New Constrained ILS Reduction

This document describes a new reduction strategy for the constrained integer least squares problem,

$$\begin{aligned}&\min_{x\epsilon X} \|Ax - y\|_2 \\ &= \min_{x\epsilon X} \left\|Rx - Q^T y\right\|_2 \\ &= \min_{x\epsilon X} \|Rx - \hat{y}\|_2\end{aligned}$$

This reduction uses ideas from Chang and Han's "Solving Box-Constrained Integer Least Squares Problems" and Su and Wassel's "A New Ordering for Efficient Sphere Decoding". First both algorithms will be briefly described, then it will be proven that both of these algorithms give the same result. Chang's algorithm is easy to understand and implement but much less time-efficient than Su's when implemented in a language like MatLab where permutation operations on a matrix involve copying data. Su's algorithm is difficult to understand but does not involve expensive memory operations. Comparing the flop counts of the two algorithms, Chang's algorithm gives a flop count about 1/3rd that of Su's. The new algorithm is easy to implement and understand while providing runtime performance very close to that of Su's algorithm when implemented in MatLab and a flop count less than either of the original algorithms.

In the first step of Chang's algorithm, for $i\epsilon 1...n$ we interchange columns i and n in the matrix R, then return R to upper-triangular form with a series of Givens rotations. We then compute $a_i^c = \arg\min_{x\epsilon X} |r_{n,n}x_n - \hat{y_n}| = \lfloor \hat{y_n}/r_{n,n} \rceil_X$ (the subscript X denotes rounding to the nearest value in the set X while the subscripts i refer to the column that has currently been swapped to the n^{th} position and superscript c denotes Changs algorithm). Also, $b_i^c = a_i^c \pm 1$ so that b_i^c is the second closest integer in X to $\hat{y_n}/r_{n,n}$. Finally, we compute $dist_i^c = |r_{n,n}b_i^c - \hat{y_n}|$ which represents the partial residual given when x_n is fixed to b_i^c and column i is chosen to be the n^{th} column in the matrix R. The final value for the n^{th} column is chosen to be the column i that maximizes $dist_i^c$. The algorithm then applies the Givens rotations that were used to restore R to upper triangular to $\hat{y}$ and sets $\hat{y}_{1:n-1} = \hat{y}_{1:n-1} - r_{1:n-1,n}a_i^c$ to impose the constraint $x_n = a_i^c$. We then continue to work on the subproblem, $\|R_{1:n-1,1:n-1}x_{1:n-1} - \hat{y}_{1:n-1}\|_2$

The first step of Su's algorithm is much the same. Let $G = (A^{-1})^T$ and g_i references the i^{th} column of G. Note that $\forall j \neq i \quad g_i \perp a_j$. In the first iteration, for each $i\epsilon 1...n$ we compute $a_i^s = \arg\min_{x\epsilon X} \left|y^T g_i - x\right| = \left\lfloor y^T g_i \right\rceil_X$. Also, $b_i^s = a_i^s \pm 1$ so that b_i^s is the second closest integer in X to $y^T g_i$. Then compute $dist_i^s = \left\|\frac{g_i g_i^T}{g_i^T g_i}(y - a_i b_i^s)\right\|_2$. The n^{th} column is chosen to be the one that maximizes $dist_i^s$. After the n^{th} column is determined, we set $y = (I - \frac{g_i g_i^T}{g_i^T g_i})(y - a_i a_i)$ this projects and shifts y into the space spanned by the remaining columns of A by removing the part of y that is orthogonal to those columns (g_i). The shift imposes the constraint $x_n = a_i^s$. Since we are effectively removing the i^{th} column of A, we must update the inverse by setting $\forall j \neq i \quad g_j = (I - \frac{g_i g_i^T}{g_i^T g_i})g_j$. This gives the pseudoinverse of A with column i removed. For proof of this, see "Representations for the Generalized Inverse of a Partitioned Matrix".

In order to prove Chang and Su's algorithms produce the same set of permutations, it will suffice to prove that $a_i^s = a_i^c, b_i^s = b_i^c, dist_i^s = dist_i^c$ and that the subproblems produced for the second step of each algorithm are equivalent. Proving $a_i^s = a_i^c$ is not difficult. Let J be the product of the Givens rotations used to return R to upper triangular after a permutation P is applied. Since $\left\|JRPx - JQ^T y\right\|_2 = \left\|RPx - Q^T y\right\|_2$, we know

that when Chang's algorithm swaps column i with column n and returns R to upper triangular through Givens rotations, the only effect this has on the real LS solution x is that elements i and n are swapped. Therefore a_c^i is just the i^{th} element of the real least squares solution rounded to the nearest integer in X. We can compute this directly by $a_i^c = \lfloor (A^{-1}y)_i^T \rceil = \lfloor y^T g_i \rceil$. Which is the same as Su's method. Obviously once a_i^c is known b_i^c can be easily found. To compute $dist_i^c$ directly, we can use the formula from Su's paper, $dist_i^s = \left\| \frac{g_i g_i^T}{g_i^T g_i}(y - a_i b_i^c) \right\|_2$. The following proof shows that this is equivalent to the method Chang uses to compute the residual.

$$\begin{aligned} dist_i^s &= \left\| \frac{g_i g_i^T}{g_i^T g_i}(y - a_i b_i) \right\|_2 \\ &= \left\| \frac{g_i g_i^T}{g_i^T g_i} y - \frac{g_i}{g_i^T g_i} b_i \right\|_2 \\ &= \left\| \frac{g_i}{g_i^T g_i}(A^{-1}y)_i - \frac{g_i}{g_i^T g_i} b_i \right\|_2 \end{aligned}$$

Let $\bar{R} = JRP$ denote R after applying permutation matrix P to swap columns i and n, and the product of Givens rotations J to restore R to upper triangular. Then we have,

$$\begin{aligned} &= \left\| \frac{g_i}{g_i^T g_i}\left(\frac{\hat{y}_n}{\bar{r}_{n,n}} - b_i\right) \right\|_2 \\ &= \left\| \frac{g_i}{g_i^T g_i} \right\|_2 \left| \frac{\hat{y}_n - \bar{r}_{n,n} b_i}{\bar{r}_{n,n}} \right| \\ &= \frac{1}{\|g_i\|_2} \left| \frac{\hat{y}_n - \bar{r}_{n,n} b_i}{\bar{r}_{n,n}} \right| \end{aligned}$$

Now we must show $\frac{1}{\|g_i\|_2} = \bar{r}_{n,n}$. It is easy to see from the definition of $G = (A^\dagger)^T$ that:

$$\begin{aligned} \|g_i\|_2 &= \left\| (R(R^T R)^{-1})_i \right\|_2 \\ &= \left\| \left(\begin{bmatrix} R \\ 0 \end{bmatrix} \left(\begin{bmatrix} R^T & 0 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \right)^{-1} \right)_i \right\| \\ &= \left\| (R^{-T})_i \right\|_2 \\ &= \left\| J(R^{-T})_i \right\|_2 \\ &= \left\| (\bar{R}^{-T})_n \right\|_2 \end{aligned}$$

Since $\bar{R}^{-1}\bar{R} = I$ and R is upper-triangular, it must be true that,

$$\begin{aligned} \left\| (\bar{R}^{-T})_n \right\|_2 &= \left| \frac{1}{r_{n,n}} \right| \\ &= \|g_i\|_2 \end{aligned}$$

Therefore, $dist_i^s = |\hat{y}_n - \bar{r}_{n,n} b_i|$

Next we must prove that the subproblems in the second step of each algorithm are equivalent. Originally we have $\hat{y} = Q^T y$. Let $\hat{y}^{(2)} \epsilon \mathbb{R}^{n-1}$ denote $\hat{y}$ in the second step of Changs algorithm and $y^{(2)} \epsilon \mathbb{R}^n$ denote y in the

second step of Su's. We must show that $\hat{y}^{(2)} = (Q^T y^{(2)})_{1:n-1}$ and $(Q^T y^{(2)})_n = 0$.

$$\begin{aligned}(Q^T y^{(2)})_i &= q_i^T (I - \frac{g_n g_n^T}{g_n^T g_n})(y - a_n x_n) \\ &= (q_i^T - q_i^T \frac{g_n g_n^T}{g_n^T g_n})(y - a_n x_n)\end{aligned}$$

This can be simplified by observing that $Q^T G$ is lower triangular.

$$\begin{aligned}Q^T G &= Q^T (QR)^{-T} \\ &= R^{-T}\end{aligned}$$

So $q_i^T g_n = 0$ for $i = 1..n-1$. When $i = n$, we have $g_n = q_n R_{n,n}^{-T}$. Because Q is orthogonal, this means $q_n = \frac{g_n}{\|g_n\|}$ which implies $(q_n^T - q_n^T \frac{g_n g_n^T}{g_n^T g_n}) = 0$. So now for $i \neq n$ we have $(q_i^T - q_i^T \frac{g_n g_n^T}{g_n^T g_n})(y - a_n x_n) = q_i^T y_n - q_i^T a_n x_n$. And since $R = Q^T A$ implies $r_{i,j} = q_i^T a_j$, we have

$$\begin{aligned}q_i^T y - q_i^T a_n x_n &= q_i^T y - r_{i,n} x_n \\ &= \hat{y}^{(2)}\end{aligned}$$

Now we know $\hat{y}^{(2)} = (Q^T y^{(2)})_{1:n-1}$, and from "Representations for the Generalized Inverse of a Partitioned Matrix" we know that the updated G for the second step of the algorithm is the pseudoinverse of A with the column that we swapped for the n^{th} removed. This implies that the subproblems in the second step of each algorithm are equivalent.

Now that we know the two algorithms are equivalent, we can take the best parts from both and combine them to form a new algorithm. The main cost in Chang's algorithm is to interchange the columns of R and return it to upper-triangular form. At the k^{th} step of the algorithm we must do this k times. We can avoid all but one of these column interchanges by computing a_i^c, b_i^c and $dist_i^c$ directly. Let $L = R^{-T}$. Then we know from the previous proof and the lower triangular structure of L we can compute $a_i^c = \left\lfloor \hat{y}_{i:n}^T l_i \right\rceil_X$ and b_i^c to be the second closest integer to $\hat{y}^T l_i$. Similarly, we can compute $dist_i^c = \left\| \frac{l_{i:n,i} l_{i:n,i}^T}{l_{i:n,i}^T l_{i:n,i}} (\hat{y_{i:n}} - r_{i:n,i} b_i^c) \right\|_2$

After the column that maximizes $dist_i^c$ is found, like in Chang's algorithm, we swap that column with the n^{th} and use Givens rotations to restore the upper-triangular structure of R. We also apply the same Givens rotations to the target vector y. To impose the constraint $x_n = a_i^c$ we set $\hat{y}_{1:n-1} = \hat{y}_{1:n-1} - r_{1:n-1,n} a_i^c$. In addition, we must also update the inverse matrix L, this is very easy. Suppose P was the permutation matrix applied to R and J denotes the product of Givens rotations to restore it to upper triangular. So, $R = JRP$ and set $L = R^{-T} = (JRP)^{-T} = JLP$. After this, as in Changs algorithm, we continue to work on the subproblem of size $n-1$. The advantage of using Chang's idea for this second part of the algorithm is that the flop count will be lower since at each step we reduce the problem size, and the operations are less costly than Su's projections. Also the algebra is easier to understand without considering the geometry of the projections.