The Fast-τ Estimator for Regression

Matias Salibian-Barrera, Gert Willems, and Ruben Zamar

Yohai and Zamar’s τ-estimators of regression have excellent statistical properties but are nevertheless rarely used in practice because of a lack of available software and the general impression that τ-estimators are difficult to approximate. We will show, however, that the computational difficulties of approximating τ-estimators are similar in nature to those of the more popular S-estimators. The main goal of this article is to compare an approximating algorithm for τ-estimators based on random resampling with some alternative heuristic search algorithms. We show that the former is not only simpler, but that when enhanced by local improvement steps it generally outperforms the considered heuristic search algorithms, even when these heuristic algorithms also incorporate local improvement steps. Additionally, we show that the random resampling algorithm for approximating τ-estimators has favorable statistical properties compared to the analogous and widely used algorithms for S- and least trimmed squares estimators.

Key Words: Random resampling; Robust regression; Simulated annealing; Tabu search.

1. INTRODUCTION

Consider the usual linear regression model

\[ y_i = \beta_0^t x_i + \epsilon_i, \quad i = 1, \ldots, n, \]  

for which we want to estimate the vector of regression coefficients \( \beta_0 \in \mathbb{R}^p \) based on \( n \) independent observations \((y_i, x_i) \in \mathbb{R} \times \mathbb{R}^p\). The error terms \( \epsilon_i \) are assumed to be independent from the covariates \( x_i \) and identically distributed with zero center and unknown scale \( \sigma_0 \).

The least squares (LS) estimator for \( \beta_0 \) 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. 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.

Approximating algorithms to several of these robust estimators are widely available
as they are implemented in various statistical software packages (see for example the re-
cent robustbase package for R (R Development Core Team, 2005) available online at
the CRAN website http://cran.r-project.org), and the robust library for S-PLUS. Un-
fortunately, another common feature of these estimators is the time-consuming nature of
their computation. In fact, exact computation is generally not feasible except for small data
sets. Considerable effort has been made, however, to construct approximate algorithms that
perform reasonably well in practice. 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 approximation of MM-estimators as well.

Of the above estimators, only the MM-estimators combine high breakdown point with
high efficiency, which makes them a good option for inference purposes. However, \( \tau \)-
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 study of algorithms to approximate \( \tau \)-estimators has been somewhat
neglected in the literature.

The \( \tau \)-estimator is defined as

\[
\hat{\beta}_n = \arg\min_{\beta \in \mathbb{R}^p} \tau_n(\beta),
\]

where the \( \tau \)-scale \( \tau_n(\beta) \) is given by

\[
\tau_n^2(\beta) = s_n(\beta) \frac{1}{n b_2} \sum_{i=1}^n \rho_2 \left( \frac{y_i - \beta' x_i}{s_n(\beta)} \right),
\]

with \( s_n(\beta) \) an M-estimator of scale that solves

\[
\frac{1}{n} \sum_{i=1}^n \rho_1 \left( \frac{y_i - \beta' x_i}{s_n(\beta)} \right) = b_1.
\]

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:

\[
b_j = E_{\phi}[\rho_j(u)]; \quad j = 1, 2,
\]

where \( \Phi \) is the standard normal distribution. In the special case where \( \rho_1 = \rho_2 \), we
have \( \tau_n(\beta) = s_n(\beta) \) so that (1.2) reduces to the definition of an S-estimator. Like MM-
estimators, \( \tau \)-estimators have the breakdown point of an S-estimator based on the loss
function \( \rho_1 \), while its efficiency is determined by the function \( \rho_2 \) used in (1.3) (see Yohai
and Zamar 1988).
The choice of the loss functions $\rho_1$ and $\rho_2$ can be of considerable practical and theoretical importance. In our examples and simulations we used the so-called optimal weight functions introduced by Yohai and Zamar (1998). This family of loss functions confers excellent robustness properties on estimators such as $\tau$, $S$, and MM-estimators (and is therefore preferred over the commonly used Tukey bisquare family). The optimal $\rho$-function with tuning parameter $c$ is given by

$$
\rho(t) = \begin{cases} 
1.38 \left( \frac{t}{c} \right)^2 & |\frac{t}{c}| \leq \frac{2}{3} \\
0.55 - 2.69 \left( \frac{t}{c} \right)^2 + 10.76 \left( \frac{t}{c} \right)^4 - 11.66 \left( \frac{t}{c} \right)^6 + 4.04 \left( \frac{t}{c} \right)^8 & \frac{2}{3} < |\frac{t}{c}| \leq 1 \\
1, & |\frac{t}{c}| > 1.
\end{cases}
$$

The breakdown point of the $\tau$-estimator $\hat{\beta}_n$ is given by $\epsilon^{\ast}(\hat{\beta}_n) = b_1 / \sup_t \rho_1(t)$. To obtain consistency and $\epsilon^{\ast}(\hat{\beta}_n) = 50\%$, we chose the tuning parameters $c_1 = 1.214$ and $b_1 = 0.5$ for $\rho_1$ in (1.4). The choices $c_2 = 3.270$ and $b_2 = 0.128$ for $\rho_2$ yield a $\tau$-estimator with 95% efficiency when the errors in (1.1) are normally distributed (see Yohai and Zamar 1988). Moreover, the associated objective value $\tau_n(\hat{\beta}_n)$ then provides an estimator of the scale $\sigma_0$ 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 $\tau$-estimators relative to those of MM-estimators. Berrendero et al. (2007) showed that, when the $\rho$ functions belong to Tukey’s bisquare family, MM-estimators have lower asymptotic bias than $\tau$-estimators for proportions of contaminations $\epsilon$ lower than 0.20, whereas for $\epsilon > 0.20$ the situation reverses. The formulas of Berrendero et al. (2007) also showed, however, that if $\rho_1$ and $\rho_2$ belong to the optimal family above, then the maximum asymptotic bias of MM- and $\tau$-estimators become very close for $\epsilon < 0.20$ while for $\epsilon > 0.20$ $\tau$-estimators still have notably lower maximum asymptotic bias.

It should be noted that for both $\tau$- 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 $\tau$-estimators are not often used 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 $\tau$-objective function in (1.3) makes its minimization a difficult task, even by robust regression standards. However, although the function $\tau_n(\beta)$ is more involved than the S-objective function $s_n(\beta)$ in (1.4), both functions present very similar features. Namely: they are nonconvex, have possibly multiple local minima, and evaluating them involves solving a nonlinear equation, which makes their calculation time-consuming.

In this article we will study approximating algorithms for $\tau$-regression estimators defined by (1.2). 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 (IRLS) 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 $\tau$-estimators, which in fact demonstrates that the computational problem of $\tau$-estimators can be handled in the same way as that of S-estimators. As suggested by a referee, we will also evaluate the improvements to these
algorithms that may be obtained by using the approach of Olive and Hawkins (2007).

The performance of the random resampling approach in terms of solving (1.2) will be compared by means of an extensive simulation study 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. In this study we also improved these two heuristic search algorithms by including local refinement steps. Finally, some statistical properties of the random resampling approximation for τ-estimators will be examined and compared to the analogous approximations for LTS and S-estimators, which are already widely used in practice.

Section 2 describes and discusses the algorithms. Section 3 presents the results from the simulation experiments, while Section 4 contains our concluding remarks.

2. FINDING THE GLOBAL τ-MINIMUM

To illustrate the difficulty of minimizing the τ-objective function (1.3) we generated a simple artificial dataset 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 \( \tau_n(\beta) \) for these data, which reveals two local minima: \( \beta^{(1)} \) and \( \beta^{(2)} \), say. The regression lines corresponding to these minima are indicated in Figure 1(a) as well. It is clear that \( \beta^{(1)} \) can be regarded as the local minimum that is unaffected by the outliers, whereas \( \beta^{(2)} \) is induced by these outliers and should arguably be avoided. A satisfactory algorithm to solve (1.2) should return \( \beta^{(1)} \) as the τ-estimate, since \( \tau_n(\beta^{(1)}) = 1.653 < \tau_n(\beta^{(2)}) = 1.927 \). In the next sections we discuss different types of algorithms to optimize functions of this form.

![Example data: two local minima](image1.png)

![τ-objective function indicating the two local minima β^(1) and β^(2)](image2.png)

Figure 1. Example of a τ-objective function \( \tau_n(\beta) \) for a simulated dataset 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 \( \beta^{(1)} \) and \( \beta^{(2)} \).
2.1 Resampling Algorithms

Figure 1 illustrates the main problem in optimizing the $\tau$-objective function $\tau_n(\beta)$: 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 nonconvex function of this form is presented by the random resampling (RR) procedure (Rousseeuw 1984). This procedure generates a large number of candidates $\beta_{(j)}$, $j = 1, \ldots, N$, by drawing random subsamples of size $h \geq p$ from the data (without replacement) and letting $\beta_{(j)}$ be the LS fit to the $j$th subsample. When $N$ is large enough, it is assumed that the candidate $\beta$’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 random resampling. See Hawkins and Olive (2002) for a discussion on the consistency and breakdown properties of this algorithm. It was implemented in the PROGRESS algorithm of Rousseeuw and Leroy (1987) and can be described as follows:

**Basic RR algorithm:**

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

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 (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 IRLS iterations.

The IRLS step for $\tau$-estimators is derived as follows. Yohai and Zamar (1988) showed
that any local minimum $\hat{\beta}_n$ of $\tau_n(\beta)$ satisfies the estimating equation

$$\sum_{i=1}^{n} \left[ W_n(\hat{\beta}_n) \psi_1(e_i(\hat{\beta}_n)) + \psi_2(e_i(\hat{\beta}_n)) \right] x_i = 0,$$  \hspace{1cm} (2.1)

where

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

Here, $e_i(\hat{\beta}_n) = (y_i - \hat{\beta}_n^t x_i)/s_n(\hat{\beta}_n)$, and $\psi_1$ and $\psi_2$ are the derivatives of $\rho_1$ and $\rho_2$, respectively. Equation (2.1) can be rewritten as

$$\hat{\beta}_n = \left( \sum_{i=1}^{n} \frac{\psi_n(e_i(\hat{\beta}_n))}{e_i(\hat{\beta}_n)} x_i x_i^t \right)^{-1} \sum_{i=1}^{n} \frac{\psi_n(e_i(\hat{\beta}_n))}{e_i(\hat{\beta}_n)} x_i y_i,$$ \hspace{1cm} (2.2)

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

Note that (2.2) is the weighted LS fit to the data $(y_i, x_i)$, $i = 1, \ldots, n$, with weights $w_i = \psi_n(e_i(\hat{\beta}_n))/e_i(\hat{\beta}_n)$. Suppose now that $\beta^{(m)}$ is the current candidate, then the next IRLS step yields $\beta^{(m+1)}$ as follows:

$$\beta^{(m+1)} := \left( \sum_{i=1}^{n} \frac{\psi_n(e_i(\beta^{(m)}))}{e_i(\beta^{(m)})} x_i x_i^t \right)^{-1} \sum_{i=1}^{n} \frac{\psi_n(e_i(\beta^{(m)}))}{e_i(\beta^{(m)})} x_i y_i, \hspace{1cm} m = 1, 2, \ldots \hspace{1cm} (2.3)$$

Iterations of this step lead to a solution of (2.2) and thus to a local minimum of $\tau_n(\beta)$. Extensive experiments have shown that these IRLS steps for $\tau$-estimators 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 IRLS, we can construct an algorithm for approximating $\tau$-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 RR algorithm (immediately after Step 1):

1’. Improve each $\beta_{(j)}$, by applying IRLS until convergence.

Hence, each randomly generated candidate is now locally improved before the best one is selected.

Unfortunately, using IRLS iterations until convergence on each random candidate is rather costly (even when using approximate IRLS steps as discussed in Section 2.3.1). However, the number of IRLS steps can be reduced, by observing that the largest reductions in objective value generally occur in the first few iterations. Therefore, just one or two IRLS steps are often enough to determine which candidates are more likely to lead to smaller values of the criterion. Hence, for each random resampling candidate $\beta_{(j)}$ we can perform $k = 1$ or $k = 2$ IRLS iterations and retain the best $t$ partially improved resampling candidates ($t$ will generally be much smaller than $N$). We can then apply IRLS until convergence to these $t$ partially improved candidates only and report the one with the smallest
final objective function as our estimate $\hat{\beta}_n$.

These considerations lead to the following version of the RR algorithm, which forms the core of what we call the fast-$\tau$ algorithm by analogy with the above-mentioned algorithms for S- and LTS estimators.

**RR with efficient local improvements (fast-$\tau$):**

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

In our experience, the choice $k = 2$ (the number of IRLS 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 is often the case when the ratio $n/p$ is small), then higher values of $k$ or $t$ may be required to find a good approximation to the global minimum of $\tau_n(\beta)$ using this algorithm. Setting $t = N$ reduces to the exhaustive local improvement algorithm mentioned above, while the choice $k = 0$ and $t = 0$ corresponds to basic RR.

**Olive and Hawkins (2007) modification:** Unless the number of candidates $N$ is allowed to increase unboundedly when the sample size $n$ tends to infinity, the above fast-$\tau$ algorithm lacks consistency and formal high-breakdown (Adrover, Bianco, and Yohai 2001). A referee conjectured that these properties may be attained by including two simple additional candidates to our algorithm, as proposed by Olive and Hawkins (2007) for the fast-LTS. More specifically, in addition to the $t$ candidates $\beta^E_{(i)}$ in Step 5 of fast-$\tau$, we evaluate the objective function on the ordinary least squares regression estimator, and the estimator obtained after 10 IRLS steps starting from the least squares fit based on the half-sample with responses $y_j$ closest to the median response $\text{median}_i(y_i)$ (see Olive and Hawkins 2007, for more details). The approximated $\tau$-regression estimator is then the vector with minimum objective function among the $t + 2$ candidates.

### 2.2 Heuristic Algorithms

It has been argued that random resampling as a technique to generate candidate $\beta$’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) showed 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 (see Maronna et al. (2006, p. 198) for a concentration step for the subsampling algorithm for MVE, and Olive (2007) for how to make this algorithm high-breakdown and \(\sqrt{n}\)-consistent). 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-\(\tau\) algorithm, greatly enhances its performance. Therefore, it is of interest to investigate how these alternative heuristic algorithms compare with random resampling when local improvement steps are used. To investigate this question in the case of \(\tau\)-regression estimators we compared the performance of the fast-\(\tau\) algorithm described earlier with that of the simulated annealing (SA) and tabu search (TS) algorithms. Furthermore, we also improved SA and TS by implementing local improvement steps into them.

Both SA and TS use a randomly generated starting point and navigate through the parameter space by moving from the current point to a neighboring one according to different criteria. Hence, we need to define a notion of neighborhood appropriate to our application. While it is possible to define neighborhoods by placing a continuous distribution on the parameter space of \(\beta\) (see, e.g., Sakata and White 2001), we prefer the more straightforward approach used by Woodruff and Rocke (1993). We will restrict our search to subsample-generated \(\beta\)’s, since these are expected to cover the more relevant “valleys” of the objective function, as mentioned in the previous section.

More specifically, at the \(k\)th iteration of these algorithms, we say that \(\beta_{(k+1)}\) is a neighbor of the current point \(\beta_{(k)}\) if the corresponding subsamples differ in only one observation. Thus, if we use elemental subsamples, each point visited by these algorithms has \(p(n-p)\) neighbors.

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, for example, Johnson et al. (1989) and Woodruff and Rocke (1993). SA’s basic idea is to move through the parameter space by randomly choosing a neighbor 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 \(\Delta\) 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 $\beta$;

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 neighbor $\beta'$ of $\beta$ and compute $\Delta = \tau_n(\beta') - \tau_n(\beta)$;
      
      ii. if ($\Delta < 0$) set $\beta = \beta'$ (accept move);
          
      else, with probability $\exp(-\Delta/T)$, set $\beta = \beta'$ (accept move);

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

3. let $\hat{\beta}_n = \arg\min \tau_n(\beta)$ (minimum among $\beta$’s in the path).

The SA algorithm crucially depends on the tuning of parameters $T_0$, $K$, and $\alpha$, and to a lesser extent on the definition of a “frozen system.” In these matters, we mainly followed Woodruff and Rocke (1993) 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 neighbor of the current point to move to, TS selects the point that has the lowest objective value among all the neighbors. 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 neighbor 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. Neighbors that appear in this list are rejected for the next move even if they would produce the best objective value. In our context, a neighbor of the current candidate $\beta$ finds itself on the tabu list if the observation in which its corresponding subsample differs from that of $\beta$, was involved in a recent previous swap (see also the following). 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 $\beta$;

2. repeat until a stopping criterion is met:
   
   (a) for every neighbor $\beta'$ of $\beta$ that is not on the tabu list, compute $\tau_n(\beta')$
   
   (b) set $\beta = \beta'$, with $\beta'$ the neighbor which minimized $\tau_n(\beta')$

   (c) update the tabu list

3. let $\hat{\beta}_n = \arg\min \tau_n(\beta)$ (minimum among $\beta$’s in the path).
Searching the entire neighborhood before making a move constitutes an enormous task when the number of neighbors \((p(n - p))\) is large. Therefore, we follow Woodruff and Rocke’s (1993) adaptation, which effectively reduces the number of neighbors from \(p(n - p)\) to alternately \(n - p\) and \(p + 1\): instead of considering every neighbor of the current point \(\beta\) 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 \(\kappa\) (“length” of the tabu list).

Moving to a new \(\beta\) in TS in general is more computationally expensive than in SA. For large samples it makes sense to draw a random subset of the data (of moderate size) and to only consider subsamples made up by points from this subset.

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 IRLS into SA and TS

The string of \(\beta\)’s visited by the basic SA or TS algorithm constitutes the “path” of the algorithm. Essentially, this path contains a large number of candidate \(\beta\)’s among which eventually the best one is chosen. This is quite similar to basic RR, which differs only in the way the candidates are chosen. Much like we can improve basic RR by using IRLS steps, we can also introduce IRLS 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 IRLS steps to every \(\beta\) in the algorithm’s path. The minimization in the last step is then simply performed over the improved \(\beta\)’s. This local improvement variant of SA and TS does not influence the path of the algorithm.

2. **Within-search**: Apply IRLS steps to every \(\beta\) 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-\(\tau\) algorithm (RR with efficient local improvement). That is, we perform a small number \(k\) of initial IRLS steps to each \(\beta\) in the path, and afterward keep the \(\tau\) best for further iteration until convergence.

For the within-search variant, one would perform \(k\) IRLS steps to each candidate \(\beta\) 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 \(\beta\)’s that belong to the established path, whereas the former also improves \(\beta\)’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 IRLS 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 $\beta$’s and therefore IRLS can make a difference (which we indeed observed in our simulations).

2.3 Reducing the Cost Associated with M-scale Computations

All three algorithms (RR, SA, and TS) involve frequent computations of M-estimators of scale as defined by (1.4). In particular, M-scales are required when computing the objective function (1.3) of the $\tau$-estimator and also in the local improvement or IRLS Step (2.3). Computing an M-estimator of scale requires an iterative algorithm and can thus be considered costly. Fortunately we can reduce the overall cost of our algorithms by using approximated M-scales and by avoiding unnecessary M-scale computations, as explained below.

2.3.1 Approximating M-scales in IRLS Steps

We can considerably reduce the computation cost of the algorithms by replacing $s_n(\beta^{(m)})$ in the IRLS Step (2.3) by an approximate value. The latter can be obtained as the $r$th step of an iterative algorithm to solve (1.4), where iterations are started from $s^{(l)} = \text{mad}(r \cdot \beta)^{(m)})$ and at the $l$th step, $s^{(l)}$ is given by

$$s^{(l)} = s^{(l-1)} - \frac{1}{n b_1} \sum_{i=1}^{n} \rho_1 \left( \frac{y_i - \beta^{(m)}}{s(t-1)} \right), \quad l = 1, 2, \ldots (2.4)$$

We refer to the resulting local improvement steps (using $r = 1$) as approximate IRLS steps. In our experience, the approximate steps perform nearly as well as the actual IRLS steps, although convergence is slightly slower. We will therefore work with approximate IRLS steps in the rest of this article.

2.3.2 Approximating M-scales in the $\tau$-Objective Function

Each algorithm also requires many computations of the objective function of the $\tau$-estimator. This especially holds for SA and TS, since the objective values 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 as in (2.4). In case of TS, this modification has the disadvantage that one cannot 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 for SA and TS in our simulation study. For RR only the full
option will be studied, but we will use the “trick” described next that reduces the number of M-scales that need to be computed.

### 2.3.3 Avoiding the Computation of M-Scales in the $\tau$-Objective Function

In case of RR, the number of required objective function computations is typically less than for SA and TS. Nevertheless, comparing the $\tau$-objective values for each of the $N$ candidates in Step 3 of the fast-$\tau$ algorithm (RR with efficient local improvement) accounts for much of the computational effort. In the analogous fast-S algorithm, one can avoid having to compute the S-objective function for each candidate $\beta$ by considering the following observation (Yohai and Zamar 1991): given two candidates $\beta_1$ and $\beta_2$, in order to have $s_n(\beta_2) < s_n(\beta_1)$ it is necessary and sufficient that

$$
\frac{1}{n} \sum_{i=1}^{n} \rho_1 \left( \frac{y_i - \beta_2^t x_i}{s_n(\beta_1)} \right) < b_1.
$$

(2.5)

It follows that a new candidate $\beta$ can be discarded without computing its associated S-objective value if it violates some condition of type (2.5). For $\tau$-estimators now it can be shown that the following holds: given two candidates $\beta_1$ and $\beta_2$, in order to have $\tau_n(\beta_2) < \tau_n(\beta_1)$ it is necessary (but not sufficient) that either condition (2.5) is fulfilled or that otherwise

$$
\frac{1}{n} \sum_{i=1}^{n} \rho_2 \left( \frac{y_i - \beta_2^t x_i}{s_n(\beta_1)} \right) < \frac{1}{n} \sum_{i=1}^{n} \rho_2 \left( \frac{y_i - \beta_1^t x_i}{s_n(\beta_1)} \right).
$$

(2.6)

Hence, $\beta_2$ will not improve the $\tau$-objective function corresponding to $\beta_1$ if it violates both (2.5) and (2.6). Similarly to the fast-S algorithm, we can use these two conditions to discard several candidates without computing their $\tau$-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-estimators. Numerical experiments show that this number heavily depends on the data. We regard the reduction in computational cost resulting from these conditions as sufficient and do not consider the option of additionally approximating the objective values that need to be computed in the RR algorithm.

### 2.4 Diversification Steps

We may assume that local improvements through IRLS 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, that is, the tabu list). These diversification steps attempt to redirect 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 $\beta$ that
determine the algorithm’s path. A diversification move can then be inserted at some point in the algorithm by letting the next $\beta$ 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 contain a tight group of outliers, the tabu list may not succeed in ridding the subsamples of those outliers, and thus all $\beta$’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 that the RR algorithms can also be enhanced in the same manner. For example, the set of candidates $\beta^{l(j)}$, $j = 1, \ldots, N$ in Step 2 of the fast-τ algorithm can be extended by adding one or more candidates corresponding to (nonrandom) subsamples consisting of observations that on average were badly fitted by $\beta^{l(j)}$, $j = 1, \ldots, N$.

In case of SA, finally, 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 the 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 first present an extensive simulation study which examines whether the directed search approach of SA and TS can outperform RR with regard to minimizing the objective function (1.3). Then we report the results of a second simulation experiment that compared the fast-τ version of RR to the widely used fast-LTS and fast-S algorithms in terms of their statistical properties.

3.1 COMPARISON BETWEEN RANDOM RESAMPLING AND HEURISTIC ALGORITHMS

The values of several tuning parameters have to be set for the algorithms. Our choices were based on our experiments and the recommendations of Woodruff and Rocke (1993). First, for all three algorithms (RR, SA, and TS) 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 the 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 five 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 neighborhood searched in each step). Finally, the size of the tabu list was set at $\kappa = 0.15 n$. 

THE FAST-τ ESTIMATOR FOR REGRESSION

13
Regarding the computational cost reduction as discussed in Section 2.3, we used approximate IRLS steps in all algorithms. Further, for the calculation of the objective values in SA and TS, both approximate M-scale (with \( r = 2 \)) and full-scale computation were tried. For RR we considered only full scale computation and we applied conditions (2.5) and (2.6) to reduce the number of required computations.

The number \( k \) of initial IRLS steps, as well as the number \( t \) of candidates to which IRLS iterations were applied until convergence, are varied throughout the study. In particular, we will first focus on the basic RR, SA, and TS algorithms (without IRLS) and subsequently move to the actual fast-\( \tau \) version of RR and its analogues in SA and TS.

All algorithms have explicit diversification steps implemented as described in Section 2.4. In case of RR, the modification of Olive and Hawkins (2007) suggested by a referee was included, except when comparing the basic versions of the algorithms. Computations were performed in MATLAB. Code for the fast-\( \tau \) algorithm (MATLAB, Octave, and R) is available online at http://www.amstat.org/publications/JCGS.

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 (1.1) 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 \( \beta_0 = 0 \);
- \( \epsilon \) 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 \( \mathbf{d} = (100, 0, \ldots, 0)' \), 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 \( \epsilon \) we generated 500 datasets, on which we applied the three 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 are such that, at the population level, two local minima appear in the \( \tau \)-objective function, one of which is caused by the high-leverage points. For finite samples, however, the \( \tau \)-objective function is naturally affected by sampling variability and the number of local minima typically depends on the ratio \( n/p \). 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 \( \tau \)-objective functions:
• When $p = 2$ and $p = 5$, $n = 100$ is relatively large and $\tau_n(\beta)$ tends to have exactly two minima;
• when $p = 10$, $n = 100$ is relatively small and $\tau_n(\beta)$ often has several local minima;
• when $p = 20$ and $n = 100$ the $\tau$-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 80% ($p = 2$) to 10% ($p = 20$) for $\epsilon = 0.10$, and from 64% to 0.7% for $\epsilon = 0.20$, approximately. Note, however, that obtaining an outlier-free subsample does not guarantee that subsequent IRLS local improvements will converge to the global minimum of $\tau_n(\beta)$, because of the following reasons: (a) an outlier-free subsample may still yield a corresponding $\hat{\beta}$ 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 $\tau$-objective function may correspond to a fit that accommodates the outliers (in these cases the $\tau$-estimator was affected by the contamination).

3.1.1 No Local Improvement ($k = 0, t = 0$)

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

For each algorithm we computed the average attained objective value $\tau_n(\hat{\beta}_n)$ over the 500 samples.

Figure 2 shows, for different values of $N$ (200, 500 and 1500), the ratio of the average 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 six data configurations corresponding to the different outlier slopes $m$. In fact, the ratios between the average $\tau_n$-values were fairly similar for all six 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 the heuristic 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 average objective value attained by the basic RR algorithm for $p = 5, 10, \text{ and } 20$ is depicted by the dash-dotted curves in the upper panels of Figure 5.

3.1.2 Local Improvement of Final Candidates ($k = 0, t = 1$)

Let us now briefly look at what happens if we locally improve the best candidate of each algorithm. That is, we perform IRLS steps until convergence to the best $\beta$ 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 RR, 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 again the ratios of the average $\tau$-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 RR 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 $\tau$-objective function.

### 3.1.3 Efficient Local Improvement ($k = 2$, $t = 5$)

Finally, we introduce initial IRLS steps by choosing $k = 2$ and we set $t = 5$, so that RR now actually represents what we refer to as fast-$\tau$. Also, from now on we apply the modification of Olive and Hawkins (2007), as suggested by a referee (see also Section 2.1).

We would like to note, however, that in this simulation study the influence of the modification was very minor. In particular, the resulting curves below showed no noticeable difference from the curves obtained without the modification (see also Section 3.2). Regarding SA and TS, we consider both the within-search and out-of-search variants for IRLS, as described in Section 2.2.3.

The improvement that the IRLS steps bring compared to 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). The solid lines now correspond to the
current setting. On a side note, these plots also show that, for the outlier configurations considered in this simulation, $N = 200$ seems sufficient for small values of $p$.

The respective ratios between the average $\tau$-objective values for this 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) IRLS options. The plots indicate that RR 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).

### 3.1.4 The Case of a Large Number of Local Minima

When the proportion of outliers was 20%, the basic variants ($k = 0$) of the three algorithms performed similarly as in 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 corresponding plots are shown in Figure 6. Note that for $p = 20$ and $N = 1500$, the RR algorithm is outperformed by some of the SA and TS variants.

This demonstrates that in situations with a large number of local minima, such as our $n = 100$ and $p = 20$ case, directed search heuristic algorithms with appropriately implemented local improvement steps can perform better than RR. Further experiments have indicated that in order for RR to keep up with the best heuristic search algorithms in these situations, the application of local improvement steps should be intensified. That is, we should preferably set either $k > 2$ or $t > 5$ (or both) in the fast-$\tau$ algorithm in case the ratio $n/p$ is small.
3.1.5 Conclusion

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 $\tau_n(\beta)$ was found to depend on the ratio $n/p$ (larger 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-$\tau$ algorithm (i.e., RR with efficient local improvement) computes an approximation for $\tau$-estimators with the same computational complexity as the fast-S algorithm computes an approximation for S-estimators. Fast-$\tau$ 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 IRLS step for $\tau$ involves more calculations than that for $S$. Some computation times for our implementations of fast-$\tau$ and fast-$S$ are listed in Table 1, for different values of $n$ and $p$. Both algorithms use $N = 1500$, $k = 2$ and $t = 5$. 

Figure 4. RR versus SA and TS: efficient local improvement ($k = 2$, $t = 5$); both within- (triangles) and out-of-search (circles) local improvement. Average $\tau$-objective values: ratio with respect to RR; averaged over six data configurations with 10% outliers
Figure 5. Average τ-objective values for RR in case $\epsilon = 0.10$; $n = 100$; $p = 5$ (left), $p = 10$ (middle), and $p = 20$ (right); averaged over six data configurations; dash-dotted: basic version ($k = 0, t = 0$)—dashed: basic version with local improvement on end result ($k = 0, t = 1$)—solid: efficient local improvement ($k = 2, t = 5$); lower panels are zoomed.

The times (in seconds) are averaged over 20 samples for each of the six data configurations 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 IRLS convergence points in an application of the RR algorithm with exhaustive local improvement (i.e., RR with $t = N$). 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.

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

<table>
<thead>
<tr>
<th></th>
<th>$p = 10$</th>
<th>$p = 20$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>time (s)</td>
<td>fast-τ</td>
<td>2.48</td>
</tr>
<tr>
<td></td>
<td>fast-S</td>
<td>1.25</td>
</tr>
<tr>
<td>#minima</td>
<td>τ</td>
<td>5.4</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>110.1</td>
</tr>
</tbody>
</table>
The extent to which this impacts the optimal choice of $N$ and the other parameters is the subject of further research.

3.2 Statistical Properties of Approximating Algorithms for Scale-Based Robust Regression Estimators

We conducted a second numerical experiment in which we compared the statistical properties of the approximations obtained by RR algorithms for LTS, S-, and $\tau$-regression estimators. We chose these estimators because they have good theoretical properties and do not require an initial estimate of residual scale. All three estimators were tuned to have a theoretical 50% breakdown point and we used the optimal $\rho$-function for the S-estimator.

The RR algorithms for approximating LTS and S were implemented completely analogously to the algorithm for $\tau$ described in Section 2.1 and we used the same cost reduction techniques and diversification steps as before. We set $N = 1500$, $k = 2$, and $t = 5$ and we will refer to the algorithms as fast-LTS, fast-S, and fast-$\tau$, respectively. Each algorithm was performed both with and without the Olive and Hawkins (2007) candidates, in order to verify whether the addition of these candidates has an impact on the performance of the algorithms.

We considered 1000 samples of sizes $n = 100$ and 200, with $p = 2, 5, 10, \text{ and } 20$ covariates. We generated observations as in the previous simulation study (see p. 14), where now the outliers have a uniformly distributed slope between 1 and 5. In other words, for each of the $\epsilon 100\%$ outliers we generated $m \sim \mathcal{U}(1, 5)$ and then $y_i | m \sim \mathcal{N}(m 100, 0.1^2)$, $i = 1, \ldots, n$. This scheme allows us to naturally obtain average performances over different outlier configurations ranging from small to moderate contamination slopes.

For each sample and each approximated estimator we recorded the estimated vector of regression coefficients $\hat{\beta}_n$ and the number of correctly and wrongly identified outliers. An
Figure 7. Root mean squared errors for 1000 random samples of size \( n = 100 \) and \( p = 2, 5, 10, \) and \( 20 \). The solid line is for fast-\( \tau \), the dashed line for fast-LTS and the dotted line for fast-S. The fast-\( \tau \) estimator dominates the other estimators for \( p = 5, 10, \) and \( 20 \).

Our first observation is that in our simulation setup where \( \beta_0 = 0 \) the inclusion of the two additional candidates proposed in Olive and Hawkins (2007) to the fast-\( \tau \) algorithm made a small difference in RMSE in a few cases, the largest being for \( \epsilon = 0.10, n = 200, \) and \( p = 20 \), where the RMSEs for the fast-\( \tau \) algorithm with and without the additional candidates were 0.539 and 0.544, respectively.

Figures 7 and 8 display the RMSEs for \( n = 100 \) and \( n = 200 \), respectively. Note that the curves corresponding to the algorithms with and without the two additional candidates cannot be distinguished in the plot, and thus we only use one line for each approximated estimator. As expected, we see that the RMSE of fast-\( \tau \) dominates that of fast-LTS and
fast-S when the sample size is moderate to small relative to the number of covariates. This can be explained by the higher efficiency of the \( \tau \)-estimator relative to the LTS and S-regression estimators.

A similar conclusion can be drawn from the results on outlier detection. More specifically, fast-\( \tau \) had a slightly better number of correctly identified outliers than fast-S for sample sizes relatively small compared to the number of covariates. The number of incorrectly labeled outliers was consistently smaller for fast-\( \tau \). Fast-LTS never dominated, but its performance in terms of detecting outliers was competitive with respect to that of the fast-\( \tau \) and fast-S approximations. For example, for \( \epsilon = 0.10 \), \( n = 100 \), and \( p = 2 \) the average number of correctly identified outliers by the fast-LTS, -S and -\( \tau \) estimators was 9.70, 9.83 and 9.72, while the number of falsely labeled outliers was 0.31, 0.22 and 0.03, respectively. For \( \epsilon = 0.20 \), \( n = 200 \) and \( p = 20 \) we have an average of 24.5, 29.8, and 30.4 correctly identified outliers by fast-LTS, -S, and -\( \tau \), and 11.8, 12.6, and 3.8 falsely labeled outliers, respectively. Complete results of this numerical experiment can be found online at http://www.amstat.org/publications/JCGS.
4. CONCLUSION

In this article we considered the problem of approximating \( \tau \)-estimators for linear regression. These estimators have very good robustness and efficiency properties and it is thus of interest to have good approximating algorithms for them.

By combining random resampling with local IRLS improvements we constructed the so-called fast-\( \tau \) 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, also with local improvement steps. We concluded from extensive simulations that random resampling is inferior to both simulated annealing and tabu search when local improvements are not applied. However, random resampling becomes much more efficient and is generally able to outperform its competitors when IRLS steps are used, even when such local improvement steps are incorporated into the heuristic algorithms as well.

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.

Our findings about the relative performance of random resampling and its fast-\( \tau \) version compared to directed search algorithms are presumably also relevant for the case of approximating the LTS and S-estimators. Specifically, recall that \( \tau \)-estimators reduce to S-estimators when \( \rho_1 = \rho_2 \) in (1.3) and (1.4). Moreover, in our experience the objective functions of S- and \( \tau \)-estimators are generally very similar.

Note that the extent to which the good robustness and efficiency properties of the \( \tau \)-estimator are inherited by the various approximating algorithms has yet to be studied in detail. Although this interesting question is beyond the scope of this article, our extensive numerical studies comparing the fast-\( \tau \), fast-LTS, and fast-S algorithms confirmed that the higher efficiency of the \( \tau \)-estimator translates into smaller RMSE and good outlier detection properties for the fast-\( \tau \) for moderate sample sizes. This suggests that the statistical properties of the approximation given by the fast-\( \tau \) algorithm are very close to that of the \( \tau \)-estimator.

Finally, the fast-\( \tau \) algorithm, like fast-LTS and 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-\( \tau \), namely an initial estimate of \( \beta_0 \) which would be reasonably robust in practice and fast to compute for any \( p \). Such an estimator was proposed by Pena and Yohai (1999).

ACKNOWLEDGMENTS

This research has been supported by Discovery Grants from the Natural Sciences and Engineering Research Council of Canada. The second author has been partially supported by a postdoctoral fellowship of the Pacific Institute for the Mathematical Sciences.

[Received August 2006. Revised January 2008.]
REFERENCES


