Computation of τ-estimates for regression

Matias Salibian-Barrera, Gert Willems and Ruben Zamar

Department of Statistics, University of British Columbia, 333-6356 Agricultural Road, Vancouver, BC V6T 1Z2, Canada

Abstract

Yohai and Zamar's (1988) τ-estimates of regression have excellent statistical properties but are nevertheless rarely used in practice due to their lack of software implementation and the general impression that τ-estimates are difficult to compute or approximate. We will show, however, that the computational difficulties we must overcome to approximate τ-estimates are similar in nature to those of the more popular S- and MM-estimators. The goal of this paper is twofold: (i) to implement a computing algorithm to approximate τ-estimates based on random resampling, and (ii) to compare this implementation with some alternative heuristic search algorithms. We show that the random resampling algorithm is not only simpler, but generally outperforms the considered heuristic search algorithms when enhanced by local improvement steps.

Keywords: Robust regression; τ-estimators; random resampling; simulated annealing; tabu search.

1 Introduction

Consider the usual linear regression model

$$y_i = \boldsymbol{\beta}_0^t \mathbf{x}_i + \epsilon_i \,, \quad i = 1, \ldots, n \,, \tag{1}$$

for which we want to estimate the vector of regression coefficients $\boldsymbol{\beta}_0 \in I\!R^p$ based on n independent observations $(y_i, \mathbf{x}_i) \in I\!R \times I\!R^p$. The error terms ϵ_i are assumed to be independent from the covariates $\mathbf{x}_i$ and identically distributed with zero center and unknown scale σ_0.

The least squares (LS) estimate for $\boldsymbol{\beta}_0$ is very easy to compute and optimal under the assumption of normally distributed errors. However, it is well known that it can be heavily affected by a very small proportion of outliers in the data. Many robust regression

estimators that are capable of resisting the negative effect of outliers have been proposed in the literature. Several of those robust estimators are widely available as they are implemented in various statistical software packages (see for example the recent `robustbase` package for `R` (R Development Core Team, 2005) available on-line at the `CRAN` website `http://cran.r-project.org`), and the `robust` library for `S-PLUS`. Among the most popular robust regression estimators are the least trimmed squares (LTS) (Rousseeuw 1984), S-estimators (Rousseeuw and Yohai 1984) and MM-estimators (Yohai 1987). These are all called high-breakdown estimators since they can be tuned to resist contamination in up to 50% of the observations.

Unfortunately, another common feature of these estimators is the time-consuming nature of their computation. In fact, exact computation is usually infeasible except for small data sets. Considerable effort has been made, however, to construct approximate algorithms that perform reasonably well. We can mention the fast-LTS (Rousseeuw and Van Driessen 1999) and the fast-S (Salibian-Barrera and Yohai 2006) algorithms. The latter is instrumental in the computation of MM-estimators as well.

Of the above, only the MM-estimators combine high breakdown point with high efficiency, which makes them a good option for inference purposes. However, τ-estimators (Yohai and Zamar 1988) also combine good robustness and high efficiency. Moreover, they have two theoretical advantages over MM-estimators: they have lower maximum bias curves, and they are associated with a robust and efficient scale estimate. Unfortunately, the computation of τ-estimators has been somewhat neglected in the literature. The τ-estimator is defined as

$$\widehat{\boldsymbol{\beta}}_n = \underset{\beta \in \boldsymbol{R}^p}{\operatorname{argmin}} \, \tau_n(\boldsymbol{\beta}) \,, \tag{2}$$

where the τ-scale $\tau_n(\boldsymbol{\beta})$ is given by

$$\tau_n^2(\boldsymbol{\beta}) = s_n^2(\boldsymbol{\beta}) \frac{1}{n\, b_2} \sum_{i=1}^{n} \rho_2 \left(\frac{y_i - \boldsymbol{\beta}^t \mathbf{x}_i}{s_n(\boldsymbol{\beta})} \right) , \tag{3}$$

with $s_n(\boldsymbol{\beta})$ an M-estimator of scale that solves

$$\frac{1}{n} \sum_{i=1}^{n} \rho_1 \left(\frac{y_i - \boldsymbol{\beta}^t \mathbf{x}_i}{s_n(\boldsymbol{\beta})} \right) = b_1 \,. \tag{4}$$

The functions $\rho_j; j = 1, 2$ are assumed to be symmetric, continuously differentiable, bounded, strictly increasing on $[0, c_j]$ and constant on $[c_j, \infty)$, with $0 < c_j < +\infty$, $j = 1, 2$. The parameters b_1 and b_2 are tuned to obtain consistency for the scale at the normal error model:

$$b_j = E_\Phi[\rho_j(u)]; \quad j = 1, 2\,, \tag{5}$$

where Φ is the standard normal distribution. In the special case where $\rho_1 = \rho_2$, we have $\tau_n(\boldsymbol{\beta}) = s_n(\boldsymbol{\beta})$ so that (2) reduces to the definition of an S-estimator.

Similarly to MM-estimators, τ-estimators have the breakdown point of an S-estimator based on the loss function ρ_1, while its efficiency is determined by the function ρ_2 used in (3) (see Yohai and Zamar, 1988).

The choice of the loss functions ρ_1 and ρ_2 can be of considerable practical and theoretical importance. In our examples and simulations we use the so-called Optimal weight functions introduced in Yohai and Zamar (1998). This family of loss functions confers excellent robustness properties on estimators such as τ-, S- and MM-estimators (and is therefore preferred over the commonly used Tukey bisquare family). The Optimal ρ-function with tuning parameter c is given by

$$\rho(t) = \begin{cases} 1.38\left(\frac{t}{c}\right)^2 & |\frac{t}{c}| \le \frac{2}{3} \\ .55 - 2.69\left(\frac{t}{c}\right)^2 + 10.76\left(\frac{t}{c}\right)^4 - 11.66\left(\frac{t}{c}\right)^6 + 4.04\left(\frac{t}{c}\right)^8 & \frac{2}{3} < |\frac{t}{c}| \le 1 \\ 1, & |\frac{t}{c}| > 1 \end{cases}$$

The breakdown point of the τ-estimator $\widehat{\boldsymbol{\beta}}_n$ is given by $\epsilon^*(\widehat{\boldsymbol{\beta}}_n) = b_1/\sup_t \rho_1(t)$. To obtain consistency and $\epsilon^*(\widehat{\boldsymbol{\beta}}_n) = 50\%$, we chose the tuning parameters $c_1 = 1.214$ and $b_1 = 0.5$ for ρ_1 in (4). The choices $c_2 = 3.270$ and $b_2 = 0.128$ for ρ_2 yield a τ-estimator with 95% efficiency when the errors in (1) are normally distributed (see Yohai and Zamar, 1988). Moreover, the associated objective value $\tau_n(\widehat{\boldsymbol{\beta}}_n)$ then provides an estimator of the scale σ_0 which has a 50% breakdown point and a Gaussian efficiency of 97.7%.

It is interesting to note that the choice of the family of loss functions also affects the robustness properties of τ-estimators relative to those of MM-estimators. Berrendero et al. (2006) show that, when the ρ functions belong to Tukey's bisquare family, MM-estimators

have lower asymptotic bias than τ-estimators for proportions of contaminations ϵ lower than 0.20, whereas for $\epsilon > 0.20$ the situation reverses. The formulas of Berrendero et al. (2006) also show, however, that if ρ_1 and ρ_2 belong to the Optimal family above then the maximum asymptotic bias of MM- and τ-estimators become very close for $\epsilon < 0.20$ while for $\epsilon > 0.20$ τ-estimators still have notably lower maximum asymptotic bias. It should be noted that for both τ- and MM-estimators the Optimal family produces lower maximum bias curves than the bisquare family (for a given efficiency).

One of the main reasons why τ-estimators are not currently used much in practice is the little attention paid in the literature to the problem of their computation. The common perception seems to be that the form of the τ-objective function in (3) makes its minimization a difficult task, even by robust regression standards. However, although the function $\tau_n(\boldsymbol{\beta})$ is more involved than the S-objective function $s_n(\boldsymbol{\beta})$ in (4), both functions present very similar features. Namely: they are non-convex, have possibly multiple local minima, and evaluating them involves solving a non-linear equation, which makes their calculation time-consuming.

In this paper we will study the problem of efficient computation of τ regression estimators defined by (2). For S-estimators, the most commonly used algorithms are based on random resampling. This procedure was first proposed by Rousseeuw (1984) and later refined using iteratively reweighted least squares (IRWLS) by Ruppert (1992). The fast-S algorithm of Salibian-Barrera and Yohai (2006) is an improvement on Ruppert's algorithm. We will construct a similar algorithm for the case of τ-estimators, which in fact demonstrates that the computational problem of τ-estimators can be handled in the same way as that of S-estimators. Furthermore, we will compare this random resampling approach to Simulated Annealing and Tabu Search. The latter are heuristic search algorithms that were also investigated by Woodruff and Rocke (1993) in the context of robust estimation for multivariate models.

The algorithms are described and discussed in Section 2. Section 3 then presents results from a simulation study that compared the performance of these algorithms, while Section 4 contains our concluding remarks.

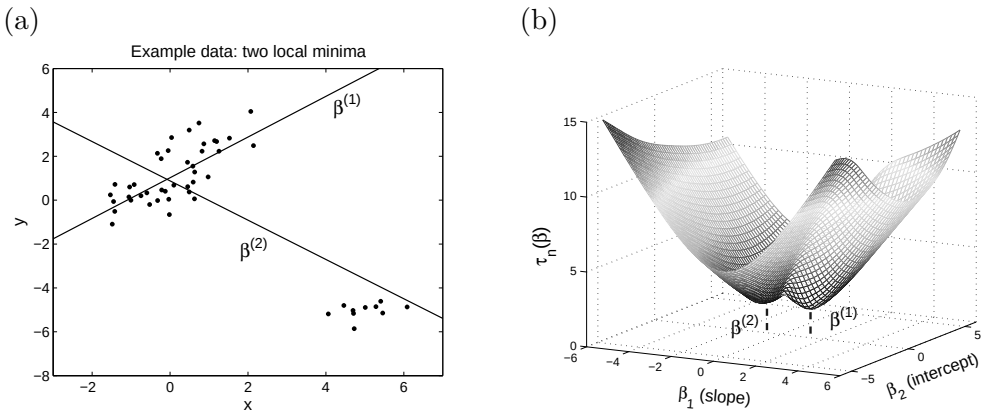

Figure 1: Example of a τ-objective function $\tau_n(\boldsymbol{\beta})$ for a simulated data set with $n = 50$ and $p = 2$ following the model $y = x + 1 + \epsilon$, with 20% of outliers; (a) scatterplot of (x, y); (b) τ-objective function indicating the two local minima $\boldsymbol{\beta}^{(1)}$ and $\boldsymbol{\beta}^{(2)}$.

2 Finding the global τ-minimum

To illustrate the difficulty of minimizing the τ-objective function (3) we generated a simple artificial data set with $n = 50$ observations following the model $y = \beta_1 x + \beta_2 + \epsilon$, where $\beta_1 = \beta_2 = 1$, $x \sim N(0, 1)$ and $\epsilon \sim N(0, 1)$. Then 20% of the points were shifted to generate the outliers as depicted in Figure 1(a).

Figure 1(b) shows the objective function $\tau_n(\boldsymbol{\beta})$ for this data, which reveals two local minima: $\boldsymbol{\beta}^{(1)}$ and $\boldsymbol{\beta}^{(2)}$, say. The regression lines corresponding to these minima are indicated in Figure 1(a) as well. It is clear that $\boldsymbol{\beta}^{(1)}$ can be regarded as the local minimum that is unaffected by the outliers, whereas $\boldsymbol{\beta}^{(2)}$ is induced by these outliers and should arguably be avoided. A satisfactory algorithm to solve (2) should return $\boldsymbol{\beta}^{(1)}$ as the τ-estimate, since $\tau_n(\boldsymbol{\beta}^{(1)}) = 1.653 < \tau_n(\boldsymbol{\beta}^{(2)}) = 1.927$. In the next sections we discuss different types of algorithms to optimize functions of this form.

2.1 Resampling algorithms

Figure 1 illustrates the main problem in optimizing the τ-objective function $\tau_n(\boldsymbol{\beta})$: we need to (a) identify the locally convex region ("valley") that corresponds to the global minimum, and (b) find the "bottom of this valley".

A simple approach to try to identify the different "valleys" of a non-convex function of this form is presented by the *random resampling* procedure (Rousseeuw 1984). This

procedure generates a large number of candidates $\boldsymbol{\beta}_{(j)}, j = 1, \ldots, N$, by drawing random subsamples of size $h \geq p$ from the data (without replacement) and letting $\boldsymbol{\beta}_{(j)}$ be the LS fit to the j-th subsample. When N is large enough, it is assumed that the candidate $\boldsymbol{\beta}$s constructed in this way will cover much of the parameter space, specifically the relevant "valleys" of the robust objective function. In particular, the hope is that at least one of the N random subsamples will be outlier-free, and that its corresponding LS fit will be close to the global minimum, so that we obtain a candidate in the right "valley".

In order to choose the number of resampling candidates, one could determine the value N that yields a sufficiently high probability of finding a subsample that does not contain outliers (see e.g. Salibian-Barrera and Yohai, 2006). Unfortunately, this number increases exponentially with the dimension p. Furthermore, it is easy to see that the probability that a random subsample of size h is outlier-free decreases with h. Therefore it is often advocated to work with $h = p$ ("elemental subsamples"). Note, however, that candidates produced using small values of h tend to be more unstable than those obtained with larger subsamples.

Random resampling without any local improvement is commonly referred to as *basic resampling*. It was implemented in the PROGRESS algorithm of Rousseeuw and Leroy (1987) and can be described as follows:

Basic resampling algorithm:

1. Draw N random subsamples, and obtain corresponding candidates $\boldsymbol{\beta}_{(j)}, j = 1, \ldots, N$;
2. evaluate $\tau_n(\boldsymbol{\beta}_{(j)})$, $j = 1, \ldots, N$;
3. let the estimate $\widehat{\boldsymbol{\beta}}_n$ be such that $\tau_n(\widehat{\boldsymbol{\beta}}_n) = \min_{1 \leq j \leq N} \tau_n(\boldsymbol{\beta}_{(j)})$.

Regarding the descent to the bottom of the "valley" (the local minimum), note that if a candidate found by the above algorithm is already sufficiently close to a local minimum, Newton-Raphson steps may be applied to quickly converge to the corresponding minimum. However, in our experience Newton-Raphson iterations can be hard to control in general (but see Arslan et al. (2002) for an application of Newton-Raphson steps in a similar context). An alternative method that works well in practice, although it has a slower rate of convergence, is to perform iteratively reweighted least squares (IRWLS) iterations.

The IRWLS step for τ-estimates is derived as follows. Yohai and Zamar (1988) show that any local minimum $\widehat{\boldsymbol{\beta}}_n$ of $\tau_n(\boldsymbol{\beta})$ satisfies the estimating equation

$$\sum_{i=1}^{n} \left[W_n(\widehat{\boldsymbol{\beta}}_n)\psi_1\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right) + \psi_2\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right)\right] \mathbf{x}_i = \mathbf{0}\,, \tag{6}$$

where

$$W_n(\widehat{\boldsymbol{\beta}}_n) = \frac{\sum_{i=1}^{n}\left[2\rho_2\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right) - \psi_2\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right) e_i(\widehat{\boldsymbol{\beta}}_n)\right]}{\sum_{i=1}^{n}\psi_1\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right) e_i(\widehat{\boldsymbol{\beta}}_n)}\,.$$

Here, $e_i(\widehat{\boldsymbol{\beta}}_n) = (y_i - \widehat{\boldsymbol{\beta}}_n^t \mathbf{x}_i)/s_n(\widehat{\boldsymbol{\beta}}_n)$, and ψ_1 and ψ_2 are the derivatives of ρ_1 and ρ_2 respectively. Equation (6) can be rewritten as

$$\widehat{\boldsymbol{\beta}}_n = \left(\sum_{i=1}^{n} \frac{\psi_n\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right)}{e_i(\widehat{\boldsymbol{\beta}}_n)}\,\mathbf{x}_i\mathbf{x}_i^t\right)^{-1} \sum_{i=1}^{n} \frac{\psi_n\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right)}{e_i(\widehat{\boldsymbol{\beta}}_n)}\,\mathbf{x}_i y_i\,, \tag{7}$$

where we denote $\psi_n\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right) := W_n(\widehat{\boldsymbol{\beta}}_n)\psi_1\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right) + \psi_2\left(e_i(\widehat{\boldsymbol{\beta}}_n)\right)$.

Note that (7) is the weighted LS fit to the data $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, with weights $w_i = \psi_n(e_i(\widehat{\boldsymbol{\beta}}_n))/e_i(\widehat{\boldsymbol{\beta}}_n)$. Suppose now that $\boldsymbol{\beta}^{(m)}$ is the current candidate, then the next IRWLS step yields $\boldsymbol{\beta}^{(m+1)}$ as follows:

$$\boldsymbol{\beta}^{(m+1)} := \left(\sum_{i=1}^{n} \frac{\psi_n\left(e_i(\boldsymbol{\beta}^{(m)})\right)}{e_i(\boldsymbol{\beta}^{(m)})}\,\mathbf{x}_i\mathbf{x}_i^t\right)^{-1} \sum_{i=1}^{n} \frac{\psi_n\left(e_i(\boldsymbol{\beta}^{(m)})\right)}{e_i(\boldsymbol{\beta}^{(m)})}\,\mathbf{x}_i y_i\,,\ m = 1, 2, \ldots \tag{8}$$

Iterations of this step lead to a solution of (7) and thus to a local minimum of $\tau_n(\boldsymbol{\beta})$. Extensive experiments have shown that these IRWLS steps for τ-estimates are very reliable in practice, in the sense that they converge to the local minimum by decreasing the objective function in every step.

Combining random resampling and IRWLS, we can construct an algorithm for τ-estimators that is similar to the fast-S and fast-LTS algorithms. First note that a simple and very effective algorithm is obtained by adding one step to the basic resampling algorithm as follows:

Resampling algorithm with exhaustive local improvements:

1. Draw N random subsamples, and obtain corresponding candidates $\boldsymbol{\beta}_{(j)}, j = 1, \ldots, N$;

2. for each $\boldsymbol{\beta}_{(j)}$, apply IRWLS until convergence, to obtain $\boldsymbol{\beta}^E_{(j)}; j = 1, \ldots, N$ respectively;

3. evaluate $\tau_n(\boldsymbol{\beta}^E_{(j)})$, $j = 1, \ldots, N$;

4. let the estimate $\widehat{\boldsymbol{\beta}}_n$ be such that $\tau_n(\widehat{\boldsymbol{\beta}}_n) = \min_{1 \le j \le N} \tau_n(\boldsymbol{\beta}^E_{(j)})$.

Hence, each randomly generated candidate is now locally improved before the best one is selected. Provided that N is large enough, the above algorithm will generally find the exact solution of (2), whereas basic resampling typically finds only an approximation.

The computation of the IRWLS steps is not trivial because it involves, for each resampling candidate $\boldsymbol{\beta}_{(j)}$, finding the M-scale $s_n(\boldsymbol{\beta}^{(m)}_{(j)})$ at each IRWLS step $m = 1, \ldots$, and this in turn requires an iterative algorithm. The number of IRWLS steps can be reduced, however, by observing that the largest reductions in objective value generally occur in the first few iterations. Therefore, just one or two IRWLS steps are often enough to determine which candidates will lead to the global minimum and which ones to another local minimum. Hence, for each random resampling candidate $\boldsymbol{\beta}_{(j)}$ we can perform $k = 1$ or $k = 2$ initial IRWLS iterations and retain the best t partially improved resampling candidates (t will generally be much smaller than N). We can then apply IRWLS until convergence to these t partially improved candidates only and report the one with the smallest final objective function as our estimate $\widehat{\boldsymbol{\beta}}_n$.

Finally, note that we can reduce the computation cost of this algorithm even further by replacing $s_n(\boldsymbol{\beta}^{(m)})$ in (8) by an approximate value. The latter can be obtained as the r-th step of an iterative algorithm to solve (4), where iterations are started from $s^{(0)} = \mathrm{mad}(r_i(\boldsymbol{\beta}^{(m)}))$ and at the l-th step, $s^{(l+1)}$ is given by

$$s^{(l+1)} = s^{(l)} \frac{1}{n\, b_1} \sum_{i=1}^{n} \rho_1 \left(\frac{y_i - \boldsymbol{\beta}^{(m)\,t} \mathbf{x}_i}{s^{(l)}} \right), \qquad l = 1, 2, \ldots$$

We will refer to the resulting local improvement steps (using $r = 1$) as approximate IRWLS steps. In our experience, the approximate steps perform nearly as well as the actual IRWLS steps, although convergence is slightly slower.

These considerations lead to the following version of the algorithm, which forms the core of what we call the fast-τ algorithm by analogy with the above-mentioned algorithms for S

and LTS.

Resampling algorithm with smart local improvements (fast-τ):

1. Draw N random subsamples, and obtain corresponding candidates $\boldsymbol{\beta}_{(j)}, j = 1, \ldots, N$;
2. for each $\boldsymbol{\beta}_{(j)}$, apply k approximate IRWLS steps, leading to $\boldsymbol{\beta}^I_{(j)}, j = 1, \ldots, N$ respectively;
3. for each $\boldsymbol{\beta}^I_{(j)}$, compute $\tau_j = \tau_n(\boldsymbol{\beta}^I_{(j)})$, $j = 1, \ldots, N$; let $\boldsymbol{\beta}^B_{(i)}$ be such that $\tau_n(\boldsymbol{\beta}^B_{(i)}) = \tau_{(i:N)}$, $i = 1, \ldots, t$, where $\tau_{(l:N)}$ denotes the l-th order statistic of the τ_js, $j = 1, \ldots, N$;
4. for each $\boldsymbol{\beta}^B_{(i)}$, apply IRWLS until convergence, leading to $\boldsymbol{\beta}^E_{(i)}; i = 1, \ldots, t$ respectively;
5. evaluate $\tau_n(\boldsymbol{\beta}^E_{(i)})$, $i = 1, \ldots, t$;
6. let the estimate $\widehat{\boldsymbol{\beta}}_n$ be such that $\tau_n(\widehat{\boldsymbol{\beta}}_n) = \min_{1 \le i \le t} \tau_n(\boldsymbol{\beta}^E_{(i)})$.

In our experience, the choice $k = 2$ (the number of IRWLS steps for each resampling candidate) and $t = 5$ (the number of candidates that are fully improved) works well in most situations. However, we will see in Section 3 that if the objective function has many local minima (as it often is the case when the ratio n/p is small) then higher values of k or t may be required to find the global minimum of $\tau_n(\boldsymbol{\beta})$ using this algorithm.

Remark Much of the computational effort in the fast-τ algorithm can be attributed to the objective function calculations in step 3. In the analogous fast-S algorithm, however, one can avoid having to compute the S-objective function for each candidate $\boldsymbol{\beta}$ by considering the following observation (Yohai and Zamar, 1991): given two candidates $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, in order to have $s_n(\boldsymbol{\beta}_2) < s_n(\boldsymbol{\beta}_1)$ it is necessary and sufficient that

$$\frac{1}{n} \sum_{i=1}^{n} \rho_1 \left(\frac{y_i - \boldsymbol{\beta}_2^t \mathbf{x}_i}{s_n(\boldsymbol{\beta}_1)} \right) < b_1. \tag{9}$$

It follows that a new candidate $\boldsymbol{\beta}$ can be discarded without computing its associated S-objective value if it violates (9). For τ-estimates now it can be shown that the following holds: given two candidates $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, in order to have $\tau_n(\boldsymbol{\beta}_2) < \tau_n(\boldsymbol{\beta}_1)$ it is necessary

(but not sufficient) that either condition (9) is fulfilled or that otherwise

$$\frac{1}{n}\sum_{i=1}^{n}\rho_2\left(\frac{y_i-\boldsymbol{\beta}_2^t\mathbf{x}_i}{s_n(\boldsymbol{\beta}_1)}\right) < \frac{1}{n}\sum_{i=1}^{n}\rho_2\left(\frac{y_i-\boldsymbol{\beta}_1^t\mathbf{x}_i}{s_n(\boldsymbol{\beta}_1)}\right). \tag{10}$$

Hence, $\boldsymbol{\beta}_2$ will not improve the τ-objective function corresponding to $\boldsymbol{\beta}_1$ if it violates both (9) and (10). Similarly to the fast-S algorithm, we can use these two conditions to discard several candidates without computing their τ-objective functions. However, since these conditions are not sufficient, the number of discarded candidates will not be as high as in the case of S-estimates. Numerical experiments show that this number heavily depends on the data.

2.2 Heuristic algorithms

It has been argued that random resampling as a technique to generate candidate $\boldsymbol{\beta}$s may be ineffective and that it can be outperformed by algorithms such as Simulated Annealing, Tabu Search or genetic algorithms. Woodruff and Rocke (1993) show that this is true for the robust estimation of multivariate location and scatter using the Minimum Volume Ellipsoid (MVE) estimator.

It is important to note that the algorithms compared by Woodruff and Rocke did not include local improvement steps. That is, the approximate minimizer of the objective function was chosen simply as the subsample-generated point with the smallest objective function. However, incorporating local improvements into the random resampling approach, as we do in the fast-τ algorithm, greatly enhances its performance. Therefore, it is of interest to investigate how these alternative heuristic algorithms compare with random resampling when local improvement is used (see also Maronna et al., 2006, page 198, for a fast-MVE algorithm). To investigate this question in the case of τ-regression estimators we compared the performance of the fast-τ algorithm described above with that of the Simulated Annealing (SA) and Tabu Search (TS) algorithms.

Both SA and TS use a randomly generated starting point and navigate through the parameter space by moving from the current point to a neighbouring one according to different criteria. Hence, we need to define a notion of neighbourhood appropriate to our application. While it is possible to define neighbourhoods by placing a continuous distri-

bution on the parameter space of $\boldsymbol{\beta}$ (see e.g. Sakata and White 2001), we prefer the more straightforward approach used in Woodruff and Rocke (1993). Since, as we mentioned in the previous section, points $\boldsymbol{\beta}$ that result from subsamples of the data tend to cover the more relevant "valleys" of the objective function, we will only consider $\boldsymbol{\beta}$s that are generated by subsamples. More specifically, at the k-th iteration of these algorithms, we say that $\boldsymbol{\beta}_{(k+1)}$ is a neighbour of the current point $\boldsymbol{\beta}_{(k)}$ if the corresponding subsamples differ in only one observation. Thus, if we use elemental subsamples, each point visited by these algorithms has $p(n-p)$ neighbours.

2.2.1 Simulated annealing

The following is a basic description of the SA algorithm as we implemented it. For a more general description and discussion, see e.g. Johnson et al. (1989) and Woodruff and Rocke (1993). SA's basic idea is to move through the parameter space by randomly choosing a neighbour of the current point that decreases the value of the objective function. However, to avoid being trapped in a local minimum, it may randomly allow a move that increases the objective function. Such an "uphill" move is accepted with probability $\exp(-\Delta/T)$, where Δ is the potential increase on the objective function that would result from this move, and $T > 0$ is a tuning constant called *temperature*. Note that when $T \to 0$ the probability of accepting a move that does not improve the objective function decreases rapidly. The algorithm starts with a relatively large value of T, which is then gradually reduced during the search. Eventually the algorithm *freezes* in a local minimum because no uphill moves can be accepted. Among all points visited by the algorithm the one with the smallest objective function is chosen as the approximate minimizer. In our setting, this translates into the following scheme.

Basic SA algorithm:

1. randomly select an initial subsample-generated $\boldsymbol{\beta}$;
2. set $T = T_0$ (initial temperature), and repeat until system is frozen:
   (a) repeat K times (K attempted moves in one "block"):
      i. randomly select a neighbour $\boldsymbol{\beta}'$ of $\boldsymbol{\beta}$ and compute $\Delta = \tau_n(\boldsymbol{\beta}') - \tau_n(\boldsymbol{\beta})$;

ii. if ($\Delta < 0$) set $\boldsymbol{\beta} = \boldsymbol{\beta}'$ (accept move);
else, with probability $\exp(-\Delta/T)$, set $\boldsymbol{\beta} = \boldsymbol{\beta}'$ (accept move);

(b) let $T = \alpha T$; $(0 < \alpha < 1)$ (lower temperature);

3. let $\widehat{\boldsymbol{\beta}}_n$ be such that $\tau_n(\widehat{\boldsymbol{\beta}}_n) = \min \tau_n(\boldsymbol{\beta})$ (minimum among $\boldsymbol{\beta}$s in the path).

The SA algorithm crucially depends on the tuning of parameters T_0, K and α, and to a lesser extent on the definition of a "frozen system". In these matters, we mainly followed Woodruff and Rocke in applying the guidelines of Johnson et al. (1989). See also Section 3.

2.2.2 Tabu search

Glover's (1986) general TS algorithm is deterministic in nature. Whereas SA randomly selects a neighbour of the current point to move to, TS selects the point that has the lowest objective value among all the neighbours. Once the search arrives at a local minimum in this way, the algorithm has to make an uphill move (the algorithm is forced to make a move even if every possible neighbour would yield an increase in objective value). In order to avoid that the algorithm would immediately move back to the local minimum in the next step, and thus become trapped, TS uses a *tabu list* based on the most recently visited points. Neighbours that appear in this list are rejected for the next move even if they would produce the best objective value. In our context, a neighbour of the current candidate $\boldsymbol{\beta}$ finds itself on the tabu list if the observation in which its corresponding subsample differs from that of $\boldsymbol{\beta}$, was involved in a recent previous swap (see also below). Often an *aspiration* criterion is used to allow for some exceptions to the tabu rule, although we will not consider this here. The following scheme describes the algorithm in its basic form.

Basic TS algorithm:

1. randomly select an initial subsample-generated $\boldsymbol{\beta}$;

2. repeat until a stopping criterion is met:

   (a) for every neighbour $\boldsymbol{\beta}'$ of $\boldsymbol{\beta}$ that is not on the tabu list, compute $\tau_n(\boldsymbol{\beta}')$

   (b) set $\boldsymbol{\beta} = \boldsymbol{\beta}'$, with $\boldsymbol{\beta}'$ the neighbour which minimized $\tau_n(\boldsymbol{\beta}')$.

   (c) update the tabu list

3. let $\widehat{\boldsymbol{\beta}}_n$ be such that $\tau_n(\widehat{\boldsymbol{\beta}}_n) = \min \tau_n(\boldsymbol{\beta})$ (minimum among $\boldsymbol{\beta}$s in the path).

Searching the entire neighbourhood before making a move constitutes an enormous task when the number of neighbours ($p(n-p)$) is large. Therefore, we follow Woodruff and Rocke's (1993) adaptation, which effectively reduces the number of neighbours from $p(n-p)$ to alternately $n-p$ and $p+1$: instead of considering every neighbour of the current point $\boldsymbol{\beta}$ that can be obtained by replacing a point in its corresponding subsample, we alternately find the best possible point to add to the subsample and the best possible one to remove. Once an observation is either added or removed from the subsample, it is placed on the tabu list (step 2(c)), so that it cannot be removed or added (step 2(a)) until after a specified number of moves κ ("length" of the tabu list).

Moving to a new $\boldsymbol{\beta}$ in TS in general is more computationally expensive than in SA. For large samples it makes sense to limit the search in step 2(a) to neighbours corresponding to a random subset of the observations with moderate size.

Although there is no obvious stopping criterion for TS, one can either specify a maximum number of moves or a time-limit. Alternatively, one can stop when no improvement of the objective value is observed in a certain number, say M, of consecutive moves. Optimally tuning the parameter M is not straightforward.

2.2.3 Incorporating IRWLS into SA and TS

The string of $\boldsymbol{\beta}$s visited by the basic SA or TS algorithm constitutes the "path" of the algorithm. Essentially, this path contains a large number of candidate $\boldsymbol{\beta}$s among which eventually the best one is chosen. This is quite similar to basic resampling, which only differs in the way the candidates are chosen. Much like we can improve basic resampling by using IRWLS steps, leading to fast-τ, we can also introduce IRWLS steps in the SA and TS algorithm. This can be done in two ways:

1. *Out-of-search*: At the end of the algorithm apply IRWLS steps to every $\boldsymbol{\beta}$ in the algorithm's path. The minimization in the last step is then simply performed over the improved $\boldsymbol{\beta}$s. This local improvement variant of SA and TS does not influence the path of the algorithm.

2. *Within-search*: Apply IRWLS steps to every $\boldsymbol{\beta}$ that is being considered for a move. This applies to the candidates obtained in step 2(a)i for SA and step 2(a) for TS. This local improvement variant generally influences the path of the algorithm.

The out-of-search local improvement described above can be implemented analogously to the structure used in the fast-τ algorithm. That is, we perform a small number k of initial IRWLS steps to each $\boldsymbol{\beta}$ in the path, and afterward keep the t best for further iteration until convergence.

For the within-search variant, one would perform k IRWLS steps to each candidate $\boldsymbol{\beta}$ and, once the path is complete, choose the t best candidates for further iteration. The difference between within-search and out-of-search is essentially that the latter only locally improves $\boldsymbol{\beta}$s that belong to the established path, whereas the former also improves $\boldsymbol{\beta}$s that are being considered but that may not be accepted. Although within-search is computationally more expensive than out-of-search, particularly for TS, this variant has the potential to produce better moves.

Finally, it is clear that the effect of introducing IRWLS in SA and TS will be less dramatic than in the random resampling case, since the basic SA and TS already attempt to move downwards during most of the search. Nevertheless, their search is limited to subsample-generated $\boldsymbol{\beta}$s and therefore IRLWS can make a difference (which we indeed observed in our simulations).

2.2.4 Reducing the cost of the objective function computations

Both SA and TS require many computations of the objective function, since these are used to determine the next move (see step 2(a)i for SA, and step 2(a) for TS). In order to be able to perform more moves in a reasonable amount of time, actual computations of M-scales can be replaced by r-step approximations. In case of TS, this modification has the disadvantage that one can not be certain that the resulting move actually corresponds to the largest objective function decrease. We will consider both full and approximate M-scale computation options in our simulation study.

In TS, where one would like to find the neighbour with the lowest objective value, another way for reducing the computational cost is again by using conditions (9) and (10) to discard neighbours without having to compute their τ-objective values. Approximate

M-scales can also be used in the random resampling algorithms. This has little effect on the computation time, however, since conditions (9) and (10) often greatly reduce the number of scale computations.

2.3 Diversification steps

We may assume that local improvements through IRWLS sufficiently succeed in moving the search path toward the bottom of the "valleys". Therefore, the main concern is that the algorithm at some point along its path should visit the parameter region corresponding to the global minimum.

For TS algorithms it is common to incorporate special "diversification steps" based on the *long-term memory* of the algorithm (regular moves can be considered as based on its *short-term memory*, i.e. the tabu list). These diversification steps attempt to re-direct the path toward regions of the parameter space that have not yet been visited. Such a move can be performed a small number of times during the algorithm. For example, in our context, we can record the mean residual of each observation across the different candidates $\boldsymbol{\beta}$ that determine the algorithm's path. A diversification move can then be inserted at some point in the algorithm by letting the next $\boldsymbol{\beta}$ be generated by the subsample consisting of those observations that were not fitted well on average up until then.

These long-term diversification steps often enhance the performance of the TS algorithm and can prove essential in certain situations. For example, when the data contains a tight group of outliers, the tabu list may not succeed in ridding the subsamples of those outliers, and thus all $\boldsymbol{\beta}$s in the path may be heavily determined by the outliers. In these cases, diversification may bring the search to a region which would otherwise have a very small chance of being visited by the search path, or of being represented by a random elemental subsample.

Note, however, that the random resampling-type algorithms can also be enhanced in the same manner. For example, the set of candidates $\boldsymbol{\beta}^I_{(j)}$, $j = 1, \ldots, N$ in step 2 of the fast-τ algorithm can be extended by adding one or more candidates corresponding to (non-random) subsamples consisting of observations that on average were badly fitted by $\boldsymbol{\beta}^I_{(j)}$, $j = 1, \ldots, N$.

Finally, for SA we find diversification steps to be somewhat contrary to the "cooling

off" spirit of the algorithm. However, some form of diversification can be implemented by running the SA algorithm multiple times with different starting points based on those observations that were poorly fitted by the previous paths.

We implemented diversification steps based on badly fitted observations and encountered some data configurations for which it critically enhanced the performance. Further study of alternative types of diversification steps would be desirable.

3 Simulation study

In this section we report the results of an extensive simulation study through which we mainly intended to examine whether the directed search approach of SA and TS can outperform random resampling (RR).

The values of several tuning parameters have to be set for the algorithms. Our choices are based on our experiments and the recommendations of Woodruff and Rocke (1993). Firstly, for all three algorithms we opted to work with subsamples of size p ("elemental" subsamples). In SA, the block size was set at $K = 0.5\,(n + p)$. The initial temperature T_0 is chosen such that acceptance rate in the first block is approximately 40% (based on a trial run). Furthermore, we set $\alpha = 0.85$ and we define the system to be "frozen" when 5 consecutive blocks have an acceptance rate smaller than 2%. In TS, a random subset of size $(n/2) + p$ is drawn from the data from which to construct the subsamples throughout the search (this reduces the size of the neighbourhood searched in each step). Finally, the size of the tabu list was set at $\kappa = 0.15\,n$.

For the local improvement, we used approximate IRWLS steps. The number k of initial improvement steps, as well as the number t of candidates to which IRWLS iterations were applied until convergence, are varied throughout the study.

Regarding the calculation of the objective values in SA and TS, both approximate M-scale (with $r = 2$) and full scale computation were tried (see Section 2.2.4). For RR and the fast-τ algorithm we only considered full scale computation and we applied conditions (9) and (10) to reduce the number of required computations. Finally, all algorithms have explicit diversification steps implemented as described in the Section 2.3.

All computations were performed in MATLAB. Code for the fast-τ algorithm (MAT-

LAB, Octave and R) is available online at `http://hajek.stat.ubc.ca/~matias/soft.html`.

Although there are infinitely many data configurations on which to test and compare the different algorithms, after extensive experiments we found that a reasonably representative picture is given by the following data:

- $(1-\epsilon)\,100\%$ of the points follow the regression model (1) where the $p-1$ covariates are distributed as $N_{p-1}(\mathbf{0}, I_{p-1})$, $x_{ip}=1$ is the intercept, $\epsilon_i \sim N(0,1)$ and $\boldsymbol{\beta}_0 = \mathbf{0}$;
- $\epsilon\,100\%$ of the points are "bad" high-leverage points: the covariates now follow a $N_{p-1}(\,\mathbf{d},\, 0.1^2\, I_{p-1})$ distribution where $\mathbf{d} = (100, 0, \ldots, 0)^t$, and the response variable $y_i \sim N(m, 0.1^2)$, with $m = 100, 120, \ldots, 200$.
- we consider $n = 100$; $p = 2$, 5, 10 and 20; and $\epsilon = 0.10$ and 0.20 .

For each combination of p, m and ϵ we generated 500 datasets, on which we applied the different algorithms. To make a fair comparison we provided each algorithm with a time-limit as follows. We first ran the RR algorithm with a fixed number of subsamples N, we recorded its computation time t_{RR} in CPU-seconds and then allowed both SA and TS to run for t_{RR} CPU-seconds. Note that SA sometimes required less than t_{RR} seconds to reach its stopping criterion. We did not use a stopping criterion, other than the time-limit, for the TS algorithm. We used $N = 200$, 500 and 1500, so that we essentially compared the algorithms with three different time intervals. In this way we intended to detect if a particular algorithm significantly improves its performance, relative to the other algorithms, when it is allowed more time to run.

The test data described above is such that, at the population level, two local minima appear in the τ-objective function, one of which is caused by the high-leverage points. However, for finite samples the τ-objective function is naturally affected by sampling variability and the number of local minima typically depends on the ratio p/n. By varying the dimension p and fixing the sample size at $n = 100$, we effectively vary the number of local minima that appear in the τ-objective functions:

- when $p = 2$ and $p = 5$, $n = 100$ is relatively large and $\tau_n(\boldsymbol{\beta})$ tends to have exactly two minima;
- when $p = 10$, $n = 100$ is relatively small and $\tau_n(\boldsymbol{\beta})$ often has several local minima;

– when $p = 20$ and $n = 100$ the τ-objective function has many (usually more than 10) local minima.

Increasing p also decreases the probability that a random subsample is free of bad leverage points from $\pm 80\%$ ($p = 2$) to $\pm 10\%$ ($p = 20$) for $\epsilon = 0.10$, and from $\pm 64\%$ to $\pm 0.7\%$ for $\epsilon = 0.20$. Note, however, that obtaining an outlier-free subsample does not guarantee that subsequent IRWLS local improvements will converge to the global minimum of $\tau_n(\boldsymbol{\beta})$, because of the following reasons: (a) an outlier-free subsample may coincidentally yield a corresponding $\boldsymbol{\beta}$ that fits the outliers better than the remaining data; (b) there may be more than one local minima that can be considered as not affected by outliers; and (c) the global minimum of the τ-objective function may correspond to a fit that accommodates the outliers (in these cases the τ-estimator was affected by the contamination).

We initially focus on the data with 10% outliers ($\epsilon = 0.10$). Let us first compare the three algorithms in their basic forms, that is without local improvement steps (which corresponds to $k = 0$ and $t = 0$). This amounts to the comparison made by Woodruff and Rocke (1993) in the context of MVE. Note that RR in this form represents the basic resampling algorithm.

For each algorithm we computed the average τ-objective obtained over the 500 samples. Figure 2 shows, for different values of N (200, 500 and 1500), the ratios of the averages of each algorithm to the average of the RR algorithm as a function of p. Hence, whenever a curve is below the horizontal line, the particular algorithm performed better than RR, and the lower the ratio the better the performance. The ratios have been further averaged over the 6 data configurations corresponding to the different m-values. In fact, the ratios between the τ-values were fairly similar for all 6 of the configurations. For SA (dashed) and TS (dotted), both the full (circles) and approximate (squares) M-scale options are shown.

It can be seen that in this case RR is clearly outperformed by all other algorithms. The difference in performance increases with p and with the allowed computation time.

Naturally, the performance of each individual algorithm also improved by allowing it more running time. The evolution over time (represented in our case by increasing values of N) of the performance of the basic RR algorithm for $p = 5$, 10 and 20 is depicted by the dash-dotted curves in the upper panels of Figure 5.

Let us now briefly look at what happens if we locally improve the end-result of each

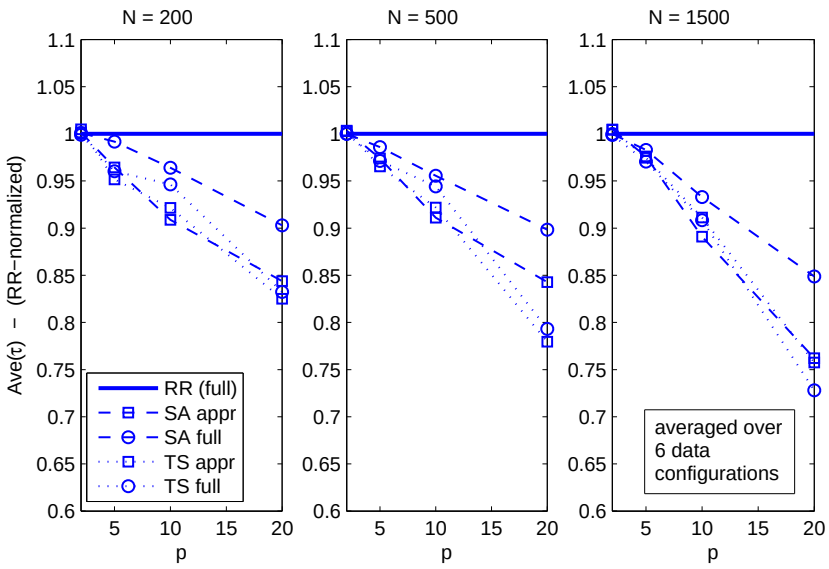

Figure 2: Basic resampling (RR) vs. basic SA and basic TS ($k = 0, t = 0$); both full (circles) and approximate (squares) M-scale computation. Average τ-objective values: ratio w.r.t. RR; averaged over 6 data configurations with 10% outliers

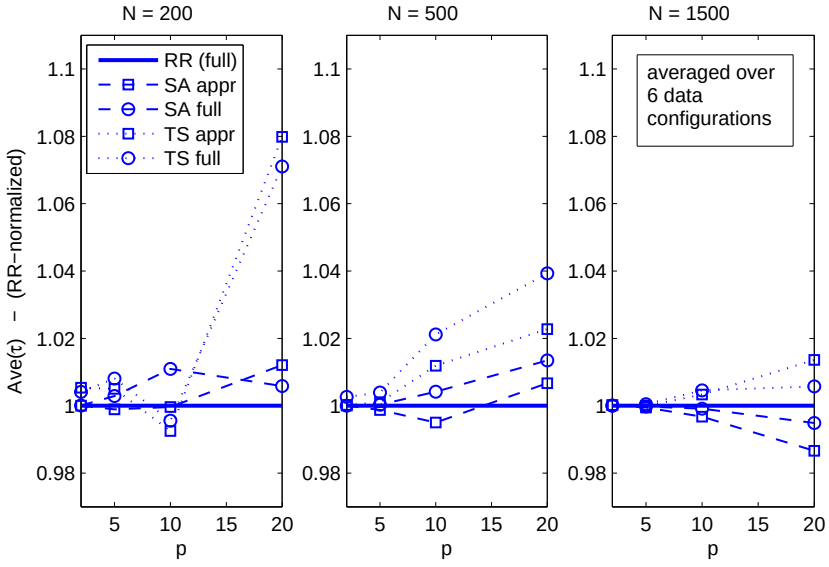

Figure 3: Basic resampling (RR) vs. basic SA and basic TS with local improvement on end-result ($k = 0, t = 1$); both full (circles) and approximate (squares) M-scale computation. Average τ-objective values: ratio w.r.t. RR; averaged over 6 data configurations with 10% outliers

algorithm here. That is, we perform IRWLS steps until convergence to the best $\boldsymbol{\beta}$ found by each algorithm. This corresponds to setting $k = 0$ and $t = 1$. We note that this yields a large improvement for all five algorithms. This is especially true for random resampling, for which the degree of improvement can be seen in Figure 5, where the current setting is represented by the dashed curves.

Figure 3 depicts the ratios of the average τ-objective values for the various algorithms to the one of RR. Comparing these plots to Figure 2, it can be seen that the local improvement on the end-result made the heuristic algorithms lose much of their advantage over basic resampling. In fact, the random resampling approach generally dominates the other algorithms when $N = 200$ and $N = 500$. On the other hand, the plot for $N = 1500$ indicates that if we allow SA to run longer it may still perform better than RR, particularly in the small-sample case ($p = 20$) that generally corresponds to many local minima of the τ-objective function.

Finally, we introduce initial IRWLS steps by choosing $k = 2$ and we set $t = 5$ (so that RR now actually represents fast-τ). For SA and TS, we consider both the within-search and out-of-search variants, as described in Section 2.2.3. The improvement that this brings over the previous settings (where $k = 0$) for the RR algorithm can be seen in Figure 5 (most notably in the lower panels, which zoom in on the respective upper panels), where the solid lines correspond to fast-τ. On a side note, these plots also show that $N = 200$ seems sufficient for fast-τ to find the global minimum of $\tau_n(\boldsymbol{\beta})$ for small values of p.

The respective ratios between the average τ-objective values for the current setting are shown in Figure 4. The performance of SA and TS with the full and approximate M-scale options were very similar to each other and only the first option is shown. The results for SA and TS are now shown with the within-search (triangles) and out-of-search (circles) IRWLS options. The plots indicate that random resampling or fast-τ now clearly outperforms the SA and TS algorithms, and it does not appear as if this can be overturned by allowing SA and TS more running time (although admittedly we only considered time limits associated with N up to 1500 random subsamples in RR).

We now turn our attention to the case $\epsilon = 0.20$. The basic variants of these algorithms ($k = 0$) performed similarly to the case $\epsilon = 0.10$. However, when $k = 2$ and $t = 5$ (as in Figure 4) we noticed a qualitative difference in the performance of the algorithms. The

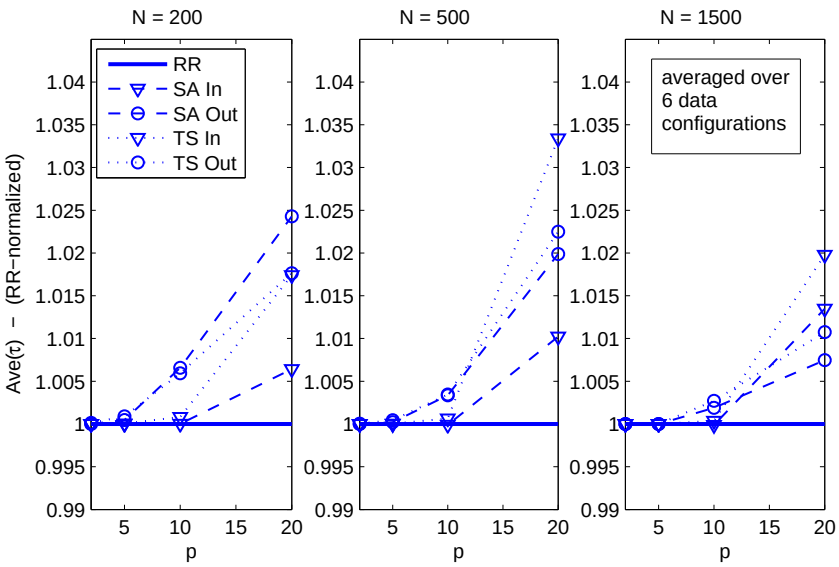

Figure 4: Fast-τ (RR) vs. SA and TS with IRWLS ($k = 2, t = 5$); both within- (triangles) and out-of-search (circles) local improvement. Average τ-objective values: ratio w.r.t. fast-τ; averaged over 6 data configurations with 10% outliers

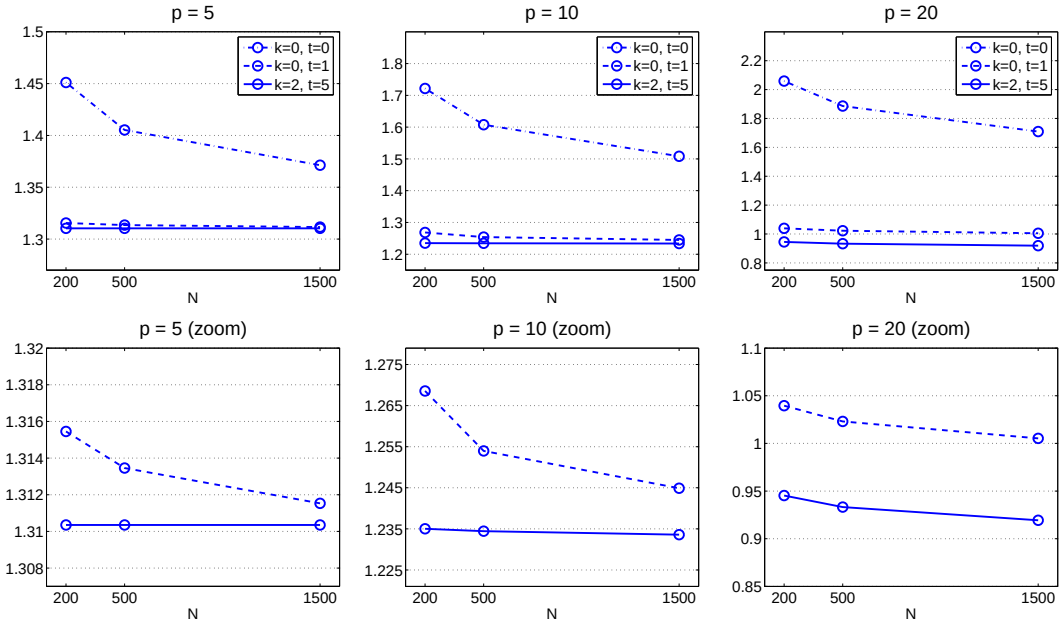

Figure 5: Average τ-objective values for random resampling/fast-τ in case $\epsilon = 0.10$; $n = 100$; $p = 5$ (left), $p = 10$ (middle) and $p = 20$ (right); averaged over 6 data configurations; dash-dotted: basic resampling – dashed: local improvement on end result – solid: fast-τ with $k = 2$ and $t = 5$; lower panels are zoomed

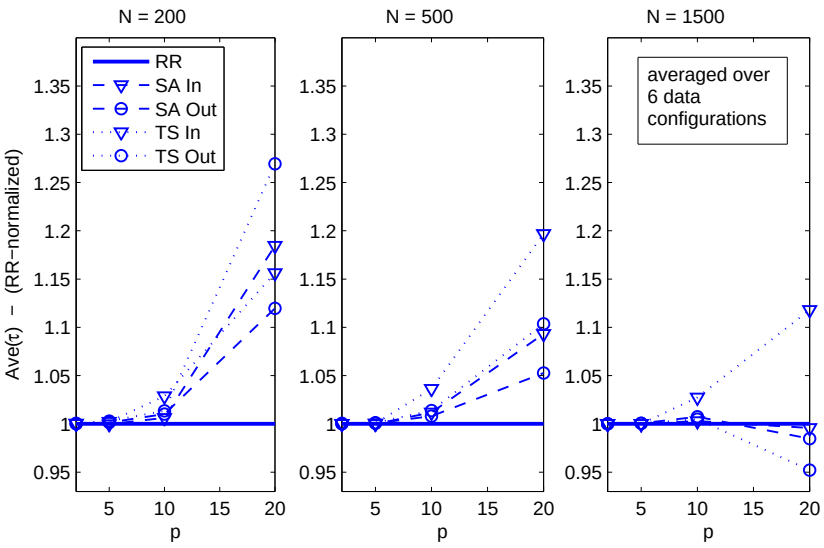

Figure 6: Fast-τ (RR) vs. SA and TS with IRWLS ($k = 2, t = 5$); both within- (triangles) and out-of-search (circles) local improvement. Average τ-objective values: ratio w.r.t. fast-τ; averaged over 6 data configurations with 20% outliers

corresponding plots are shown in Figure 6. Note that for $p = 20$ and $N = 1500$, the fast-τ algorithm is outperformed by some of the SA and TS variants.

This appears to demonstrate that in situations with many local minima, such as our $n = 100$ and $p = 20$ case, directed search heuristic algorithms can perform well if computation time is not a concern. However, assuming that time is indeed not severely limited, our next experiment shows that the fast-τ algorithm still yields the best results: we only need to increase either k or t, which measure the amount of local improvement applied.

We considered the case $\epsilon = 0.20$, $p = 20$, $N = 1500$ with a larger number t of candidates that are locally improved until convergence, by setting $t = 10$, $t = 20$ and $t = 50$. The left panel of Figure 7 summarizes the results. It shows again the ratios between the average τ-objective values, as a function of t. We only show the results for the out-of-search SA and TS variants, since they generally outperformed the corresponding within-search versions. Since it was not clear a priori to what extent SA and TS may benefit from increasing t, we considered two options for SA and TS to make the comparisons fair: either the algorithm uses a larger value of t (as it is done for the fast-τ), or it is allowed to spend the additional CPU-time on performing more moves while keeping $t = 5$. The results of both options are depicted in the plot using circles and squares respectively.

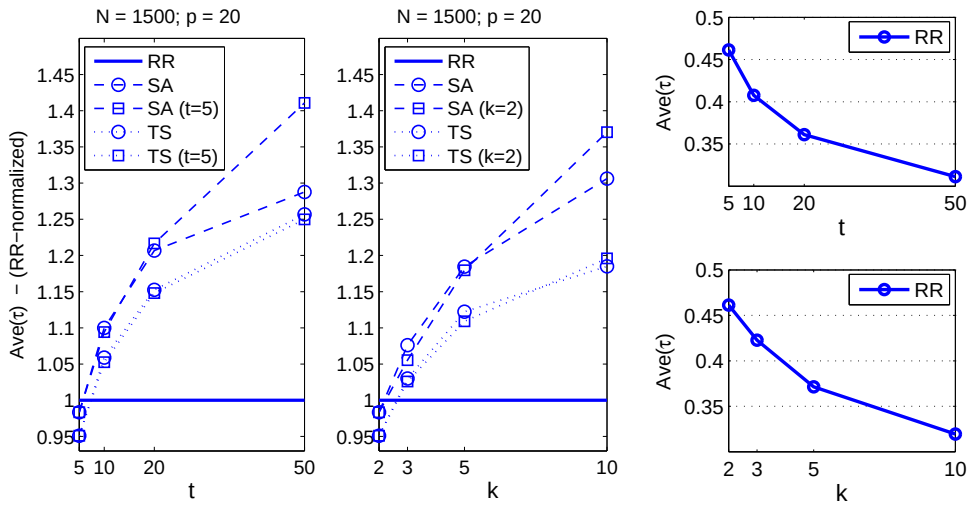

Figure 7: The case $p = 20$, $n = 100$ and 20% outliers; $N = 1500$; fast-τ (RR) vs. SA and TS, out-of-search (ratios); averaged over 6 data configurations; left: $k = 2$, t varies – middle: $t = 5$, k varies – right: fast-τ absolute values (upper: t varies, lower: k varies)

The middle panel of Figure 7 contains the results of a similar experiment: we kept $t = 5$ and increased the number k of initial local improvement steps performed on each candidate to $k = 3$, $k = 5$ and $k = 10$. Again we considered two options for SA and TS: increasing k or maintaining $k = 2$ and spending the additional running time on additional moves. It can be seen that increasing either t or k in this situation has a large positive effect on the fast-τ performance. While the other algorithms also see their average objective values decreased, the effect is much smaller. The right panel of Figure 7 plots the (absolute) average objective values for fast-τ corresponding to the two other panels respectively.

We can summarize the results of this simulation study as follows:

- Without local improvements, random resampling is inferior to the directed search approach of SA and TS. However, when one introduces local improvement steps the latter algorithms quickly lose most of their advantage.

- With sufficient local improvement steps, random resampling was found to be generally more efficient than the directed search algorithms.

- Usually a small number of initial local improvement steps k on each candidate, and a small number of fully improved candidates t will be sufficient. In situations with

many local minima, at least one of these two numbers should be higher. In the data sets considered in this study the number of local minima of $\tau_n(\boldsymbol{\beta})$ was found to depend on the ratio p/n (smaller values of this ratio producing fewer local minima).

- Ideally the best strategy is to apply exhaustive local improvements to a *large* number N of random candidates ($N \geq 1500$, say).

Remark The fast-τ algorithm computes τ-estimates with the same computational complexity as the fast-S algorithm computes S-estimates. Fast-τ requires a bit more time than fast-S (given the same settings for N, k and t), mainly because it needs to compute a larger number of M-scales and because the IRWLS step for τ involves more calculations than that for S. Some computation times for our implementations of fast-τ and fast-S are listed in Table 1, for different values of n and p. Both algorithms use $N = 1500$, $k = 2$ and $t = 5$. The times (in seconds) are averaged over 20 samples for each of the six data configuration described above with $\epsilon = 0.10$. Table 1 also lists the average estimated number of local minima that appear in the respective τ- and S-objective functions for the corresponding samples. These estimates were obtained by counting the number of different IRWLS convergence points in an application of the resampling algorithm with exhaustive local improvement (see Section 2.1). The fact that the τ-objective function has fewer local minima (and thus is more "smooth"), reflects the τ-estimator's higher efficiency. It also demonstrates that we can expect the fast-τ to have a better accuracy than fast-S (in terms of finding the respective global minimum) for any given value for N, since fast-τ is more likely to cover every local minimum than fast-S is. The extent to which this impacts the optimal choice of N and the other parameters is the subject of further research.

		$p = 10$				$p = 20$			
	n	100	200	500	1000	100	200	500	1000
time (s)	fast-τ	2.48	3.26	5.82	10.74	2.52	3.40	7.49	15.94
	fast-S	1.25	1.67	3.47	6.82	1.66	2.36	5.96	14.35
#minima	τ	5.4	2.4	2.0	2.0	94.7	6.2	2.0	2.0
	S	110.1	24.5	13.6	2.2	882.5	319.2	81.2	11.7

Table 1: Computation times for fast-τ and fast-S ($N = 1500$, $k = 2$, $t = 5$), with the average estimated number of local minima in the τ- and S-objective function

4 Conclusion

In this paper we considered the problem of computing τ-estimators for linear regression. By combining random resampling with local IRWLS improvements we constructed the so-called fast-τ algorithm, which is analogous to well-known algorithms for LTS (Rousseeuw and Van Driessen 2002) and S-estimators (Salibian-Barrera and Yohai 2006). We then investigated how this random resampling approach compares with heuristic algorithms such as Simulated Annealing and Tabu Search. We concluded from extensive simulations that random resampling is inferior to both Simulated Annealing and Tabu Search when local improvements are not applied. However, it becomes much more efficient and generally outperforms its competitors when IRWLS steps are introduced.

Although it is hardly feasible to consider all possible ways to tune the Simulated Annealing and Tabu Search algorithms, we made a careful effort to find the best tuning options for these algorithms based on our own experiments and on guidelines from the literature.

We would like to note that our findings about the relative performance of random resampling are presumably also relevant for the case of S-estimators and the fast-S algorithm. Notice that τ-estimators reduce to S-estimators when $\rho_1 = \rho_2$ in (3) and (4) and that the objective functions of S- and τ-estimators are very similar.

Finally, the fast-τ algorithm, like fast-S, does not scale well to problems in very large dimensions (since it becomes exponentially more difficult to draw an outlier-free subsample). Therefore, if $p > 20$ say, an adaptation can be considered which adds one extra candidate to the N random candidates in step 1 of fast-τ, namely an initial estimate of $\boldsymbol{\beta}_0$ which would be reasonably robust in practice and fast to compute for any p. Such an estimate was proposed by Pena and Yohai (1999).

Acknowledgment

This research has been supported by NSERC Discovery Grants. The second author has been partially supported by a PIMS postdoctoral fellowship.

References

Arslan, O., Edlund, O. and Ekblom, H. (2002). Algorithms to compute CM- and S-estimates for regression. *Metrika*, 55, 37–51.

Berrendero, J.R., Mendez, B.V.M. and Tyler, D.E. (2006). On the maximum bias functions of MM-estimates and constrained M-estimates of regression. *The Annals of Statistics*, to appear.

Glover, F. (1986). Future paths for Integer Programming and Links to Artificial Intelligence *Computers and Operations Research*, 5, 533–549.

Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation; Part I, graph partitioning. *Operations Research*, 37, 865–892.

Maronna, R.A., Martin, R.D. and Yohai, V.J. (2006). *Robust Statistics: Theory and methods*, Wiley and Sons, West Sussex, England.

Pena, D. and Yohai, V.J. (1999). A fast procedure for outlier diagnostics in large regression problems. *Journal of the American Statistical Association*, 94, 434–445.

R Development Core Team (2005). *R: A language and environment for statistical computing*, Vienna, Austria. Available on-line at `http://www.r-project.org`.

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

Rousseeuw, P.J. and Leroy, A.M. (1987). *Robust regression and outlier detection.* Wiley, New York.

Rousseeuw, P.J. and Van Driessen (2002). Computing LTS regression for large data sets. *Estadistica*, 54, 163–190.

Rousseeuw, P.J. and Yohai, V.J. (1984). Robust regression by means of S-estimators. *Robust and Nonlinear Time Series Analysis. Lecture Notes in Statistics*, 26, 256–272. Springer, New York.

Ruppert, D. (1992). Computing S estimators for regression and multivariate location/dispersion.

Journal of Computational and Graphical Statistics, 1, 253–270.

Sakata, S. and White, H. (2001). S-estimation of nonlinear regression models with dependent and heterogeneous observations. *Journal of Econometrics*, 103, 5–72

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

Woodruff, D.L. and Rocke, D.M. (1993). *Journal of Computational and Graphical Statistics*, 2, 69–95.

Yohai, V.J. (1987). High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, 15, 642–656.

Yohai, V.J. 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, 406–413.

Yohai, V.J. and Zamar, R.H. (1991), Discussion of "Least Median of Squares Estimation in Power Systems", by Mili, L., Phaniraj, V., and Rousseeuw, P. J., *IEEE Transactions on Power Systems*, **6**, 520.

Yohai, V.J. and Zamar, R.H. (1998). Optimal locally robust M-estimates of regression. *Journal of Statistical Planning and Inference*, 64, 309–323.