figure 2. If d_i is greater than or equal to this minimum, the search in the current branch of the tree can be skipped. This branch-and-bound approach can be enhanced by computing a good approximation of the minimal determinant for *bestDet* instead of setting it to ∞ at the beginning. An additional improvement can be achieved by sorting $x_1, \ldots, x_n$ in decreasing order according to their Mahalanobis distance wrt. the sample mean and covariance. Points near the boundary of the sample will then be considered first. Hence we can hope that more branches near the top of the tree can be skipped after sorting the points. Figure 5 shows the resulting algorithm MCD-BB.
A second bounding heuristic that can be used in low dimensions d will be derived in lemma 2.3 which is a consequence of the following theorem 2.2. It says that the substitution of some points of a subsample of $x_1, \ldots, x_n$ by points with lesser sum of the squared Mahalanobis distances decreases the determinant of the subsample covariance.

Algorithm MCD-FA

Input: $k \in \{1, \dots, n\},\ x[1], \dots, x[n] \in \mathbb{R}^d$

begin
  bestDet $:= \infty$;
  for $i_1 := 1$ **to** $n - k + 1$ **do**
    $T_1 := x[i_1]$;
    $S_1 := \mathbf{0}$;
    for $i_2 := i_1 + 1$ **to** $n - k + 2$ **do**
      $T_2 := T_1 + x[i_2]$;
      $S_2 := S_1 + \frac{1}{2}(T_1 - x[i_2])(T_1 - x[i_2])^T$;
      $\vdots$
      for $i_k := i_{k-1} + 1$ **to** n **do**
        $S_k := S_{k-1}$
            $+ \frac{1}{(k-1)k}(T_{k-1} - (k-1)x[i_k])(T_{k-1} - (k-1)x[i_k])^T$;
        $d := \det(S_k)$;
        if $d <$ bestDet **then**
          bestDet $:= d$;
          $\mu := \frac{1}{k}(x[i_k] + T_{k-1})$;
        fi
      od
      $\vdots$
    od
  od
  Output: μ;
end

Figure 4: Algorithm MCD-FA

Algorithm MCD-BB

Input: $k \in \{1, \ldots, n\},\ x[1], \ldots, x[n] \in \mathbb{R}^d$

begin

  bestDet $:= \infty$;

  for $i_1 := 1$ **to** $n - k + 1$ **do**

    $T_1 := x[i_1]$;

    $S_1 := \mathbf{0}$;

    for $i_2 := i_1 + 1$ **to** $n - k + 2$ **do**

      $T_2 := T_1 + x[i_2]$;

      $S_2 := S_1 + \frac{1}{2}(T_1 - x[i_2])(T_1 - x[i_2])^T$;

      if $\det(S_2) \geq$ bestDet **then** continue; **fi**

      $\vdots$

      for $i_k := i_{k-1} + 1$ **to** n **do**

        $S_k := S_{k-1}$

          $+ \frac{1}{(k-1)k}(T_{k-1} - (k-1)x[i_k])(T_{k-1} - (k-1)x[i_k])^T$;

        if $\det(S_k) \geq$ bestDet **then** continue; **fi**

        bestDet $:= \det(S_k)$;

        $\mu := \frac{1}{k}(x[k] + T_{k-1})$;

      od

      $\vdots$

    od

  od

  Output: μ;

end

Figure 5: Algorithm MCD-BB

2.2 Theorem *Let μ be the mean, V the positive definite covariance matrix of the sample $x_1, \ldots, x_k$ and $y_1, \ldots, y_m$ some points, $m \leq k$, such that $\tilde{V} := \operatorname{var}(y_1, \ldots, y_m, x_{m+1}, \ldots, x_k)$ is positive definite and*

$$\sum_{i=1}^{m}(x_i-\mu)^T V^{-1}(x_i-\mu) > \sum_{i=1}^{m}(y_i-\mu)^T V^{-1}(y_i-\mu).$$

Then $\det(V) > \det\left(\tilde{V}\right)$.

Proof: Without loss of generality let $\mu = \mathbf{0}$ and $V = I$ (otherwise we can use the transformation $x \mapsto V^{-1/2}(x-\mu)$). The assumption then becomes $\sum_{i=1}^{m} x_i^T x_i > \sum_{i=1}^{m} y_i^T y_i$. For $a \in \mathbb{R}^d$ let

$$\tilde{V}(a) := \frac{1}{k}\left(\sum_{i=1}^{m}(y_i-a)(y_i-a)^T + \sum_{i=m+1}^{k}(x_i-a)(x_i-a)^T\right).$$

Then $\det\left(\tilde{V}\right) \leq \det\left(\tilde{V}(a)\right)$; see Mardia, Kent and Bibby [5], exercise 1.4.2, page 22. The same is true for all sub-matrices $(\tilde{V}_{i,j})_{1\leq i,j\leq l}$, i.e.

$$\forall l \in \{1, \ldots, d\}: \quad \det\left((\tilde{V}_{i,j})_{1\leq i,j\leq l}\right) \leq \det\left((\tilde{V}(a)_{i,j})_{1\leq i,j\leq l}\right).$$

Because $\tilde{V}$ is positive definite, all these determinants are strictly positive. Hence $\tilde{V}(a)$ is positive definite and

$$\det\left(\tilde{V}(a)\right) \leq \left(\frac{\operatorname{tr}\left(\tilde{V}(a)\right)}{d}\right)^d; \tag{4}$$

see Mardia, Kent and Bibby [5], exercise 1.4.2, page 22. We now look at the case $a = \mathbf{0}$ and derive an upper bound for $\operatorname{tr}\left(\tilde{V}(\mathbf{0})\right)$.

$$\begin{aligned}
\operatorname{tr}\left(\tilde{V}(\mathbf{0})\right) &= \operatorname{tr}\left(\frac{1}{k}\left(\sum_{i=1}^{m} y_i y_i^T + \sum_{i=m+1}^{k} x_i x_i^T\right)\right) \\
&= \operatorname{tr}\left(\frac{1}{k}\sum_{i=1}^{k} x_i x_i^T - \frac{1}{k}\sum_{i=1}^{m}\left(x_i x_i^T - y_i y_i^T\right)\right) \\
&= \operatorname{tr}\left(I - \frac{1}{k}\sum_{i=1}^{m}\left(x_i x_i^T - y_i y_i^T\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=1}^{d} \left(1 - \frac{1}{k} \sum_{i=1}^{m} \left(x_{i,j}^2 - y_{i,j}^2 \right) \right) \\
&= d - \frac{1}{k} \sum_{i=1}^{m} \sum_{j=1}^{d} \left(x_{i,j}^2 - y_{i,j}^2 \right) \\
&= d - \frac{1}{k} \underbrace{\sum_{i=1}^{m} \left(x_i^T x_i - y_i^T y_i \right)}_{>0 \text{ by assumption}} \\
&< d
\end{aligned}
$$

Using (4) we have

$$
\det\left(\tilde{V}\right) \le \det\left(\tilde{V}(\mathbf{0})\right) \le \left(\frac{\operatorname{tr}\left(\tilde{V}(\mathbf{0})\right)}{d} \right)^d < \left(\frac{d}{d} \right)^d = 1 = \det(V)
$$

which completes the proof. $\square$

2.3 Lemma *Let $x_1, \ldots, x_n \in \mathbb{R}^d$ be lexicographically ordered and $1 \le k \le n$. For all MCD(k) subsets $\{x_{i_1}, \ldots, x_{i_k}\}$ with $i_1 < \cdots < i_k$, $\operatorname{var}(x_{i_1}, \ldots, x_{i_k})$ positive definite and all $1 \le j \le k$ we have*

$$
\text{open-convexhull}\left(x_{i_1}, \ldots, x_{i_j}\right) \cap \{x_1, \ldots, x_n\} \subseteq \{x_{i_1}, \ldots, x_{i_j}\}. \tag{5}
$$

Proof: Let $M := \{x_{i_1}, \ldots, x_{i_k}\}$, $\mu := \operatorname{mean}(M)$, $V := \operatorname{var}(M)$ and $\alpha := \max_{x \in M} (x - \mu)^T V^{-1} (x - \mu)$. Then $E := \{y \mid (y - \mu)^T V^{-1} (y - \mu) \le \alpha\}$ is an ellipsoid which contains M. With theorem 2.2 we have

$$
\forall x \in \{x_1, \ldots, x_n\} \setminus M : \quad (x - \mu)^T V^{-1} (x - \mu) \ge \alpha,
$$

because otherwise the covariance determinant could be decreased which is a contradiction to the assumption. Hence $\overset{\circ}{E} \cap \{x_1, \ldots, x_n\} \subseteq M$. Because E is a convex set, we have $\operatorname{convexhull}\left(x_{i_1}, \ldots, x_{i_j}\right) \subseteq E$. By assumption $x_1, \ldots, x_n$ are lexicographically ordered, hence $x_{i_l} \notin \operatorname{convexhull}\left(x_{i_1}, \ldots, x_{i_j}\right)$ for all

$l > j$. Therefore

$$\begin{aligned}
\text{open-convexhull}\big(x_{i_1}, \ldots, x_{i_j}\big) \cap \{x_1, \ldots, x_n\} \\
&\subseteq \Big(\overset{\circ}{E} \cap \{x_1, \ldots, x_n\}\Big) \setminus \{x_{i_{j+1}}, \ldots, x_{i_k}\} \\
&\subseteq M \setminus \{x_{i_{j+1}}, \ldots, x_{i_k}\} \\
&= \{x_{i_1}, \ldots, x_{i_j}\}.
\end{aligned}$$

□

If the x_i are lexicographically ordered, the convex hull of the first j points of a MCD subset contains none of the other $n - j$ points. Hence subtrees corresponding to subsets with a convex hull containing one or more of the $n - j$ points need not be considered. Figure 6 shows an example of a five element subset that need not be considered (on the left) and a five element subset that has to be considered (on the right).

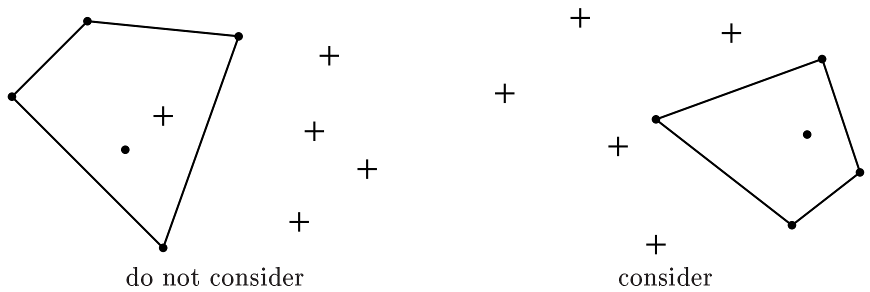

Figure 6: Bounding Heuristic

Algorithms for the incremental construction of the convex hull can be found in Preparata and Shamos [6].
We call the resulting sweepline algorithm *MCD-SW*. See figure 7. It can be combined with the first branch-and-bound algorithm MCD-BB in order to skip more subtrees. The disadvantage of this combination, denoted by *MCD-SWBB* (figure 8), is the necessity to compute all the d_i, $i \in \{1, \ldots, k\}$, and not just the d_k in the cases where the bounding heuristic (5) could not be applied.

Algorithm MCD-SW

Input: $k \in \{1, \dots, n\},\ x[1], \dots, x[n] \in \mathbb{R}^d$

begin

  sort $x[1], \dots, x[n]$ lexicographically;

  bestDet $:= \infty$;

  for $i_1 := 1$ **to** $n - k + 1$ **do**

    $R_1 := \{x[i_1]\}$;

    for $i_2 := i_1 + 1$ **to** $n - k + 2$ **do**

      $R_2 :=$ convexhull$(R_1, x[i_2])$;

      if $\overset{\circ}{R}_2 \cap \{x[1], \dots, x[n]\} \not\subseteq \{x[i_1], x[i_2]\}$ **then** continue; **fi**

      $\vdots$

      for $i_k := i_{k-1} + 1$ **to** n **do**

        $R_k :=$ convexhull$(R_k, x[i_k])$;

        if $\overset{\circ}{R}_k \cap \{x[1], \dots, x[n]\} \not\subseteq \{x[i_1], \dots, x[i_k]\}$ **then** continue; **fi**

        $d := \det(i\,\mathrm{var}(x[i_1], \dots, x[i_k]))$;

        if $d <$ bestDet **then**

          bestDet $:= d$;

          $\mu :=$ mean$(x[i_1], \dots, x[i_k])$;

        fi

      od

      $\vdots$

    od

  od

end

Figure 7: Algorithm MCD-SW

Algorithm MCD-SWBB

Input: $k \in \{1, \ldots, n\},\ x[1], \ldots, x[n] \in \mathbb{R}^d$

begin

  sort $x[1], \ldots, x[n]$ lexicographically;
  bestDet $:= \infty$;
  for $i_1 := 1$ **to** $n - k + 1$ **do**
      $T_1 := x[i_1]$;
      $S_1 := \mathbf{0}$;
      $R_1 := \{x[i_1]\}$;
      for $i_2 := i_1 + 1$ **to** $n - k + 2$ **do**
          $T_2 := T_1 + x[i_2]$;
          $S_2 := S_1 + \frac{1}{2}(T_1 - x[i_2])^2$;
          $R_2 :=$ convexhull$(R_1, x[i_2])$;
          if $\overset{\circ}{R_2} \cap \{x[1], \ldots, x[n]\} \not\subseteq \{x[i_1], x[i_2]\}$
             or $\det(S_2) \geq$ bestDet **then**
            continue;
          fi
          $\vdots$
          for $i_k := i_{k-1} + 1$ **to** n **do**
              $S_k := S_{k-1}$
                    $+ \frac{1}{(k-1)k}(T_{k-1} - (k-1)x[i_k])(T_{k-1} - (k-1)x[i_k])^T$;
              $d := \det(S_k)$;
              $R_k :=$ convexhull$(R_k, x[i_k])$;
              if $\overset{\circ}{R_k} \cap \{x[1], \ldots, x[n]\} \not\subseteq \{x[i_1], \ldots, x[i_k]\}$
                 or $d \geq$ bestDet **then**
                continue;
              fi
              bestDet $:= d$;
              $\mu :=$ mean$(x[i_1], \ldots, x[i_k])$;
          od
          $\vdots$
      od
  od
end

Figure 8: Algorithm MCD-SWBB

3 Heuristic Algorithms

In real world applications an algorithm for sample sizes n of some thousands and more and dimension d of some dozens is needed. Simulations show that the branch-and-bound algorithms introduced in this paper can not be used for this purpose in practice. See section 4. Hawkins [4] presented a steepest descent method to compute a MCD subset. In this method, denoted by MCD-SD, the current determinant can be decreased by exchanging an outlier against a non outlier. The pair with the largest decrease of the determinant is chosen. This is repeated until no such pair can be found. The result is a local minimum. Simulations show that the global minimum is reached with high probability. Unfortunately the average time complexity is quadratic wrt. the sample size. For larger sample sizes a faster heuristic is needed.
A widely used heuristic for robust estimation of location is the iterative trimming as introduced by Gnanadesikan and Kettenring [3]. See figure 9.

Algorithm Iterative Trimming
Input: $x_1, \ldots, x_n \in \mathbb{R}^d,\ k \in \{d+1, \ldots, n\}$
begin
  repeat
    $\mu := \text{mean}(x_1, \ldots, x_k)$;
    $\Sigma := \text{var}(x_1, \ldots, x_k)$;
    rearrange $x_1, \ldots, x_n$ such that $x_1, \ldots, x_k$ have
      the k smallest Mahalanobis distances wrt. μ, Σ;
  until no rearrangement was necessary;
  Output: $\text{mean}(x_1, \ldots, x_k)$
end

Figure 9: Iterative Trimming

In the case of different points with the same Mahalanobis distance they are sorted according to their indices. Theorem 2.2 shows that the determinant of Σ in the algorithm is strictly decreasing until termination. Therefore the algorithm can be seen as a heuristic algorithm for the MCD estimator.
Applied to different permutations of the $x_1, \ldots, x_n$ iterative trimming will in general compute different results. Therefore we can extend it by applying iterative trimming to many different random permutations and taking the minimum of the resulting covariance matrix determinants. We call this method *multistart iterative trimming* (MCD-MIT), see figure 10.

Algorithm MCD-MIT
Input: $x_1, \dots, x_n \in \mathbb{R}^d, \; k \in \{d+1, \dots, n\}, \; l \in \mathbb{N}$
begin
  for $i := 1$ **to** l **do**
    permute $x_1, \dots, x_n$ randomly;
    repeat
      $\mu := \text{mean}(x_1, \dots, x_k)$;
      $\Sigma := \text{var}(x_1, \dots, x_k)$;
      rearrange $x_1, \dots, x_n$ such that $x_1, \dots, x_k$ have
        the k smallest Mahalanobis distances wrt. μ, Σ;
    until no rearrangement was necessary;
    $d := \det(\text{var}(x_1, \dots, x_k))$;
    if $d <$ bestDet **then**
      bestDet $:= d$;
      $\mu := \text{mean}(x_1, \dots, x_k)$;
    fi
  od
  Output: μ
end

Figure 10: Algorithm MCD-MIT

A combination of this algorithm and Hawkins steepest descent algorithm is possible by using the result of iterative trimming as a starting point for steepest descent. This combination is denoted by MCD-MITSD.

4 Experimental Results

The algorithms have been implemented in C++ and compiled on a SUN Ultra 1 Model 140. Figure 11 shows the average running time of the exact algorithms applied to five different realizations of independent standard normal distributed samples in the two dimensional space. The algorithms computed the MCD($\lfloor n/2 \rfloor$) estimator. Note that the time axis is logarithmically scaled. The linear curve of the algorithm MCD-NA shows its exponential complexity. MCD-FA is about 8 times faster. It needs more than 22 hours for a sample of size 36. The branch-and-bound algorithms can reduce the exponential complexity significantly. The best algorithm for this case is MCD-SWBB. It needs less than 21 hours for samples of size 100.

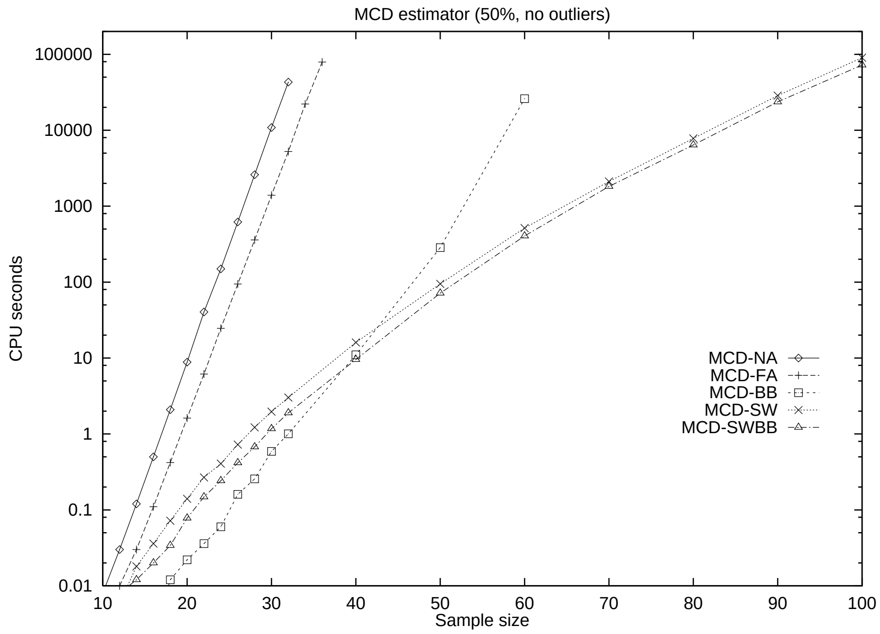

Figure 11: CPU time of the MCD(50%) algorithms

Figure 12 shows the results for the MCD($\lfloor 9n/10 \rfloor$) estimator. Due to the large value $\lfloor 9n/10 \rfloor$ of k, the bounding heuristic (3) can be used only near the leaves of the subset tree. Therefore the algorithm MCD-BB becomes less efficient and MCD-SW is faster than MCD-SWBB. The best algorithm for this case is MCD-SW. It needs less than seven hours for samples of size 200. Figure 13 shows the application of the algorithms to the MCD($9n/10$) estimator in the presence of 10% outliers. The algorithms MCD-BB and MCD-SWBB are much faster in the presence of outliers. MCD-SW and MCD-SWBB need approximately the same time. They need less than eight hours to compute the MCD estimator for samples of size 240.
To compare MCD-SD with MCD-MIT and MCD-MITSD each algorithm was applied to five different realizations of two dimensional standard normal distributed samples of size 100. To simulate the presence of outliers, 10 of these points are realizations of a centered normal distribution with covariance $36I$.

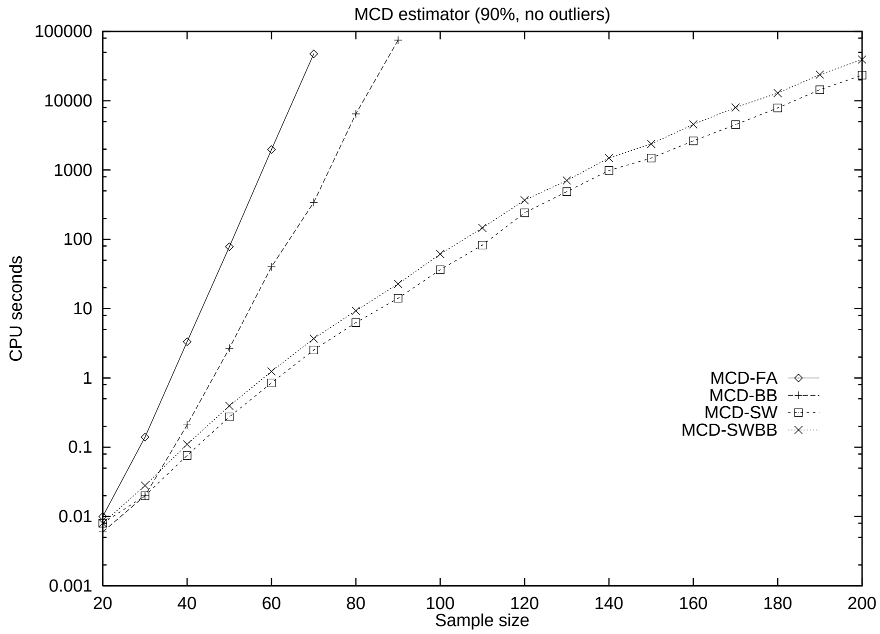

Figure 12: CPU time of the MCD(90%) algorithms

The algorithms tried to find the MCD(50) estimator and each one was started from 10 000 different permutations of the 100 points. Table 1 shows the minimal determinant (computed by the exact algorithms), the best value found by the heuristics, the relative error, the percentage of starts that resulted in the best value and the average computation time for one start in seconds. The practical performance of all three algorithms is very good. MCD-MIT is the fastest of the algorithms but could not find the global optimum in one case.
Table 2 shows the results for the MCD(180) estimator applied to samples of size 200 with 20 outliers. All three algorithms become more stable with $k = 180$ near to $n = 200$. Again MCD-MIT is the fastest of the algorithms but could not find the global optimum in one case.
Table 3 compares the heuristics for a sample size of 4 000 of which 400 points are realized as outliers. The algorithms tried to compute the MCD(2 000) estimator. MCD-MIT was applied to 10 000, MCD-MITSD to 1 000 and MCD-

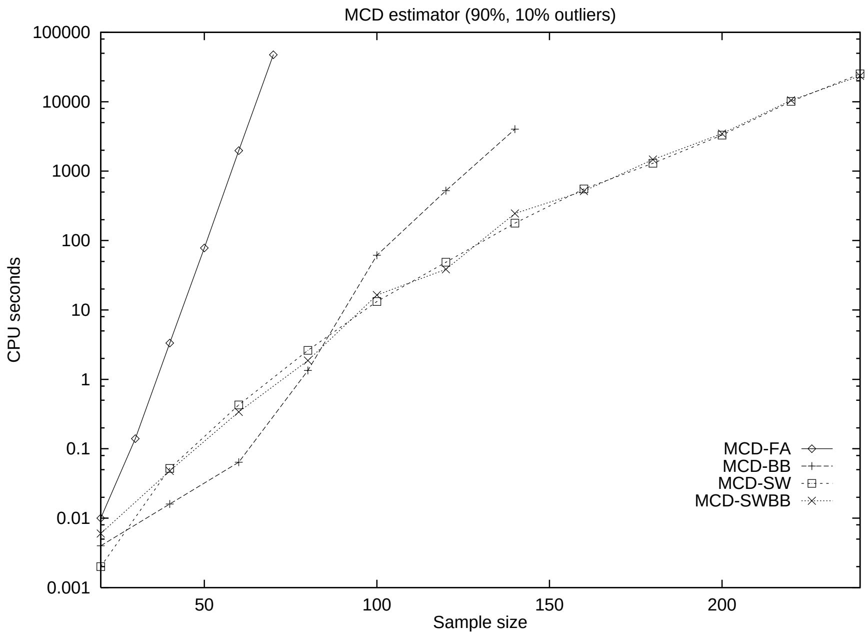

Figure 13: CPU time of the MCD(90%) algorithms, 10% outliers

SD to 10 starts. All three algorithms found the same minimum. MCD-MIT is about 30 000 times faster than MCD-SD. Therefore it can be applied to much more starts than MCD-SD. This significantly increases the probability of finding the global minimum. In this case MCD-MIT or MCD-MITSD should be the method of choice.

	Exact	**Heuristic**	**Error**	#	**Seconds**
			MCD-MIT		
1	0.080246	0.080246	0%	1.6%	0.00390
2	0.133001	0.133001	0%	6.3%	0.00512
3	0.113353	0.113353	0%	7.7%	0.00460
4	0.164044	0.164178	0.08%	12.4%	0.00516
5	0.136785	0.136785	0%	0.2%	0.00578
			MCD-MITSD		
1	0.080246	0.080246	0%	71.1%	0.03843
2	0.133001	0.133001	0%	100.0%	0.03273
3	0.113353	0.113353	0%	26.2%	0.02683
4	0.164044	0.164044	0%	0.8%	0.07012
5	0.136785	0.136785	0%	96.8%	0.04623
			MCD-SD		
1	0.080246	0.080246	0%	93.3%	0.23158
2	0.133001	0.133001	0%	100.0%	0.23692
3	0.113353	0.113353	0%	41.1%	0.22653
4	0.164044	0.164044	0%	39.6%	0.23272
5	0.136785	0.136785	0%	76.8%	0.22832

Table 1: Comparison of MCD-MIT, MCD-MITSD and MCD-SD ($n = 100$)

	Exact	Heuristic	Error	#	Seconds
		MCD-MIT			
1	0.828053	0.828053	0%	100%	0.00493
2	1.095320	1.095320	0%	100%	0.00800
3	0.992113	0.992113	0%	100%	0.00795
4	0.866272	0.866272	0%	100%	0.00959
5	0.890182	0.890194	0.001%	100%	0.00788
		MCD-MITSD			
1	0.828053	0.828053	0%	100%	0.01821
2	1.095320	1.095320	0%	100%	0.02141
3	0.992113	0.992113	0%	100%	0.02131
4	0.866272	0.866272	0%	100%	0.02324
5	0.890182	0.890182	0%	100%	0.03459
		MCD-SD			
1	0.828053	0.828053	0%	100%	0.23967
2	1.095320	1.095320	0%	100%	0.23982
3	0.992113	0.992113	0%	100%	0.24024
4	0.866272	0.866272	0%	100%	0.24281
5	0.890182	0.890182	0%	100%	0.24653

Table 2: Comparison of MCD-MIT, MCD-MITSD and MCD-SD ($n = 200$)

		MCD-MIT		**MCD-MITSD**		**MCD-SD**	
	Heuristic	**#**	**Sec.**	**#**	**Sec.**	**#**	**Sec.**
1	0.122268	12.07%	0.3488	19.8%	25.43	70%	13844.2
2	0.121194	12.85%	0.4275	24.9%	54.21	10%	13684.9
3	0.115212	5.75%	0.2791	28.7%	58.29	40%	13635.5
4	0.114514	3.21%	0.3034	5.2%	38.96	90%	13668.1
5	0.113181	27.44%	0.4517	100.0%	93.52	100%	13807.6

Table 3: Comparison of MCD-MIT, MCD-MITSD and MCD-SD ($n = 4000$)

5 Conclusion

The sweepline algorithms MCD-SW and MCD-SWBB presented in this text provide exact methods to compute the MCD estimator for small sample sizes. Sample sizes of some hundreds can be handled by a steepest descent method MCD-SD introduced by Hawkins [4]. For sample sizes of several thousands or more steepest descent becomes inadequate because of its high computation time. Multistart iterative trimming and the combination of iterative trimming and steepest descent as presented in this paper are able to close this gap and compute the MCD estimator with high probability in far less time. With these algorithms the MCD estimator becomes a usable alternative to the MVE estimator.
Further research on this topic will be contained in my Ph.D. thesis including the investigation of other optimization techniques like the Metropolis algorithm, simulated annealing or genetic algorithms.

References

[1] R. W. Butler, P. L. Davies, M. Jhun. Asymptotics for the Minimum Covariance Determinant Estimator. *The Annals of Statistics*, 21(3):1385–1400, 1993.

[2] Laurie Davies. The Asymptotics of Rousseeuw's Minimum Volume Ellipsoid Estimator. *The Annals of Statistics*, 20(4):1828–1843, 1992.

[3] R. Gnanadesikan, J. R. Kettenring. Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data. *Biometrics*, 28:81–124, March 1972.

[4] Douglas M. Hawkins. The feasible solution algorithm for the minimum covariance determinant estimator in multivariate data. *Computational Statistics & Data Analysis*, 17:197–210, 1994.

[5] K. V. Mardia, J. T. Kent, J. M. Bibby. *Multivariate Analysis.* Probability and mathematical statistics. Academic Press, 1979. ISBN 0-12-471252-5.

[6] Franco P. Preparata, Michael I. Shamos. *Computational geometry.* Texts and monographs in computer science. Springer Verlag, Corr. and expanded 2. print, 1988. ISBN 0-387-96131-3.

[7] Peter J. Rousseeuw. Multivariate Estimation with High Breakdown Point. In W. Grossmann, G. Pflug, I. Vincze, W. Werts, editors, *Mathematical statistics and applications: Proceedings of the 4th Pannonian Symposium on Mathematical Statistics, Bad Tatzmannsdorf, Austria*, Seiten 283–297. Reidel, 1983. ISBN 90-277-2088-6.

[8] Peter J. Rousseeuw, Annick M. Leroy. *Robust Regression and Outlier Detection.* Wiley series in probability and mathematical statistics. Applied probability and statistics. John Wiley & Sons, Inc., 1987. ISBN 0-471-85233-3.

[9] Edward A. Youngs, Elliot M. Cramer. Some Results Relevant to Choice os Sum and Sum-of-Product Algorithms. *Technometrics*, 13(3), August 1971.