On the efficient computation of robust regression estimators

Salvador Flores[*]

Université de Toulouse, UPS, Institut de Mathématiques, UMR CNRS 5219.
118 route de Narbonne, F-31062 Toulouse Cedex 9, France.

Abstract

The problem of providing efficient and reliable robust regression algorithms is considered. The impact of global optimization methods, such as stopping conditions and clustering techniques, in the calculation of robust regression estimators is investigated. The use of stopping conditions permits us to devise new algorithms that perform as well as the existing algorithms in less time and with adaptive algorithm parameters. Clustering global optimization is shown to be a general framework encompassing many of the existing algorithms.

Key words: Robust regression; clustering; global optimization; stopping conditions.

1. Introduction

Robust regression methods have been introduced to cope with the need left by classical techniques for methods that work well in the presence of contamination in the data. Particular interest has been focused on estimators with a high breakdown point as defined by Donoho and Huber (1983). Most of these methods combine robustness with desirable statistical properties such as consistency and asymptotic normality.Nonetheless, they are defined by means of difficult global minimization problems; hence, their computation is very time

[*]Tel.: +33 5 61 55 86 25; fax: +33 5 61 55 83 85.
Email address: `salvador.flores@math.univ-toulouse.fr` (Salvador Flores)

consuming. We address the problem of the efficient computation of robust estimators for statistical regression based on M-scales. The principal aim of our work is to investigate to what extent the most recent developments in the field of optimization can help improve the existing computational methods.

Let us consider the classical linear regression model

$$y_i = \mathbf{x}_i'\beta + \varepsilon_i, \quad i = 1, ..., n,$$

with errors terms ε_i identically distributed with zero center and independent from the covariates $\mathbf{x}_i$. We shall denote by y the vector with components y_i, by X the $n \times p$ matrix with ith row $\mathbf{x}_i$, and by r the vector of residuals $r(\beta) = y - X\beta$ with components $r_i = y_i - \mathbf{x}_i'\beta$.

Many robust estimators of the regression coefficients $\beta \in \mathbb{R}^p$ based on n independent observations $(y_i, \mathbf{x}_i) \in \mathbb{R} \times \mathbb{R}^p$ can be defined as:

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^p}{\text{Argmin}} \quad \hat{\sigma}(r(\beta)), \tag{P}$$

where $\hat{\sigma}$ is a scale estimator. An important case of (P) is the S-estimator, defined with $\hat{\sigma}(r) = s(r)$, where $s : \mathbb{R}^n \to \mathbb{R}_+$ is an M-estimator of scale, or M-scale (Huber, 1981), defined implicitly through:

$$\frac{1}{n} \sum_{i=1}^{n} \rho\left(\frac{r_i}{s(r)}\right) = b. \tag{1}$$

The function $\rho : \mathbb{R} \to \mathbb{R}_+$ is required to satisfy the following assumptions:

1. ρ is even and twice continuously differentiable,
2. $\rho(0) = 0$,
3. ρ is strictly increasing on $[0, c]$, for some $0 < c < +\infty$,
4. ρ is constant on $[c, \infty)$.

Any function satisfying these assumptions will be called in the sequel a "rho function".

The constant b is conveniently defined as

$$b = \mathbb{E}_\Phi(\rho), \tag{2}$$

where Φ denotes the standard normal distribution, in order to obtain consistency for the scale at the normal error model.

The S-estimator has very good robustness properties, but it lacks efficiency. Robustness is to be understood in the sense of the *breakdown point*, which is the minimum fraction of the observations that need to be shifted for the estimator to take on arbitrary values. In fact, there is a tradeoff between robustness and efficiency, and for a breakdown point of

50%, efficiency can be as low as 28.7% (Maronna et al., 2006, pp. 131). MM-estimators were introduced to fill this gap. They are obtained by local minimization of a suitable function using an S-estimator as starting point. In this way, they can combine efficiency with a high breakdown point. See Maronna et al. (2006, Sec. 5.5) for details. However, τ-estimators have lower bias curves, and their computing effort is comparable to computing the S-estimator, which is instrumental in the computation of MM-estimators.

In this paper, we shall focus on τ-estimators, introduced in Yohai and Zamar (1988), which are defined as minimizers of the function

$$\hat{\sigma}(\beta) = \frac{s(r(\beta))^2}{nb_2} \sum_{i=1}^{n} \rho_2 \left(\frac{r_i(\beta)}{s(r(\beta))} \right), \quad \beta \in \mathbb{R}^p, \tag{3}$$

where ρ_2 is a rho function and b_2 is adjusted to ρ_2 as in (2). This choice is motivated by the robustness and efficiency properties of this estimator. Indeed, τ estimators have a breakdown point $\epsilon^* = b/\rho_1(c)$; hence, for an adequate choice of the function ρ_1, the maximal breakdown point of 50% can be obtained. Similarly, by adjusting the function ρ_2, the asymptotic efficiency at the normal distribution can be made arbitrarily close to 1. However, computing τ-estimators involves solving a difficult global optimization problem. Roughly speaking, global optimization methods can be classified (Archetti and Schoen, 1984) into deterministic methods and stochastic or probabilistic methods. Deterministic methods look for a guaranteed global optimum, while stochastic methods settle for a point that is a global optimum within an allowed margin or with a certain probability. More ambitious, deterministic methods are time consuming, and their range of applicability is limited to very specific classes of problems, or to problems that are of small size. The interested reader should refer to Agulló (2001) for a deterministic robust regression algorithm.

Most of the existing methods for computing robust estimators are stochastic and, more specifically, based on random subsampling. These methods operate by computing candidates βs based on subsamples of the observations and then starting local minimizations from each of these candidates. Therefore, they are also called multistart methods.

Section 2 is devoted to the local minimization aspects of computing τ-estimators. We should stress that this is the only part of our paper that is estimator-specific. The global part of our discussion is relevant to any objective function $\hat{\sigma}$, provided that a way to perform local minimizations is available. Then, in Section 3, we will briefly describe clustering global optimization methods, which is a class of multistart methods that uses clustering analysis techniques (Törn and Žilinskas, 1989) in order to reduce the number of local minimizations needed to find a global minimum. Section 4 discusses stopping conditions for multistart methods. Section 5 shows how the existing methods for robust regression fit into the framework of clustering global optimization. In Section 6, we present a few numerical tests comparing the different methods. In particular, the effectiveness of clustering techniques and stopping conditions is evaluated. We finish with our conclusions in Section 7.

2. Local Minimization Issues

As already mentioned in the introduction, we focus on the problem of finding global minima of the function

$$\hat{\sigma}(\beta) = \frac{s(r(\beta))^2}{nb_2} \sum_{i=1}^{n} \rho_2 \left(\frac{r_i(\beta)}{s(r(\beta))} \right), \quad \beta \in \mathbb{R}^p,$$

where the *robust scale* $s(r(\beta))$ is implicitly defined by:

$$\frac{1}{n} \sum_{i=1}^{n} \rho_1(r_i(\beta)/s(r(\beta))) = b_1.$$

All the global optimization methods that we consider in this paper rely on local minimizations, and moreover, a major part of their computing time is spent in local minimizations. This is why fast and reliable local minimization algorithms are crucial. When the Hessian of the objective function is available, the most efficient local minimization algorithm is the Newton-Raphson method. Nevertheless, due to the flat parts present in the function ρ, the Hessian of the τ-objective function may contain large portions filled with zeros, and therefore, it is ill-conditioned. A good alternative for computing local minima of (3) is the *Iterated Reweighted Least Squares* (IRLS) algorithm (Salibian-Barrera et al., 2008). Although it has a slower rate of convergence, in practice it has proven to be quite efficient and stable. At each iteration, the IRLS algorithm solves a weighted least squares problem, which is equivalent to minimizing a quadratic local approximation of the objective function. In this section, we propose to use inexact solutions of the weighted least squares problems at each iteration, and we evaluate the gain in efficiency.

The IRLS step is derived from the necessary condition for optimality:

$$g_\tau(\beta) := \frac{\partial \hat{\sigma}}{\partial \beta} = 0.$$

An expression for $g_\tau(\beta)$ has been obtained in Yohai and Zamar (1988):

$$g_\tau(\beta) = \frac{-2}{n} X' W(\beta) r(\beta).$$

Here, $W(\beta)$ denotes the diagonal matrix with entries $w_j(\beta)$, where

$$w_j(\beta) = \frac{\omega(\beta)\rho_1'(e_j(\beta)) + \rho_2'(e_j(\beta))}{e_j(\beta)}, \tag{4}$$

with the following notations:

$$e_i(\beta) = \frac{r_i(\beta)}{s(\beta)}, \quad \omega(\beta) = \frac{\sum_{i=1}^n [2\rho_2(e_i(\beta)) - {\rho_2}'(e_i(\beta))e_i(\beta)]}{\sum_{i=1}^n \rho_1'(e_i(\beta))e_i(\beta)}.$$

This leads to the matrix form of the optimality condition:

$$X' W(\beta) X \beta = X' W(\beta) y. \tag{5}$$

Disregarding the fact that the matrix W depends (nonlinearly) upon β, the system of equations (5) are the normal equations associated to the *Weighted Least Squares* problem with weights W. It is a fixed point equation, so the iterative method

$$\beta_{k+1} = (X'W(\beta_k)X)^{-1}X'W(\beta_k)y \tag{6}$$

has been proposed to solve it in Salibian-Barrera et al. (2008). These are the IRLS iterations.

For each iteration, IRLS constructs a quadratic approximation of the true objective function, where the minimum of this quadratic approximation is the subsequent iterate. The computational cost of each iteration is equivalent to the cost of computing the weights (4) and solving the system (6). Computing the weights (4) is costly because this requires the computation of an M-estimator of scale. Therefore, Salibian-Barrera et al. (2008) have proposed to replace the M-scale in (4) with an approximate value whereby the resulting iterations are known as approximated IRLS iterations. Approximated IRLS solves the $p \times p$ linear system (6) for each iteration, which corresponds to finding the minimum of a quadratic local approximation of the true objective function. However, when p is not small, the computational burden of solving a linear system for each iteration can be non-negligible. Keeping in mind that our objective is to solve (5), an approximate solution of the instrumental subproblem (6) should be enough. In fact, replacing $y = r(\beta) + X\beta$ and setting $B_k = 2X'W(\beta_k)X/n$ in (6), we obtain

$$-B_k\beta_{k+1} = -B_k\beta_k + g_\tau(\beta_k).$$

In this form, the IRLS resembles the so called *quasi-Newton methods*, whose iterations are of the form (Nocedal and Wright, 1999) $-B_k\beta_{k+1} = -B_k\beta_k + \alpha_k g_\tau(\beta_k)$, for an adequate steplength $\alpha_k > 0$.

It has been proved (Nocedal and Wright, 1999, Subsec. 5.7.1) that for quasi-Newton methods, a "good enough" direction suffices to keep convergence.

In order to evaluate the efficacy of this approach, we compared the computing time required to perform local minimizations by solving approximately

(6), which will henceforth be denoted as the *Iterated Inexact Reweighted Least Squares* (IIRLS), with the computing time of the approximated IRLS iterations.

In our tests, we used the LSQR algorithm (Björck, 1996) to calculate approximated solutions to least squares problems. This algorithm can handle problems with non-square matrices, and it is stable and easily available on the Matlab environment.

In Figure 1, we show the results obtained from the IIRLS. The abbreviation `IIRLS5` (respectively `IIRLS20`) stands for the algorithm performing 5 (respectively 20) iterations of the LSQR algorithm for each approximated IRLS iteration. We denoted `IARLS` as the approximated IRLS algorithm introduced in Salibian-Barrera et al. (2008) that uses approximated M-scales for computing the weights. This is subsequently used to solve (6) using a direct method.

We plot, for different values of p from 5 to 200, the average time spent by each algorithm over 500 local minimizations of the function (3) with different datasets and different starting points generated as described in Section 6. In all the cases, the IIRLS algorithm found the same local minimum as IARLS, although it usually needed more iterations to converge. Nevertheless, the economy in time of performing inexact iterations compensated for the increase in the number of iterations.

We see in Figure 1 that for $p = 5$, there is not a great difference among the three methods, but the difference increases with p and becomes of great importance for p in the medium and large range. Considering the fact that the quality of the results is exactly the same, it is worthwhile to use IIRLS for local minimizations.

3. Clustering Methods

This section describes a technique for solving global optimization problems, such as (P), by incorporating clustering analysis techniques. The objective of clustering methods in global optimization (Törn and Žilinskas, 1989) is to identify groups of βs such that, if used as an initial point in a local minimization,

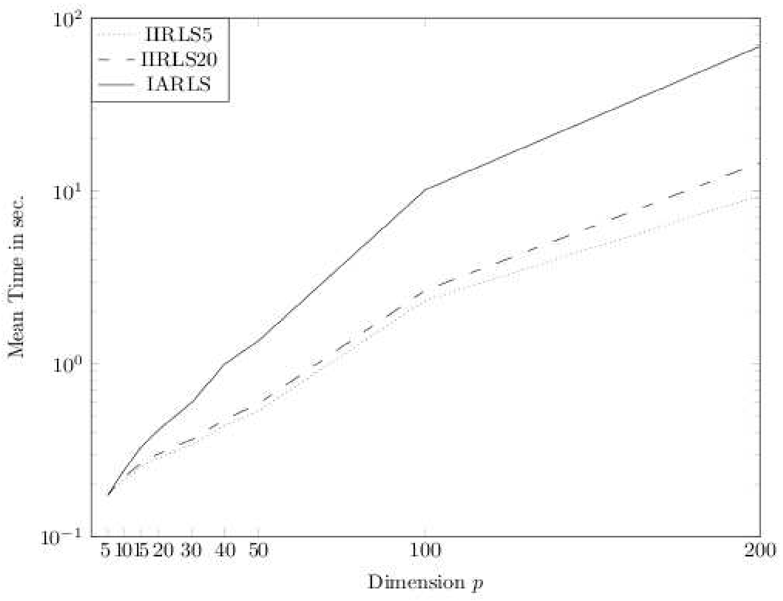

Figure 1: Average times needed to find a solution to (5) with respect to p, using approximated (`IARLS`) and inexact (`IIRLS5` and `IIRLS20`) iterations.

every member of the same group yields the same local minimum. Thus, it would suffice to perform only one local minimization from each of these groups in order to locate all local minima, and the best of these minima would be a global minimum. A schematic explanation of clustering methods is presented in Figure 2.

It consists in repeating the following steps iteratively, which we shall describe in detail in the rest of the section.

1. Sampling: sample candidates βs. Add them to the candidates sampled in previous iterations.
2. Concentration and/or selection: concentrate the sampled candidates around the minima in order to facilitate the clustering.
3. Clustering: identify groups of candidates suspected to converge towards the same minimum.
4. Stopping condition: decide whether it is worthwhile to continue, taking

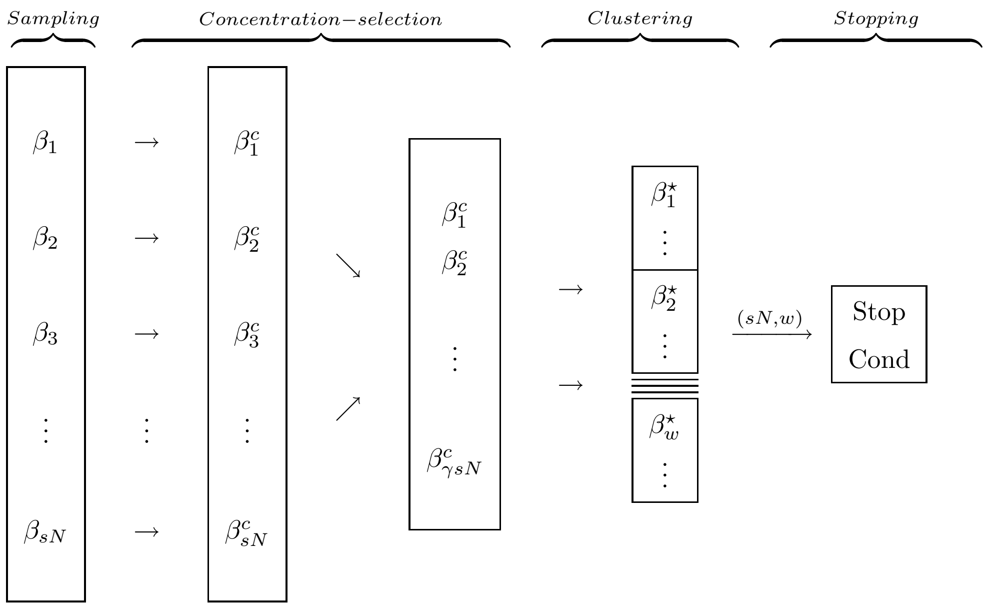

Figure 2: Schematic representation of clustering methods, at the sth iteration.

into account the outcome of the local minimizations. If so, go back to (1), otherwise stop.

Let us describe in detail the first three steps: sampling, concentration/selection and clustering. For clarity of the exposition, stopping conditions will be discussed separately in the next section.

3.1. Sampling

In absence of additional information, candidates are usually sampled uniformly in the feasible region. However, when the feasible region is unbounded, it is not completely clear how the sampling should be done.

For the particular case of robust regression, Rousseeuw (1984) proposed a method known as *random subsampling*: candidates $\beta_i, i = 1, ..., N$, are constructed by drawing random subsamples of size $h \geq p$ from the data and letting β_i be the least squares fit to the ith subsample. The rationale behind this method is that for a large enough N, at least one outlier-free subsample could be sampled, which should give a good candidate, hopefully in the neighborhood of a global minimum.

3.2. Concentration and/or selection

The concentration step consists of performing some iterations of a local minimization procedure, usually one, starting from each candidate. In the selection step, a prespecified fraction of the sampled candidates with lowest function value is retained. It has been proposed (Törn and Žilinskas, 1989) to do only the concentration step, only the selection step or both. For robust regression, Ruppert (1992) proposed to do selection and then concentration, while Salibian-Barrera et al. (2008) explored the use of concentration followed by selection. The term "concentration step" has already been used in Rousseeuw and Van Driessen (2000) to denote a particular local minimization procedure for the LTS estimator. In the sequel, we will use this term in the broader context described previously.

3.3. Clustering

Many ways to perform clustering for global optimization have been proposed (Törn and Žilinskas, 1989). Here, we review only Single Linkage, which is the simplest method (Rinnooy Kan and Timmer, 1987).

Single Linkage

At each iteration s, compute the radius $r_s = \frac{1}{\sqrt{\pi}}\left(\Gamma(1+p/2)\frac{\xi \ln(sN)}{sN}\right)^{1/p}$, for some $\xi > 0$. Where Γ denotes the gamma function.
The single linkage algorithm consists in iterating the following three steps until all points have been assigned to a cluster

1. Choose a local minimum to be used as seed.
2. Initialize the cluster with the seed.
3. Grow cluster : given a partially constructed cluster,
   iterate :

(a) find the unclustered point closest to the cluster.

(b) if this point is within distance r_s from the cluster, add it to the cluster and repeat (3a). Otherwise, go back to step (1) and start the next cluster.

The theoretical convergence properties of Single Linkage have been proved to minimize a function over a bounded set S, supposing that the initial candidates have been sampled uniformly over S (and that no concentration step has been performed). The proof proceeds by estimating the probability that a local minimization is started from a point a at iteration s. This probability is bounded by the probability that there exists another candidate within distance r_s with a lower function value. This is because if the ball with center a and radius r_s contains a candidate z with lower function value, then if z is assigned to a cluster, a will be assigned to the same cluster. Moreover, if in step (1) the local minimizations are performed by first considering the candidates with lower function values, then we will not apply a local minimization to a before z is assigned to a cluster.

Therefore, the choice

$$r_s = \frac{1}{\sqrt{\pi}} \left(\Gamma\left(1 + \frac{p}{2}\right) \xi \text{Volume}(S) \frac{\ln(sN)}{sN} \right)^{1/p}, \ \xi > 0,$$

makes the probability of applying a local minimization to any candidate decrease with s, for any $0 < \eta < 1/2$, as $O(s^{1-\eta\xi})$. See Rinnooy Kan and Timmer (1987) for the details.

In the interest of improving the effectiveness of the clustering method, we take into account the following facts that could be detrimental to the clustering method:

In Kaufman and Rousseeuw (1990, Ch.5), various techniques of agglomerative clustering are examined. The authors alert about the chaining effect of single linkage, which makes the clusters stick to each other because of the formation of chains, and argue, based on theoretical analysis and practice, that some other techniques such as *complete linkage* or *average linkage* would be better suited for most problems. We have incorporated this insight into our numerical experiments, despite the fact that the previous theoretical analysis for adjusting the radius r_s at each iteration does not carry over to the case where we consider the maximum distance to the cluster (complete linkage) or the average distance to the cluster (average linkage). In both cases the existence of a point within radius r with a lower function value does not ensure that a local minimization will not be started, unless there is a relationship between r and the (unknown) diameter of the cluster.

In Beyer et al. (1998), it had been pointed out that, for points sampled from a broad set of distributions, the distance of any of this points to its nearest neighbor becomes very close to the distance to the farthest point as the dimension increases. This essentially means that the notion of nearest neighbor loses much of its meaning in high dimension. Later on, in Aggarwal et al. (2001), the role played in this phenomenon by the norm used to measure distances was devealed. It was shown that, in expectation as $p \to \infty$, the gap goes to 0 for the ∞-norm, tends to a constant for the Euclidean norm, and goes to ∞ for the 1-norm.

In our numerical tests in Section 6 we compare the actual impact of these two factors on the performance of clustering.

4. A Stopping condition

A crucial issue in global optimization algorithms is the tradeoff between solution accuracy and computing time. In practice, this dilemma is the decision about when to stop.

When using random subsampling (Maronna et al., 2006, Sec 5.7.2), the minimization (P) over the whole $\mathbb{R}^p$ is reduced to a search over a finite set of candidates generated from subsamples. For the probability of picking one outlier-free subsample from a sample with a fraction ε of contamination to be greater than $1 - \delta$, we need to sample a number N of candidates such that:

$$N \geq \frac{|log(\delta)|}{|log(1-(1-\varepsilon)^p)|}. \tag{7}$$

Unfortunately, this approach has some drawbacks. First, the number N in (7) grows exponentially with p. For instance, if $\delta = 0.01$ and $\varepsilon = 0.25$, we should sample 1450 candidates for $p = 20$, 25786 for $p = 30$ and more than 80 millions for $p = 50$. Furthermore, it depends on the fraction ε of contamination, which is not known in advance.

The main disadvantage of this criterion is its rigidity; it is an *a priori* criterion. Information about actually sampled candidates is completely disregarded. Thus, the algorithm will continue in the same way after 10 local minimizations as if it has found 8 local minima or only 1.

Adaptive criteria try to estimate the fraction of the search region that has been actually explored, using the observed information about the structure of each particular problem. Then, for a given level of accuracy, the running time of the algorithm will depend on the problem complexity, expressed mainly through the number of local minima that are found given a number of local minimizations.

In the sequel, we will describe an approach to this problem that uses Bayes' theorem to incorporate information gathered during the optimization process in order to decide when to stop. It was introduced in Boender and Rinooy Kan (1983) and refined in Piccioni and Ramponi (1990). The framework is the following: a sequential sample is drawn from a multinomial distribution with an unknown number of cells and unknown cell probabilities. In our context, each cell will be associated with one minimum of problem (P) and will be filled with the subsamples whose candidates converge after a local minimization to the minimum associated with this cell. The cell probabilities will be the fraction of subsamples in the cell.

Let us consider a small illustrative example. In Figure 3, we elucidate a case with $n = 20$ and $p = 4$ where the objective function in (P) has 4 local minima $\beta_1^\star, \beta_2^\star, \beta_3^\star$ and $\beta_4^\star$. Seven subsamples have been drawn, thus yielding by least squares seven candidates $\beta_1, ..., \beta_7$. If local minimizations were started from these candidates, β_3 would converge towards $\beta_1^\star$. β_2 and β_4 would converge to $\beta_3^\star$. Candidates $\beta_1, \beta_5, \beta_6$ and β_7 would converge to $\beta_4^\star$, and the minimum $\beta_2^\star$

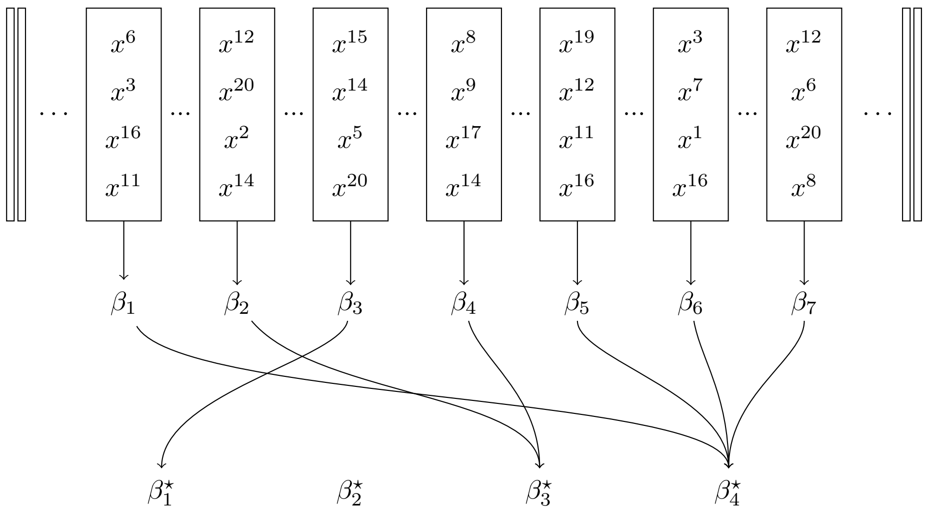
x^6 x^3 x^{16} x^{11}
x^{12} x^{20} x^2 x^{14}
x^{15} x^{14} x^5 x^{20}
x^8 x^9 x^{17} x^{14}
x^{19} x^{12} x^{11} x^{16}
x^3 x^7 x^1 x^{16}
x^{12} x^6 x^{20} x^8
β_1 β_2 β_3 β_4 β_5 β_6 β_7
$\beta_1^\star$ $\beta_2^\star$ $\beta_3^\star$ $\beta_4^\star$

Figure 3: An example of subsampling with $n = 20$ and $p = 4$. Seven subsamples have been sampled among the 4845 possible subsamples, coming from three of the four cells.

would remain undiscovered. Thus, we have observed 3 cells associated with the minima $\beta_1^\star, \beta_3^\star$ and $\beta_4^\star$, with observed frequencies of 1, 2 and 4.

In general, let $\beta_1^\star, \beta_2^\star, ..., \beta_k^\star$, be the local minima. Let ϑ_i, $i = 1, ..., k$, denote the probabilities of each cell. In the Bayesian approach, the unknowns $k, \vartheta_1, ..., \vartheta_k$ are supposed to be themselves random variables $K, \Theta_1, ..., \Theta_k$ with realizations $k, \vartheta_1, ..., \vartheta_k$ for which *a priori* distributions can be specified. Given the outcome $(m_1, ..., m_w)$ of a number of local searches, Bayes theorem is used to compute an *a posteriori* distribution. Following the approach of Piccioni and Ramponi (1990), we will suppose that different minima have different function values, $\hat{\sigma}(\beta_i^\star) \neq \hat{\sigma}(\beta_j^\star)$ for $i \neq j$. In such case, the minima can be ordered according to their function values $\hat{\sigma}(\beta_1^\star) < \hat{\sigma}(\beta_2^\star) < ... < \hat{\sigma}(\beta_k^\star)$. The same will be done with the minima found in experiments. Hence, we can compute the statistics $O_l = M_{J_l}, l = 1, ..., W_m$, where M_i denotes the observed frequency of the ith minimum ($M_1 = 1$, $M_2 = 0$, $M_3 = 2$ and $M_4 = 4$ in our example), J_l the index of the lth observed minimum (in the example $J_1 = 1$, $J_2 = 3$ and $J_3 = 4$) and W_m the number of (different) observed minima after sampling m candidates. In simple words, O_l is the frequency of the lth *observed* minimum.

A particularly interesting quantity from the stopping criteria viewpoint is $H = J_1 - 1$, which denotes the number of undiscovered local minima *with better function value than the observed minima.* Supposing *a priori* that the number of cells K follows an improper uniform discrete distribution on $[1, \infty)$, and that given $K = k$, the cell probabilities $\Theta_1, ..., \Theta_k$ are jointly uniformly distributed on the $k-1$ unit simplex, the following conditional probability for H can be obtained, when $m \geq 2$ and $w \leq m-2$ (Piccioni and Ramponi, 1990, Corollary 2):

$$\mathbb{P}(H = h | W_m = w, O_l = m_l, {}_{l=1,\ldots,w}) = (m - w - 1)\frac{(m-2)!(w+h-1)!}{(w-1)!(m+h-1)!}.$$

In particular, the probability that the global minimum has been already discovered is

$$\mathbb{P}_{m,w} := \mathbb{P}(H = 0 | W_m = w) = \frac{m - w - 1}{m - 1}, \qquad w \leq m - 2. \tag{8}$$

Using (8) we can readily devise a stopping condition, namely, to stop when the probability of having found the global minimum reaches a prespecified threshold.

The probability $\mathbb{P}_{m,w}$ is undefined when $w = m-1$ or when $w = m$. However, in practice, this only occurs during the first two or three iterations. In this case, because we do not have evidence that the search region has been well explored, we should keep the algorithm running.

For example, let us consider the robust regression problem (P) for the stackloss dataset (Maronna et al., 2006, pp. 381). The Multistart method consist of sampling candidates βs and starting a local minimization from each of them, until the probability (8) reaches a given threshold. Let us consider the thresholds 0.3, 0.6 and 0.9. The first threshold 0.3 is reached after 4 local minimizations, which give 2 different local minima, none of which is the global one. The threshold 0.6 is reached after 9 local minimizations and 3 minima, one of which is the global minimum. Finally, the threshold 0.9 is reached after 42 local minimizations, producing 4 local minima.

Of course, since it is a random algorithm, its running is unlikely to be the same every time. What should be retained is that higher thresholds give more accurate results in the sense that the search region is more exhaustively explored, and this is done adaptively. Needless to say, a more exhaustive search will take longer than a rougher one. In general, there is not an easy way to guess how long will it take to solve problem (P) within a given accuracy, but one can always impose a time limit, and the value (8) can be given as information to the user at the end.

We would like to stress the fact that this approach works for any algorithm based on subsampling and local searches, even if the objective is to compute other estimators, such as LTS, or beyond the context of linear regression, such as the location and scatter estimation problem.

Stopping conditions based on (8) can also be used for algorithms described in Section 3 that try to foresee the result of a local minimization, and to avoid it if it is likely to re-discover an existing minimum. In that case, in (8) m will be the number of sampled candidates and not the number of local searches actually carried out. The reason is that clustering methods are supposed to give the same outcome one could have obtained by starting local searches from each candidate. Nevertheless, the precision of the stopping condition will be subordinated to that of the clustering method; if it alters the outcome of the algorithm, the current *a posteriori* probability will be updated with wrong information.

5. A clustering global optimization point of view of robust regression estimation

We have already mentioned that most existing methods for computing robust regression estimators can be described as clustering methods, as described in Section 3. In this section we shall describe them, and we will see how they perform each of the steps discussed in Section 3.

5.1. Random Resampling

The original version of random resampling was introduced by Rousseeuw in Rousseeuw (1984) for computing the least median of squares estimator, and it was further refined and adapted for S-estimators by Ruppert in Ruppert (1992). Rousseeuw's original version introduced the sampling technique used by most existing algorithms, and his algorithm consisted only of sampling and choosing the candidate with the least scale. The modification of Ruppert, illustrated in Figure 4, included selection and local minimization applied, without clustering, to the best candidates. The details of each step are described below.

Given parameters N, t, do once:

- Sampling: Use Rousseeuw's random subsampling.
- Concentration-Selection: No concentration step is performed, but a selection of the t best candidates is done.
- Clustering: No clustering is performed. A local minimization is started from each candidate until convergence.
- Stopping condition: There is no stopping criterion. The number of sampled candidates is fixed in advance.

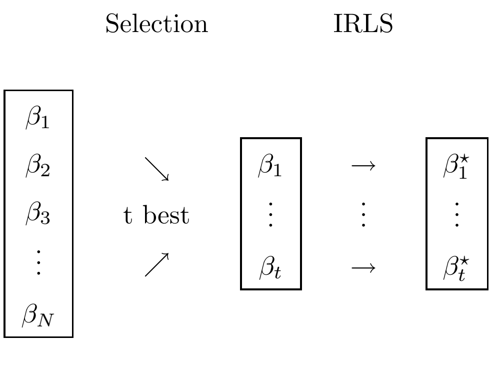

Figure 4: Schematic representation of Ruppert's version of random resampling.

5.2. Fast-S, Fast-τ

The "Fast-S" algorithm was presented in Salibian-Barrera and Yohai (2006) for S-estimators, and it was modified to cope with τ-estimators in Salibian-Barrera et al. (2008). This modification was called "Fast-τ". From the global optimization viewpoint, it extended Random Resampling by adding a concentration step. As for random resampling, we give below a schematic illustration and a detailed description of each step. Given parameters N, k and t, iterate the following steps:

- Sampling: Use Rousseeuw's random resampling.
- Concentration: Apply k steps of IRLS to each candidate. Select the t candidates with best objective function.
- Clustering: No clustering is performed.
- Stopping condition: There is no stopping condition; the number of sampled candidates is fixed in advance.

$$\begin{tikzcd}
\beta_1 \arrow[r, "\text{IRLS}"] & \beta_1^k & & & \\
\beta_2 \arrow[r] & \beta_2^k & \searrow & \beta_1^k \arrow[r, "\text{IRLS}"] & \beta_1^\star \\
\beta_3 \arrow[r] & \beta_3^k & \text{t best} & \vdots & \vdots \\
\vdots & \vdots & \nearrow & \beta_t^k \arrow[r] & \beta_t^\star \\
\beta_N \arrow[r] & \beta_N^k & & &
\end{tikzcd}$$

Selection

Figure 5: Schematic representation of the Fast-S and Fast-τ methods.

5.3. Fast-τ with stopping condition.

As previously observed, the considered algorithms for computing robust regression estimators are all particular instances of the general clustering procedure depicted in Figure 2.

The first one, random resampling, consists only of sampling and selection, Fast-τ adds a concentration step. Unlike clustering, the addition of an adequate stopping condition does not add computational work. For this reason, we introduce here a modification of Fast-τ for including stopping criteria, without doing clustering. It simply executes Fast-τ iteratively, and at the end of each iteration, it adds the (eventually new) minima found on the list of minima encountered in previous iterations, and evaluates a stopping condition. In our case, we require the probability $\mathbb{P}_{sN,w}$ defined by (8) to exceed a given threshold θ.

Our proposition is summarized in Figure 6. For given parameters N, k and t, and a probability of success θ,

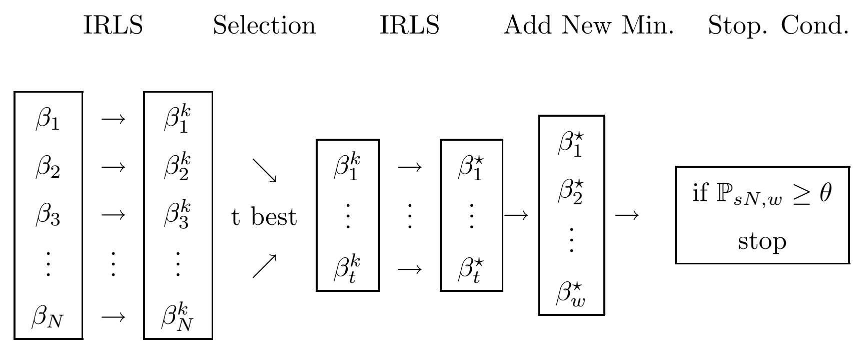
IRLS
Selection
IRLS
Add New Min.
Stop. Cond.
β_1 β_2 β_3 $\vdots$ β_N
β_1^k β_2^k β_3^k $\vdots$ β_N^k
t best
β_1^k $\vdots$ β_t^k
$\beta_1^\star$ $\vdots$ $\beta_t^\star$
$\beta_1^\star$ $\beta_2^\star$ $\vdots$ $\beta_w^\star$
if $\mathbb{P}_{sN,w} \geq \theta$
stop

Figure 6: The proposed modification of Fast-τ including a stopping condition.

Note that under this form, the modified version of Fast-τ does not exactly fit in the framework illustrated in Fig 2 because the selection is performed considering only the candidates sampled in the last iteration and disregarding the ones from previous iterations.

6. Numerical Tests

We conducted numerical experiments in order to investigate the impact of clustering techniques and stopping criteria. As in Ruppert (1992) and Salibian-Barrera et al. (2008), we consider simulated data contaminated with clustered outliers, where contamination is highly concentrated around outlying values.

- $(1-\delta)100\%$ of the points follow the regression model $y = X\beta + \varepsilon$, where the $p-1$ covariates are distributed as $N_{p-1}(0, I_{p-1})$, $x_{ip} = 1$ is the intercept, $\varepsilon_i \sim N(0,1)$ and $\beta = 0$.

- $\delta 100\%$ of the points are bad high-leverage points: the covariates now follow a $N_{p-1}(d, 0.1^2 I_{p-1})$ distribution where $d = (100, 0, ..., 0)^t$, and the response variables $y_i \sim N(100m, 0.1^2)$. In the following we will call the parameter m the "contamination slope".

For evaluating the stopping condition, we tested only Multistart, and we compare it with Fast-τ using $N = 50$ samples. For performance evaluations, we tested all the algorithms described in Section 5, except for Random Resampling (Subsection 5.1). The reason for excluding Random Resampling from those tests is that we included the Fast-τ algorithm, and the numerical tests reported in Salibian-Barrera et al. (2008) already compared Fast-τ with Random Resampling, among others, and showed that it outperforms Random Resampling.

The clustering methods were implemented by the author in Matlab and are available upon request. In our test, it was always executed with the same parameters: at each iteration, $N = 100$ candidates are sampled and added to the sample, then a concentration step consisting of one IRLS iteration (cf. Section 2) and a selection of the best 10% of the concentrated candidates is performed (see Figure 2). Concerning the radius used for the clustering, as the minimization in (P) is done over the unbounded domain $\mathbb{R}^p$, we used the formula $r_s = \pi^{-1/2} \left(\Gamma(1 + p/2)\xi \ln(sN)/(sN)\right)^{1/p}$, for a predefined $\xi > 0$. After extensive experiments, we realized that the results are rather insensitive to the choice of the parameter ξ. We set $\xi = 40$ in our tests. For the tests involving the Fast-τ algorithm, we used the code available from the webpage of Matias Salibian-Barrera. We used the parameters $k = 2$ and $t = 5$ (see Figure 5). The number N of initial candidates changed from test to test and is indicated each time.

Both Fast-τ and Single Linkage clustering were fitted with a stopping condition. In all cases, it consisted in stopping when the probability $\mathbb{P}_{m,w}$ defined in (8) reached a given threshold θ.

Concerning the performance in terms of computing time, for each algorithm we saved the required time T_A. Then, we computed the ratio $RT = T_A/T_R$, where T_R is the time required by an algorithm used as reference. We usually used Fast-τ as reference, with different parameters depending on the test. This ratio is what we call the relative computing time. In this way, the results are machine-independent; and easier to compare.

As it is not feasible to know certainly if the returned solution is the global minimum, we only give the relative τ-scale with respect to a reference algorithm, $\hat{\sigma}_A/\hat{\sigma}_R$, where $\hat{\sigma}_A$ and $\hat{\sigma}_R$ are the values of the objective (3). We call this ratio "relative τ-scale". Similarly to the results reported in Salibian-Barrera and Yohai (2006) for S-estimators, for the particular type of contamination that we are considering, often the coefficient $\hat{\beta}_1$ over the 500 samples forms two well-separated groups: one highly concentrated around the contamination slope, and another one more dispersed around 0, the slope of the data without contamination, hereafter referred to as "clean slope". In those cases, we also show the percentage of samples for which the "slope" is around 0. Note that it is not uncommon that minima with the contamination slope had better function values (3) than minima around the "clean" slope, especially for those with high proportions of outliers.

In Subsection 6.1 we evaluate the impact of the stopping condition. We discuss in detail how it behaves when applied to Multistart with threshold values ranging in a wide range. Our second and third test, presented in Subsection 6.2 and 6.3 , evaluate the performance of the considered algorithms in two different kinds of situations. In 6.2, this is done for a small problem, whose complexity varied only through the change in the dimension p. In the third test, presented in Subsection 6.3, we compare the performance of the considered algorithms, the parameter used to control the complexity of the problems was the contamination slope m.

6.1. Analysis of the stopping condition

The objective of this Subsection is to scrutinize the impact of the stopping condition presented in Section 4 on the algorithms in Section 5. The datasets were generated as indicated at the beginning of Section 6, with $n = 400$ observations in dimension $p = 15$. The contamination fraction was $\delta = 40\%$ forming two groups of 20% each with contamination slopes $m = 2$ and $m = 4$. The objective of using two groups of outliers was to increase the number of local minima, which would better illustrate the effectiveness of the stopping condition, since the complexity of the problems heavily depends on the number of local minima. The algorithms compared were Multistart (MS), which consist of sampling candidates and launching a local minimization from each of them; it is the same as the Random Resampling algorithm (cf. Section 5.1) without selection. and Fast-τ with $N = 50$ samples (FT50). Multistart stopped as soon as the probability $\mathbf{P}_{m,w}$ in (8) reached a given threshold θ. In this section, we used the thresholds $0.3, 0.6, 0.9$ and 0.95.

In Table 1 we show the following:

- The number of local minima, w.
- The quantity of sampled candidates, m.
- The effective value of the probability (8) at the termination of the algorithm, $\mathbf{P}_{m,w}$.
- The percentage of samples for which the estimate given by the algorithm had the clean slope, around 0, at the row "% Slope ".
- The relative running time with respect to FT50, T_{MS}/T_{FT50}.
- The relative τ-scale with respect to FT50, $\hat{\sigma}_{MS}/\hat{\sigma}_{FT50}$.

All the entries, except for the clean slope, are the average over 500 simulations.

By examinating the column corresponding to FT50 in Table 1 we see that, in this example with few local minima, most of the time the effective value of (8) is already around 0.96; if the function used to have 5 local minima, it would be around 0.9 and it would be around 0.8 if the function had 10 local minima. The reason is that candidates are sampled in batches; thus, if each batch is of size N, the only attainable threshold values are of the form $(N - w - 1)/(N - 1)$, for

Table 1: Details of the execution of Multistart (MS) and Fast-τ with 50 samples (FT50). The datasets consisted of $n = 400$ observations in dimension $p = 15$, with a fraction of contamination of $\delta = 40\%$.

	MS				FT50
θ	0.3	0.6	0.9	0.95	
w	1.2	1.2	1.7	2.4	2.2
m	3.19	4.69	19.08	50.24	50
$\mathbf{P}_{m,w}$	0.47	0.66	0.9	0.95	0.96
% Slope	95.4	94.6	86.4	77.2	68.6
T_{MS}/T_{FT50}	0.17	0.24	0.96	2.52	1
$\hat{\sigma}_{MS}/\hat{\sigma}_{FT50}$	1.012	1.0116	1.0083	1.0045	1

positive integers w. Since in MS, candidates are sampled one by one, it stops as soon as the threshold is reached so it can be combined with low threshold values. On the contrary, algorithms that do some kind of selection need to sample in batches; for performing local searches, only from promising candidates. In the case of algorithms performing clustering, a harsh selection is needed in order to well separate clusters and prevent sticking to each other, thus the use of small batches is discouraged. As a consequence, for all of the algorithms of Section 5, if the objective function does not have many local minima, only relatively high threshold values will be observed in practice.

However, MS illustrates quite well how the stopping condition works, since we see how the number of discovered local minima and the number of sampled candidates increases as the threshold increases. Even if the percentage of datasets for which the estimate had the clean slope seems to indicate the contrary, the accuracy of the result also improved. This can be seen by examining the relative τ-scale. We see that MS gives worse results than FT50 for low threshold values, but this ratio decreases for higher threshold values, showing the better performance of MS.

6.2. Many dimensions, many thresholds on a small problem

In our second test, we set the contamination fraction to $\delta = 0.2$ and the contamination slope to $m = 2$. For each $p \in \{5, 10, 15, 20\}$, we generated 500 datasets of $n = 100$ points. We compare four algorithms:

- Fast-τ with $N = 500$ candidates (FT500).
- Fast-τ with $N = 250$ candidates (FT250).
- Fast-τ with a stopping condition (SC), described in subsection 5.3. The parameters are (see Figure 6) $N = 100$, $k = 1$ and $t = 5$.
- Single Linkage clustering as described in Subsection 3.3 (SL), with the parameters indicated at the beginning of this Section. Motivated by the observation at the end of Section 4, we tested Single Linkage using the l_1 and the Euclidean norm. They are denoted as SL1 and SL2, respectively.

For those algorithms incorporating a stopping condition (SC and SL), the threshold parameter θ took the values 0.95, 0.97 and 0.99. These results are shown in Table 2 for the algorithms with stopping condition, and in table 3 for Fast-τ with $N = 250$ and $N = 1500$ samples.

For algorithms with stopping condition, a curious situation appears, as the algorithms performing clustering always perform worse than those without clustering, but in low dimension, they find quite often a better solution than the optimal one, in the sense that they find a non-global minimum with the clean slope. As the dimension increases, however, their performance degrades both in terms of objective function and in the percentage of times that they find the clean slope. Single Linkage using the l_1 norm generally had better function values than using the Euclidean norm, but the difference is negligible. In preliminary tests we also tried Complete Linkage and Average Linkage clustering, but the results were essentially identical, so we only used Single Linkage.

Overall, the Fast-τ algorithm with stopping condition using thresholds 0.95 and 0.97 achieves a good tradeoff between computing time and quality of the

Table 2: Quality of the solution and computing effort for SC, SL1 and SL2. The parameters of the contamination were $m = 2$ and $\delta = 0.2$. The relative time and the relative τ-scale are with respect to FT500

		% Clean Slope			Relative τ-scale			Relative Time		
p		95%	97%	99%	95%	97%	99%	95%	97%	99%
	SC	58	58	58.2	1	1	1	0.34	0.35	0.74
5	SL1	88	88	88.2	1.019	1.019	1.02	0.15	0.16	0.49
	SL2	88.4	88.4	88.4	1.021	1.021	1.021	0.1	0.1	0.29
	SC	64.4	64.4	64.6	1	1	1	0.466	0.51	1.44
10	SL1	78.8	78.8	85.8	1.025	1.025	1.020	0.21	0.23	0.63
	SL2	78	78	85	1.024	1.024	1.021	0.12	0.13	0.33
	SC	68.8	69.2	69.8	1.003	1.002	0.999	0.48	0.64	2.45
15	SL1	32.8	33	52.4	1.104	1.103	1.069	0.23	0.24	0.65
	SL2	36.4	36.4	53.8	1.090	1.090	1.064	0.14	0.15	0.4
	SC	64.8	64.8	65.2	1.016	1.014	0.998	0.51	0.72	3.46
20	SL1	7.6	7.8	19.8	1.193	1.192	1.149	0.25	0.26	0.62
	SL2	8.4	8.4	14.8	1.206	1.206	1.192	0.18	0.18	0.40

solution, as it gives results almost as good as FT500 or even FT1500, but within a fraction of time. By raising the threshold to 0.99, the results are similar to those of FT1500, but with a larger computing time.

6.3. Varying complexity for a fixed dimension

In the next test, we examine the behavior of the algorithms for a fixed dimension, but also for various proportions of outliers and contamination slopes. The closer the contamination slope is to the clean slope, the more difficult it becomes to identify the clean one.

We tested FT250, FT500, FT1500, SL and SC in dimension $p = 10$ with different contamination slopes. The number of observations was fixed to $n = 400$, and the proportion of outliers was $\delta = 10\%, 15\%$ and 20%. For SL and

Table 3: Quality of the solution and computing effort for FT250 and FT1500. The parameters of the contamination were $m = 2$ and $\delta = 0.2$. The relative time and the relative τ-scale are with respect to FT500

p		% Clean Slope	Relative τ-scale	Relative Time
	FT250	58.2	1.000	0.57
5	FT1500	58.4	1.000	2.8
	FT250	64.8	1.000	0.62
10	FT1500	64.6	0.999	2.63
	FT250	71.4	1.001	0.64
15	FT1500	72.2	0.998	2.57
	FT250	65.4	1.007	0.65
20	FT1500	69	0.996	2.6

SC, the stopping condition uses the threshold $\theta = 0.95$; because the preliminary test did not show significative improvements when we raised the threshold in these problems. Similarly to the previous test, the use of different variants of clustering did not significantly change the results; therefore we only tested Single Linkage clustering, with the usual Euclidean norm.

The performance was measured as in the previous test. Namely, we compare the objective function value with respect to a reference algorithm, and we keep the percentage of the samples from which convergence occurred to a minimum with a slope around 0. These two quantities were the measure of quality of the solution. We also record the time relative to the time spent by FT500, as explained in the previous section.

The quality of the solutions obtained by Fast-τ was identical to SC; thus, we do not include it in the tables. The results of these test are in Table 4.

As in the previous test, the algorithm with clustering has a very good running time, about 20% of the time spent by Fast-τ, and in some cases, it gives a

Table 4: Percentage of samples where the minimum found had the clean slope, and time spent relative to FT500. For $p = 10$.

δ	10%				15%				20%			
Slope	1.1	1.4	1.7	2	1.1	1.4	1.7	2	1.1	1.4	1.7	2
	% Clean Slope											
SC	62	100	100	100	0	26.4	98	100	0	0	3.2	61.8
SL	99	100	100	100	44.2	94.6	100	100	2.8	21.4	77.4	96.4
	Relative Time											
SC	0.34	0.33	0.34	0.35	0.37	0.38	0.40	0.39	0.39	0.41	0.44	0.39
SL	0.18	0.21	0.20	0.20	0.22	0.17	0.20	0.21	0.24	0.21	0.19	0.14
FT250	0.57	0.59	0.59	0.60	0.60	0.59	0.59	0.61	0.61	0.62	0.61	0.57
FT1500	2.71	2.70	2.69	2.68	2.58	2.69	2.62	2.57	2.57	2.56	2.60	2.78
	Relative τ-scale											
SL	1.01	1	1	1	1.07	1.04	1	1	1.01	1.04	1.07	1.01

worse solution to the minimization problem (P) than algorithms without clustering. Thus, it does not compute the τ-estimator, but the result it gives has the clean slope. We do not have a clear explanation for that, and we think this phenomenon bears closer examination; in another article. We see once again that the stopping condition permits to find the global minimum with a high probability in a proper computing time.

7. Conclusions and Future Work

We have investigated the effectiveness of the usage of clustering techniques and stopping conditions for global optimization in the particular case of robust regression. Our viewpoint is that of an user of robust regression who wants to compute robust estimators without having to adjust parameters that depend upon the details of the chosen algorithm.

The integration of a stopping condition is completely justified both by the quality of the results and by the performance in terms of computing time. Additionally, it is very simple to implement in new and existing software. It should be incorporated in algorithms not only for computing τ-estimators, but also in any algorithm based on subsampling and concentrations steps.

A threshold between 0.95 and 0.97 achieves a good compromise between efficiency and quality of the results. A higher threshold value ensures a very good solution; at an extra cost in terms of computing time. For routine utilization, a threshold of 0.96 or 0.97 should give good results at a competitive computing time, which will adapt by itself to the complexity of the problem. Additionally, it possesses a very appealing interpretation in terms of probability of finding the global optimum, independently of the particular problem under consideration.

In their present states, the existing clustering techniques for global optimization do not seem to fit the needs of robust estimation problems. However, as shown in Section 5, they are a natural extension of the existing methods, and their computing times and behavior in some difficult problems suggest that they deserve further investigation.

Acknowledgments

The author wishes to thank Anne Ruiz-Gazen for her advice and many suggestions, and Marcel Mongeau for his careful reading of the previous versions of this paper. The author is also grateful to the Editor, the Associated Editor and the referees for their work and their comments and suggestions on the original manuscript.

References

Aggarwal, CH., Hinneburg, A. and Keim, D., 2001. On the surprising behaviour of distance metrics in high dimensional space. In: Van den Bussche, J. and Vianu, V. (eds.), ICDT 2001, LNCS 1973, Springer-Verlag, Berlin Heidelberg, 420-434.

Archetti, F. and Schoen, F., 1984. A survey on the global optimization problem: general theory and computational approaches. Annals of Operations Research, 1(2) 87-110.

Agulló, J., 2001. New algorithms for computing the least trimmed squares regression estimator. Comput. Statis. Data Anal. 36(4), 425-439.

Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U., 1998 When is the nearest neighbor meaningful? In: Beeri, C., Buneman, P. (eds.), ICDT 1999, LNCS 1540, Springer-Verlag, Berlin Heidelberg, 217-235.

Björck, Å., 1996. Numerical methods for least squares problems, SIAM, Philadelphia.

Boender, E. and Rinnooy Kan, A.H., 1983. A Bayesian analysis of the number of cells of a multinomial distribution. The Statistician, 32(1/2) 240-248.

Donoho, D.L. and Huber, P., 1983. The notion of breakdown point. In: Bickel, P.J., Doksum, K. A. and Hodges, J. L., Jr., (eds.), A Festschrift for Erich L. Lehmann, Wadsworth Statist./Probab. Ser., Wadsworth, CA., 157-184.

Huber, P., 1981. Robust Statistics, Wiley, New York.

Kaufman, L. and Rousseeuw, P., 1990. Finding groups in data: an introduction to cluster analysis. Wiley, New York.

Maronna, R.A., Martin, R. D. and Yohai, V.J., 2006. Robust Statistics. Theory and methods. Wiley Series in Probability and Statistics.

Nocedal, J. and Wright, S., 1999. Numerical Optimization, Springer-Verlag, New York.

Piccioni, M. and Ramponi, A., 1990. Stopping rules for the multistart method when different local minima have different function values. Optimization, 21(5) 697-707.

Rousseeuw, P., 1984. Least median of squares regression. Journal of the American Statistical Association, 79(388) 871-880.

Rousseeuw, P. and Van Driessen, K., 2000. An algorithm for positive-breakdown regression based on concentration steps. In: Gaul, W., Opitz, O. and Schader, M. (eds.), Data Analysis: Modeling and Practical Applications, Springer, New York, 335-346..

Rinnooy Kan, A.H. and Timmer, G.T., 1987. Stochastic global optimization methods; part I: Clustering methods. Math. Prog., 39(1) 27-56.

Ruppert, D., 1992. Computing S-estimators for regression and multivariate location/dispersion. Journal of Computational and Graphical Statistics, 1(3) 253-270.

Salibian-Barrera, M., Willems, G. and Zamar, R. H., 2008. The fast-τ estimator for regression. Journal of Computational & Graphical Statistics, 17(3) 659-682.

Salibian-Barrera, M. and Yohai, V., 2006. A fast algorithm for S-regression estimates. Journal of Computational & Graphical Statistics, 15(2) 414-427.

Törn, A. and Žilinskas, A., 1989. Global Optimization, Lecture Notes in Comput. Sci. 350, Springer-Verlag, New York.

Yohai, V. and Zamar, R.H., 1988. High breakdown-point estimates of regression by means of the minimization of an efficient scale. Journal of the American Statistical Association, 83(402) 406-413.