Supplement to "Eliminating redundancy among protein sequences using submodular optimization"

December 15, 2017

1 Supplementary Note: Alternative subset quality functions

In this section we consider two other potential objective functions that could be used to measure redundancy (as opposed to sum-redudancy).

First, we define the *max-redundancy* function as

$$f_{\text{max-redundancy}}(A) \triangleq \kappa_{\max} - \sum_{r_1 \in R} \max_{r_2 \in R, r_2 \neq r_1} \text{sim}(r_1, r_2). \tag{1}$$

where $\kappa_{\max} = \sum_{r_1 \in S} \max_{r_2 \in S, r_2 \neq r_1} \text{sim}(r_1, r_2)$. The max-redundancy function is very similar to sum-redundancy, except that it penalizes only the most similar neighbor for each sequence in R instead of all neighbors. The max-redundancy is submodular and normalized, but not monotone non-decreasing and (like above) is monotone non-increasing. We chose to focus on sum-redundancy over max-redundancy because optimizing sum-redundancy produced better results according to SCOPe (Supplementary Figure 7)

Second, we define the *independent-set* function as

$$f_{\text{independent-set}}(R) \triangleq \kappa_{\text{indset}} - \max_{r_1, r_2 \in R, r_2 \neq r_1} \text{sim}(r_1, r_2). \tag{2}$$

where $\kappa_{\text{indset}} = \max_{r_1, r_2 \in S, r_2 \neq r_1} \text{sim}(r_1, r_2)$. The independent set function penalizes according to the most-similar pair among all chosen sequences. The threshold algorithm was originally motivated as optimizing this function. Unfortunately, this function is not submodular, so the optimization methods we apply here do not apply to it.

Supplementary Figures/Algorithms

Algorithm 1 Threshold algorithm. S is the set of sequences and $\text{sim} : S \times S \to \mathbb{R}$ is a similarity function over S. $N : S \to 2^S$ is the set of neighbors of s with nonzero similarity. τ is a threshold determining the size of the returned subset (low τ results in a small subset).

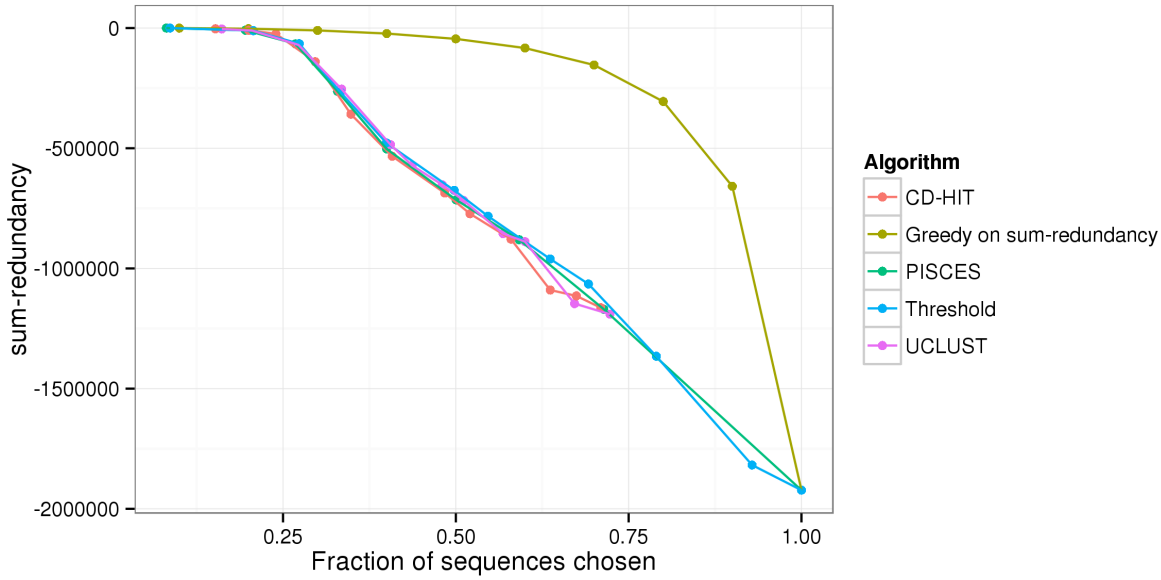
function THRESHOLD(S, τ)
    Order S into a list $(s_1, s_2, \ldots, s_n)$ in decreasing order of sequence length.
    $R \leftarrow \{s_1\}$
    for $i = 2 \ldots N$ **do**
        $m \leftarrow \max\{\text{sim}(s_i, s') : s' \in R \cap N(v_i)\}$
        if $m < \tau$ **then**
            $R \leftarrow R \cup \{v_i\}$
        end if
    end for
    return R
end function

Algorithm 2 Greedy submodular maximization algorithm for optimizing a function $f(R)$. S is the set of sequences, and k is the size of subset to return.

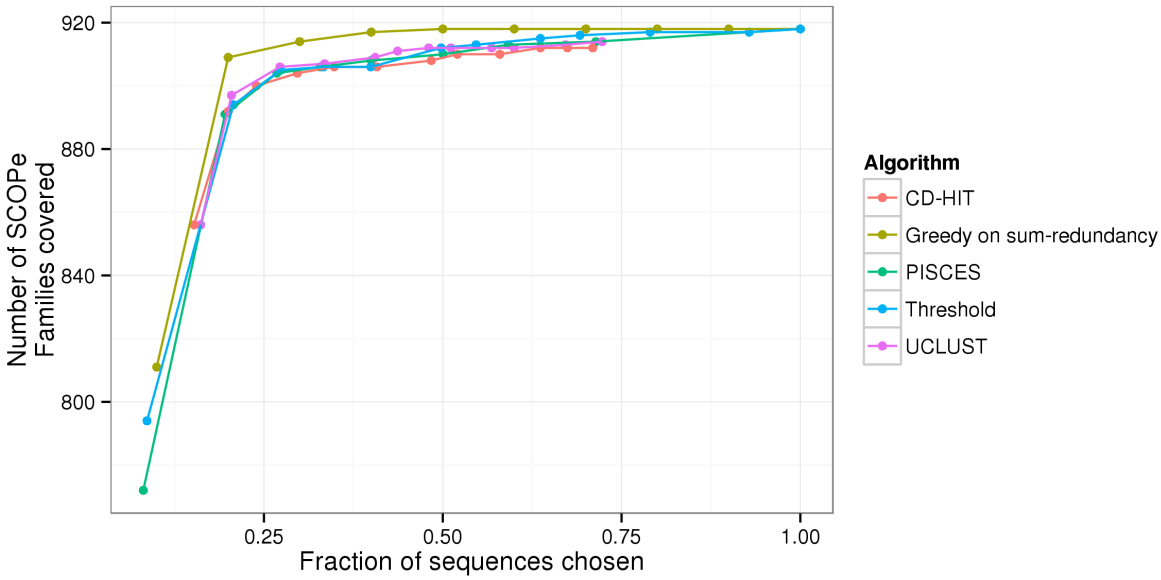
function MONOTONE_GREEDY(S, f, k)
    $R \leftarrow \{\}$
    for $1 \ldots k$ **do**
        for $s \in S \setminus R$ **do** ▷ Can be accelerated by considering only certain $s \in S$ (Methods).
            $\delta[s] \leftarrow f(R \cup \{s\}) - f(R)$
        end for
        $s^* \leftarrow \text{argmax}_{s \in S \setminus R}\ \delta[s]$
        $R \leftarrow R \cup \{s^*\}$
    end for
    return R
end function

Algorithm 3 BidirectionalGreedy maximization algorithm for optimizing a function $f(R)$. S is the set of sequences.

function BIDIRECTIONAL_GREEDY(S, f)
  $A \leftarrow \{\}, B \leftarrow V$
  for $s \in S$ **do** ▷ Arbitrary order
    $\delta_A \leftarrow \max(0, f(A \cup \{s\}) - f(A))$
    $\delta_B \leftarrow \max(0, f(B \setminus \{s\}) - f(B))$
    if $\delta_A = \delta_B = 0$ **then**
      $\mathrm{p} = 1/2$
    else
      $\mathrm{p} = \delta_A/(\delta_A + \delta_B)$
    end if
    if $\mathrm{random}(0, 1) \leq p$ **then**
      $A \leftarrow A \cup \{s\}$
    else
      $B \leftarrow B \setminus \{s\}$
    end if
  end for
  return A (or B, equivalently)
end function

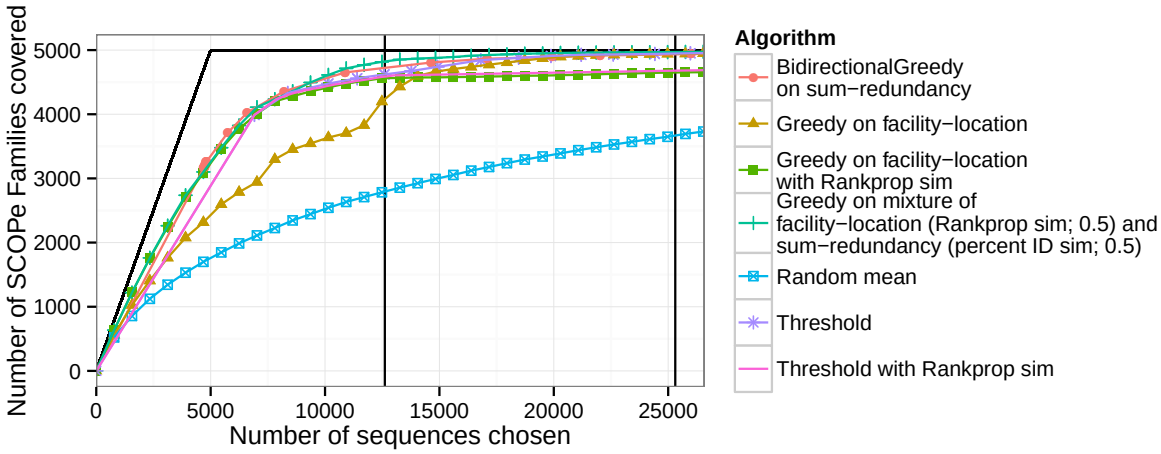


(a)

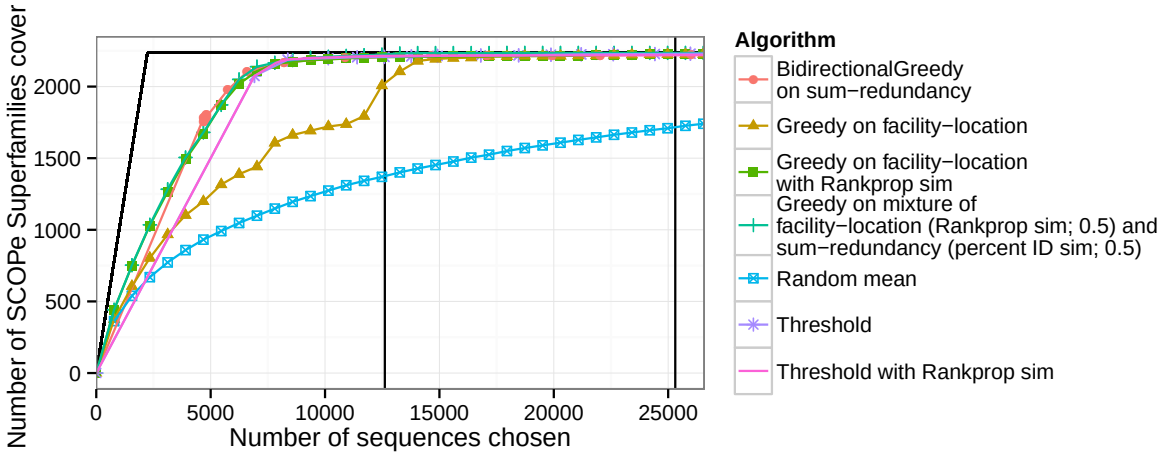


(b)

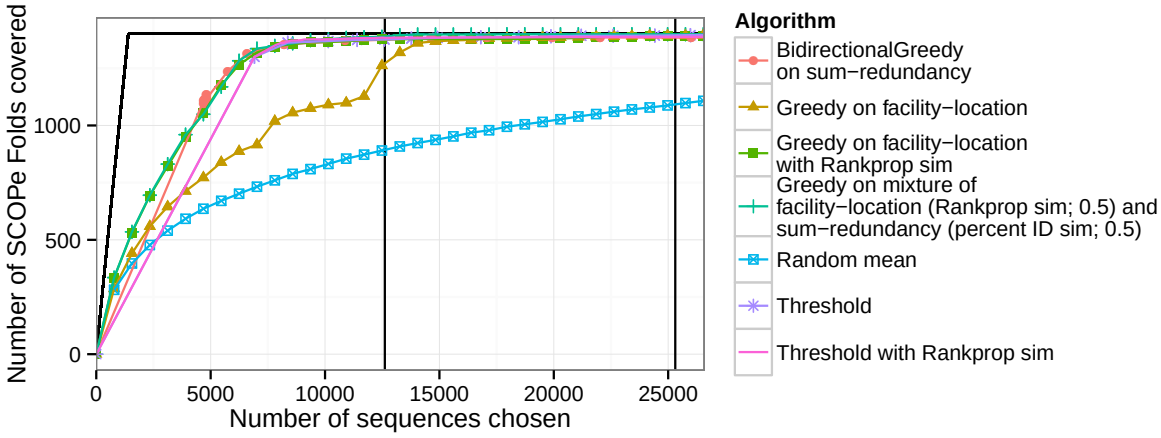
Figure 1: Comparison of methods based on the THRESHOLD algorithm on development set.



(a)

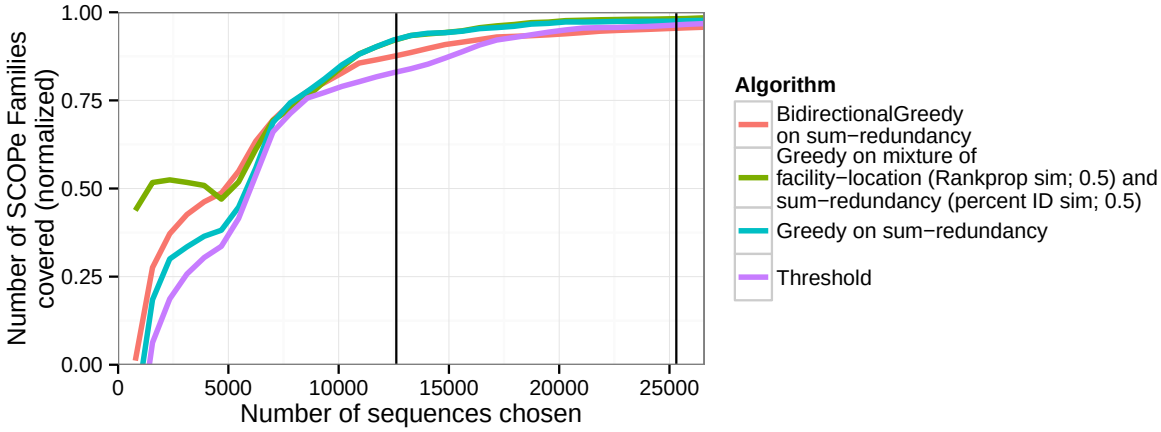


(b)

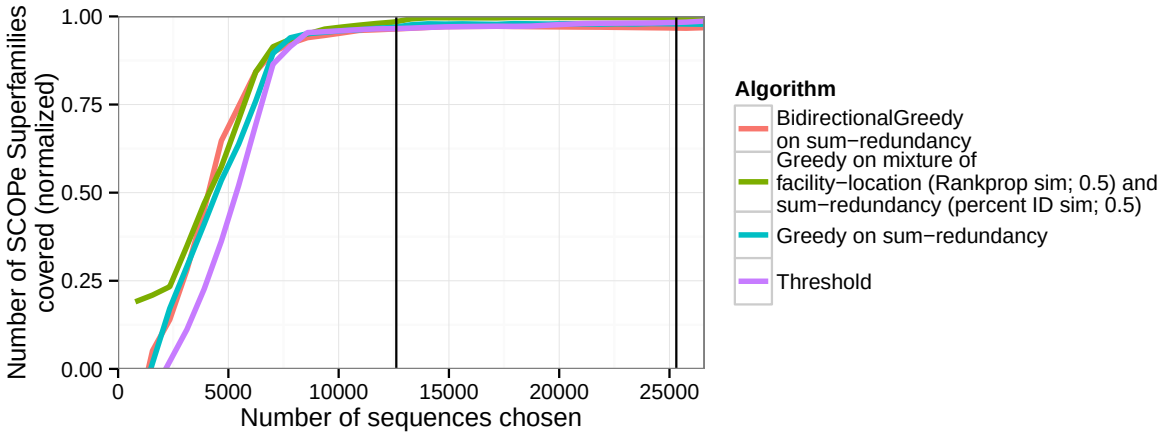


(c)

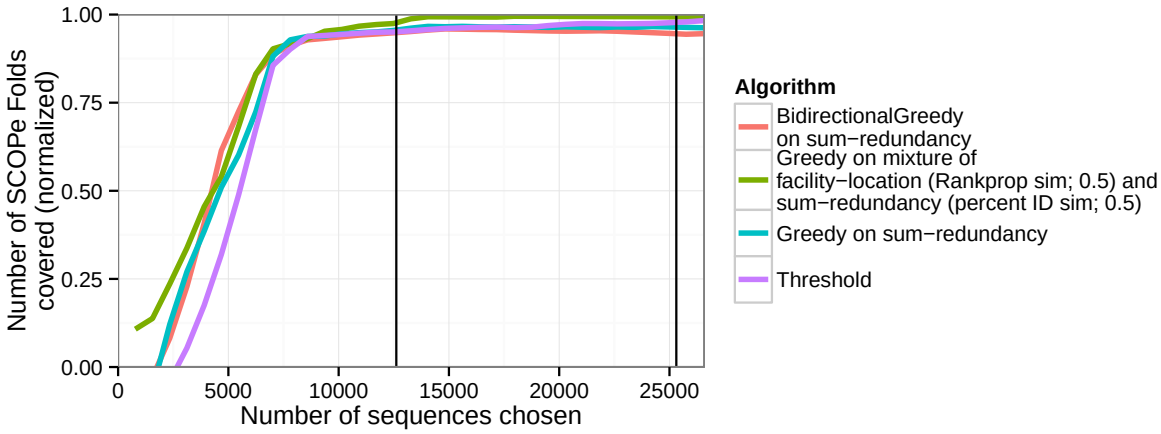
Figure 2: Coverage of SCOPe categories: (A) Families; (B) Superfamilies, and (C) Folds. Horizontal axis indicates size of representative set. Vertical axis indicates number of SCOPe categories with at least one representative. Color and point type indicates choice method. Black curve indicates maximum possible performance. Vertical black lines indicate subset sizes reported by CD-HIT for thresholds of 40% and 90%.



(a)



(b)



(c)

Figure 3: Same as Supplementary Figure 2, but vertical axis is normalized as in Figure 3.

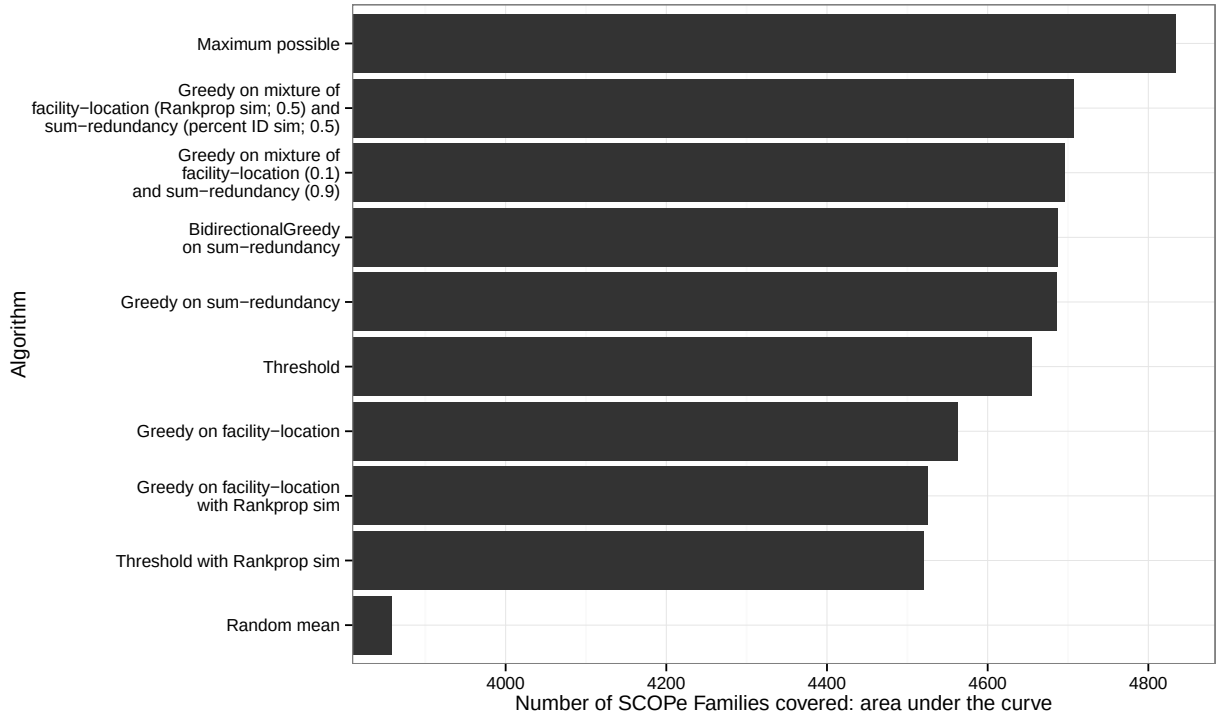


Figure 4: Area under the curve for SCOPe Family coverage. Bars indicate selection algorithms, ordered by their area under the curve. Horizontal axis indicates area under the curve, calculated as follows. Let V be the full set of sequences, $i \in 1 \ldots |V|$ be a subset size, m be a particular method, $A_i^{(m)}$ be the subset of size i chosen by m, and $f(A)$ be the number of SCOPe Families covered by A. Area under the curve $= \frac{1}{|V|} \sum_{i=1}^{|V|} f(A_i^{(m)})$. For subset sizes for which we did not compute a subset, $f()$ is imputed with linear interpolation.

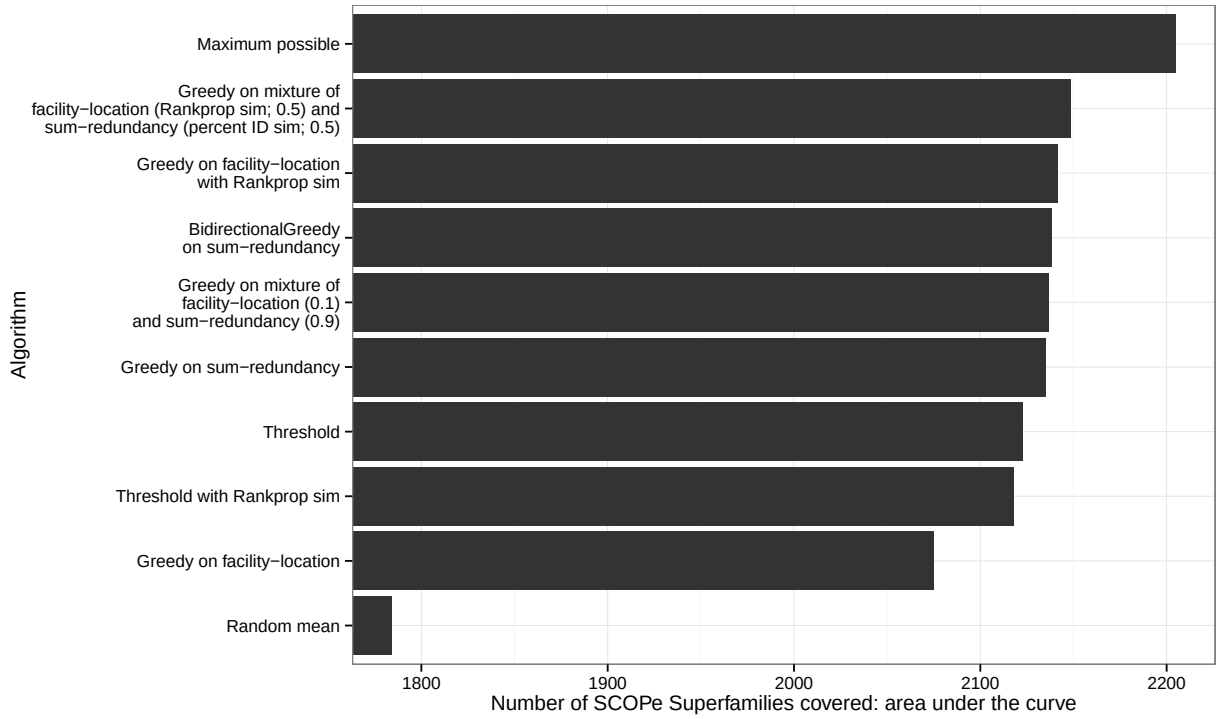


Figure 5: Area under the curve for SCOPe Superfamily coverage. Plot is same as Supplementary Figure 4, except that horizontal axis indicates Superfamily coverage.

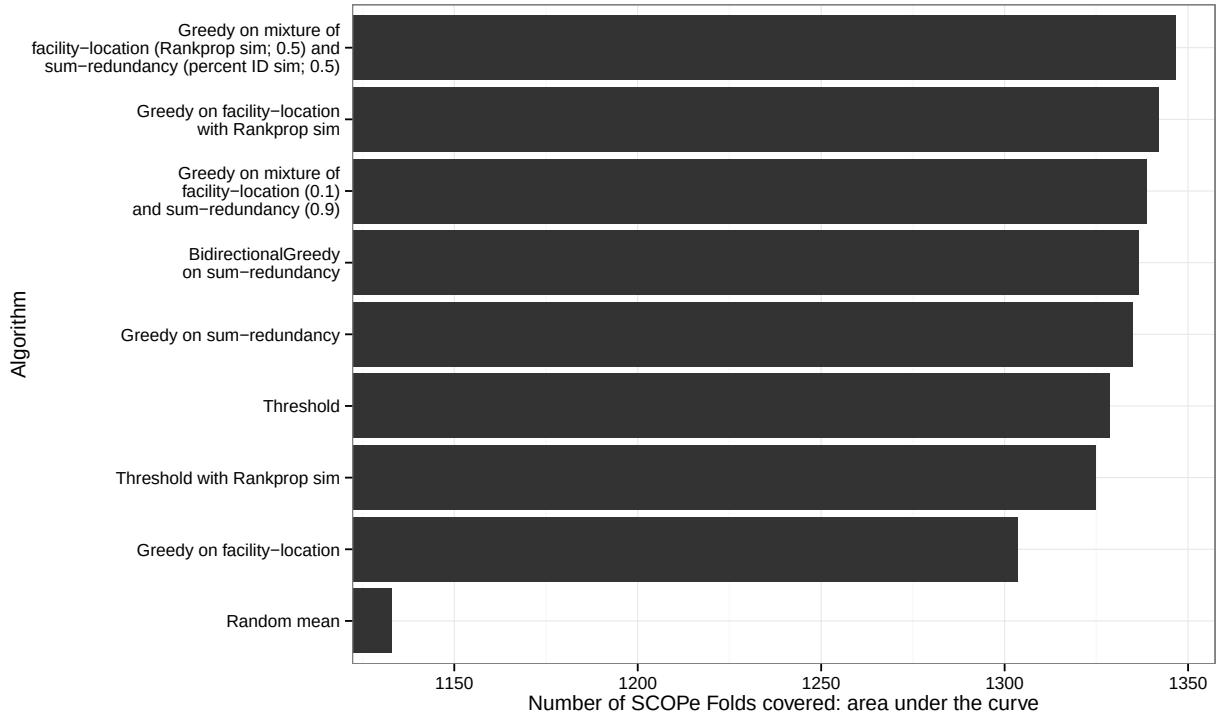


Figure 6: Area under the curve for SCOPe Fold coverage. Plot is same as Supplementary Figure 4, except that horizontal axis indicates Fold coverage.

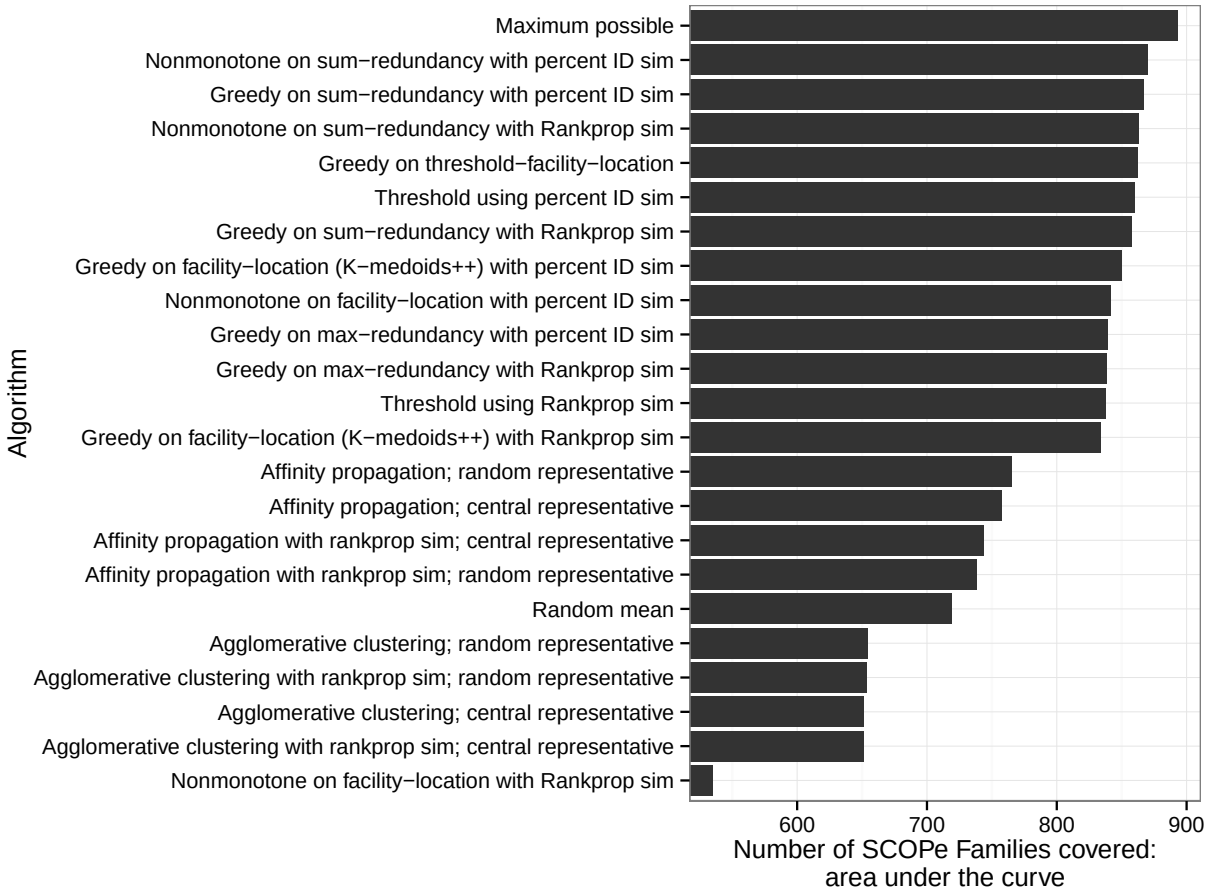


Figure 7: Area under the curve for SCOPe Family coverage on development set. Plot is same as Supplementary Figure 4, except that results are computed on our development set, composed of the subset of the data (∼21%) with the SCOPe Class "All beta proteins".

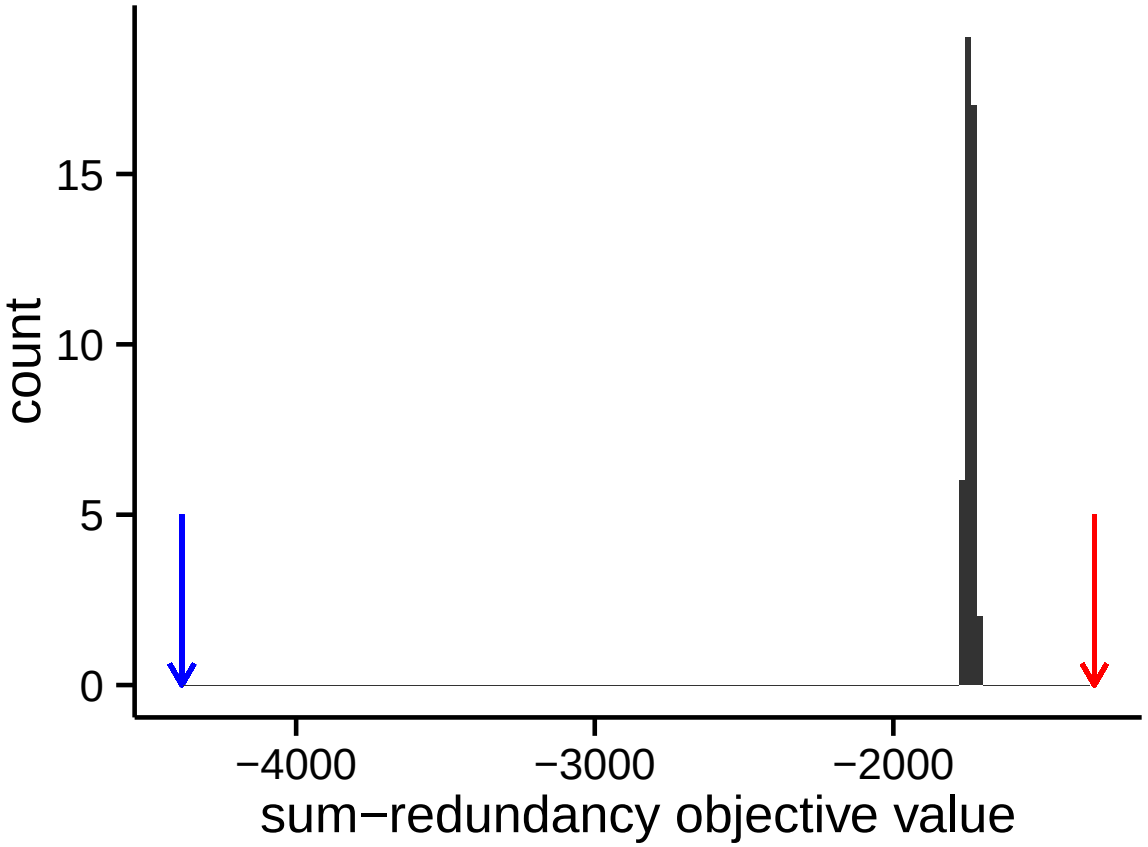


Figure 8: Variance in performance of BidirectionalGreedy. Histogram shows frequency of sum-redundancy objective values achieved over 45 runs of BidirectionalGreedy on the same data set. Blue and red arrows show performance of Threshold and Greedy respectively. Results are from our development set with a target representative set size of 3118 (chosen arbitrarily).

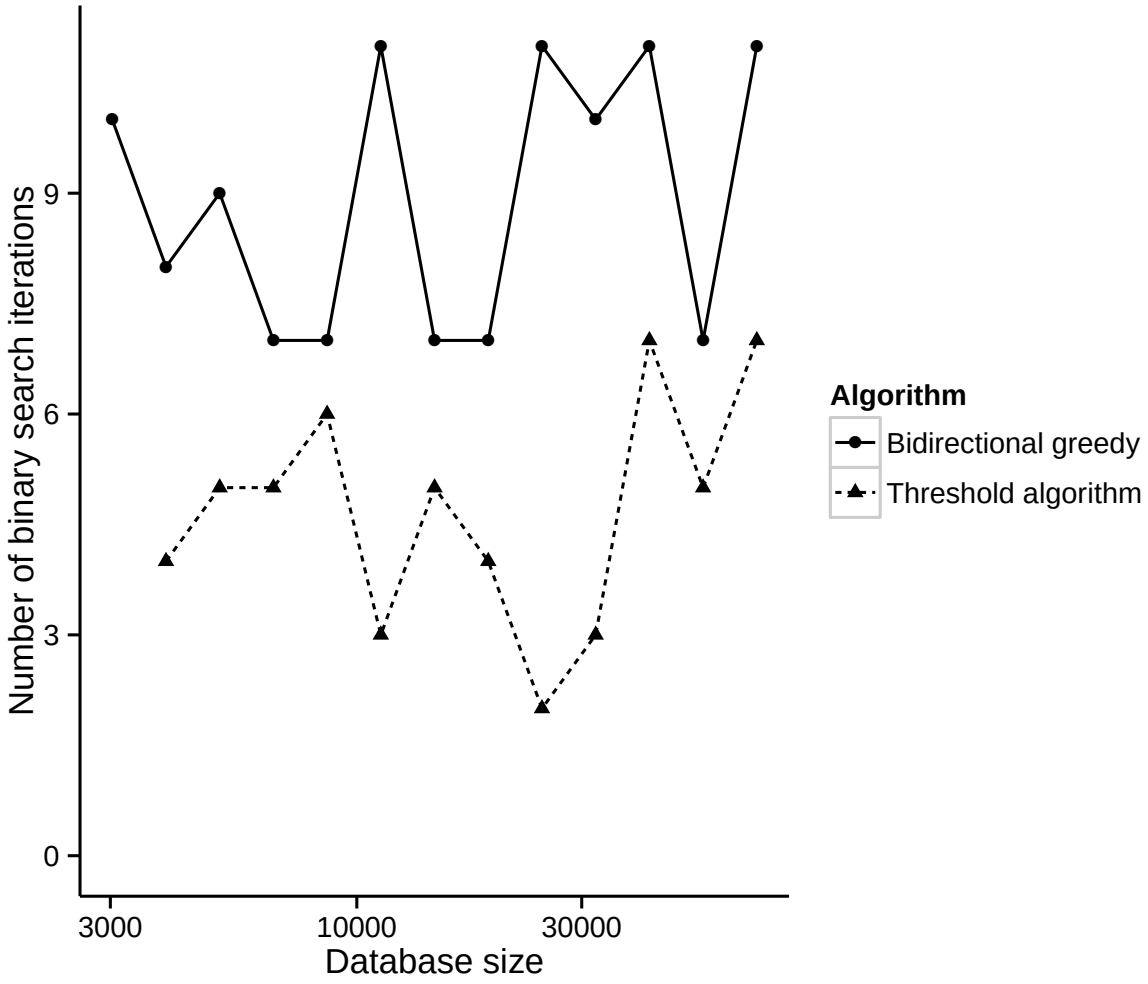


Figure 9: Number of iterations for binary search to converge as a function of database size. Target representative set size was set as 1/2 of database size. We ran binary search iterations until the representative set had size within $\texttt{database\_size} \cdot 0.01$ of the target size.

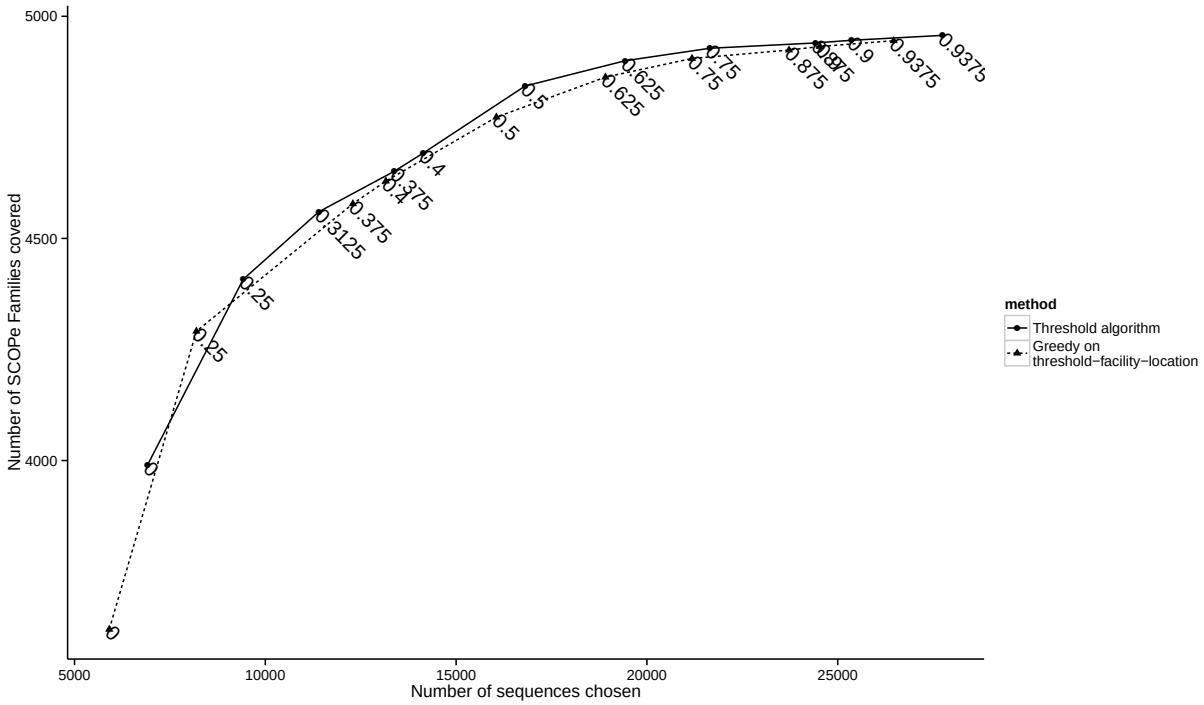


Figure 10: Performance of threshold-facility-location function on SCOPe. Labels on each point correspond to the threshold that produces that set.

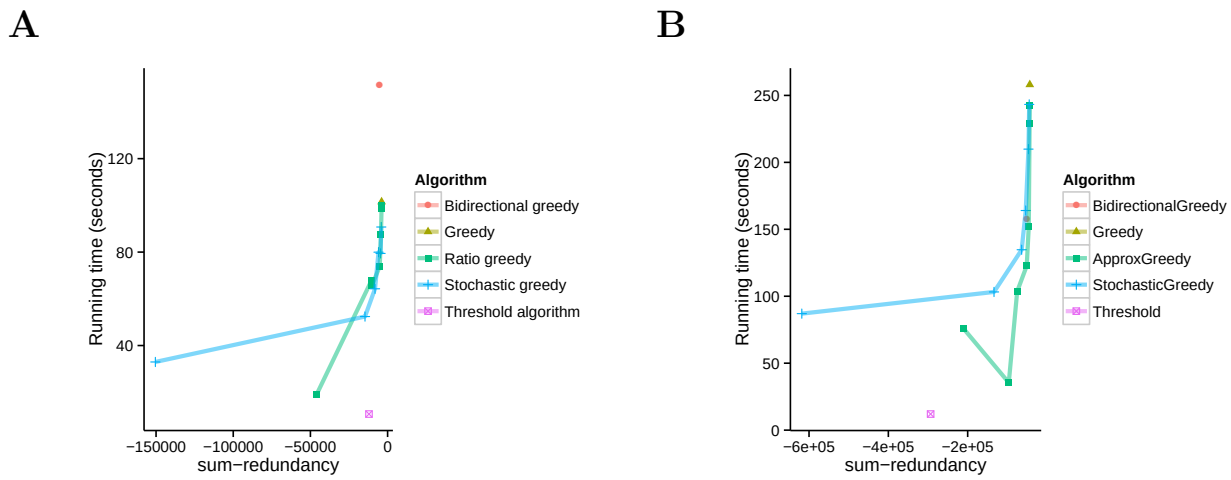


Figure 11: **Relationship between running time and optimization performance.** Vertical axis indicates running time. Horizontal axis indicates the final sum-redundancy objective value achieved. Lines for ApproxGreedy and StochasticGreedy indicate different values of the respective hyperparameters (β for ApproxGreedy and ϵ for StochasticGreedy). All algorithms are applied to optimize sum-redundancy with a percent ID similarity. (A) and (B) show results when choosing subsets of 15% and 30% of the total ground set respectively.

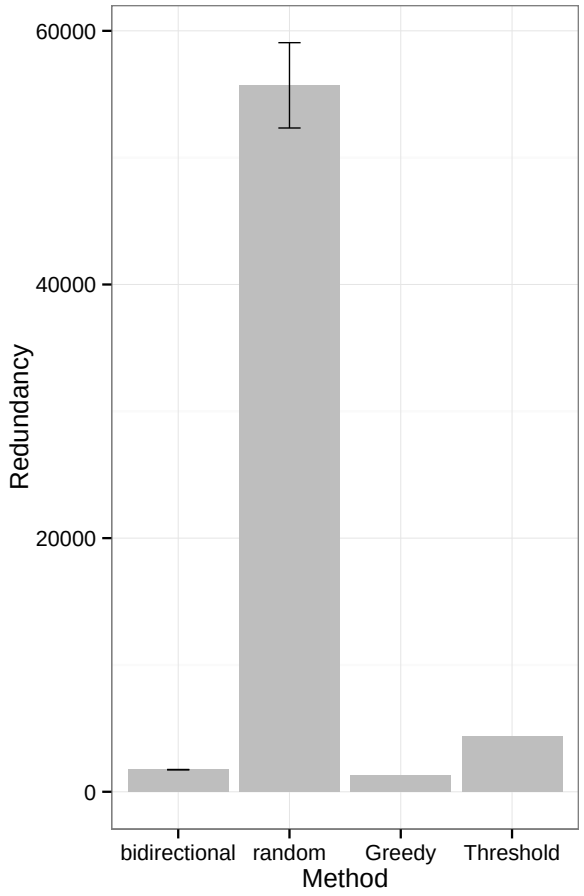


Figure 12: Performance achieved by choosing random subsets. The figure shows the redundancy (sum of percent identity) resulting from 45 runs of RANDOM and BIDIRECTIONAL on the same data set. Vertical axis indicates redundancy, with error bars indicating standard deviation over 45 runs. For comparison, we also include GREEDY and THRESHOLD methods, which do not have variance because they are deteriministic. Results are from our development set with a target representative set size of 3118 (chosen arbitrarily).