Vol. 30 no. 8 2014, pages 1138–1145
doi:10.1093/bioinformatics/btt735

Advance Access publication December 25, 2013

Improving computational efficiency and tractability of protein design using a piecemeal approach. A strategy for parallel and distributed protein design

Derek J. Pitman[1], Christian D. Schenkelberg[1], Yao-Ming Huang[1,2], Frank D. Teets[1], Daniel DiTursi[3] and Christopher Bystroff[1,3,4,*]

[1]Department of Biology, Rensselaer Polytechnic Institute, Troy, NY 12180, [2]Department of Bioengineering and Therapeutic Sciences, University of California San Francisco, San Francisco, CA 94143, [3]Department of Computer Science and [4]Center for Biotechnology and Interdisciplinary Studies, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Associate Editor: Anna Tramontano

ABSTRACT

Motivation: Accuracy in protein design requires a fine-grained rotamer search, multiple backbone conformations, and a detailed energy function, creating a burden in runtime and memory requirements. A design task may be split into manageable pieces in both three-dimensional space and in the rotamer search space to produce small, fast jobs that are easily distributed. However, these jobs must overlap, presenting a problem in resolving conflicting solutions in the overlap regions.

Results: Piecemeal design, in which the design space is split into overlapping regions and rotamer search spaces, accelerates the design process whether jobs are run in series or in parallel. Large jobs that cannot fit in memory were made possible by splitting. Accepting the consensus amino acid selection in conflict regions led to non-optimal choices. Instead, conflicts were resolved using a second pass, in which the split regions were re-combined and designed as one, producing results that were closer to optimal with a minimal increase in runtime over the consensus strategy. Splitting the search space at the rotamer level instead of at the amino acid level further improved the efficiency by reducing the search space in the second pass.

Availability and implementation: Programs for splitting protein design expressions are available at www.bioinfo.rpi.edu/tools/piecemeal.html.

Contact: bystrc@rpi.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on August 29, 2013; revised on December 9, 2013; accepted on December 15, 2013

1 INTRODUCTION

Computational protein design is the process of finding an amino acid sequence that best accommodates a desired protein backbone conformation, with additional optional constraints on the structure. Designed proteins have demonstrated increased binding affinities, increased thermal stability (Dantas *et al*., 2003; Joachimiak *et al*., 2006; Malakauskas and Mayo, 1998) and entirely new functions, such as small molecule binding (Cochran, 2005) or protein binding (Guntas *et al*., 2010; Karanicolas *et al*., 2011; Kortemme *et al*., 2004). A novel fold has been designed (Kuhlman *et al*., 2003), as have proteins with switchable folds (Ambroggio and Kuhlman, 2006), and at least two novel enzymes (Jiang *et al*., 2008; Röthlisberger *et al*., 2008). Computational protein design has been used as a protein engineering tool and as an approach to understanding protein folding, stability and function.

Algorithms for protein design must overcome a massive conformational search space (Samish *et al*., 2011; Street and Mayo, 1999). Simplifying and reducing the space, by fixing the backbone atoms and discretizing the side-chain conformations into rotamer libraries (Dunbrack, 2002; Lovell *et al*., 2000), makes protein design feasible but creates new challenges. Energy calculations tend to be less accurate as the sampling of conformational space becomes more coarse-grained. This translates to a trade-off between design accuracy and runtime.

The complexity of the design problem, defined as the total number of possible sequences in the search space, can be reduced by splitting the search space into smaller tasks. Each task may be solved independently, and the results assembled into a single solution. Splitting the search space introduces the possibility of using a fine-grained search, which in turn enhances accuracy in energy calculations. Splitting has been implemented for the side-chain packing problem using graph theory (Canutescu *et al*., 2003; Krivov *et al*., 2009) and for the dead-end elimination (DEE) algorithm (Desmet *et al*., 1992) by applying a divide-and-conquer approach (Georgiev *et al*., 2006). However, these solutions still encounter a complexity ceiling due to memory requirements. This memory problem was overcome by Moltó *et al*. (2009), who were able to fit a large design into small memory by random splitting of the rotamer search space and stochastic, iterative re-estimation of the rotamer profile. But unfortunately as 'dimensionality reduction' was increased, the quality of the design decreased.

Another option is to design small regions of the target in isolation and to combine those results into the final output. This kind of approach has been done using a sliding window of

*To whom correspondence should be addressed.

tetrapeptide units, resulting in thermophilic versions of the TOP7 protein (Dallüge *et al.*, 2007). Analogous methods have been employed for the inverse RNA folding problem, using hierarchical decomposition (Andronescu *et al.*, 2003) and loop decomposition (Gao *et al.*, 2010) to extend the range of designable structures. However, splitting the design into overlapping three-dimensional (real) spaces downplays the covariant nature of energetic interactions, which may be important for accurate energy calculations. To illustrate covariant interactions, if residue A interacts with residue B, and B interacts with C but not with A, the choice of rotamer at A still does affect the choice of rotamer at C. In this work, we explored the trade-offs between computational efficiency and accuracy in computational protein design.

To evaluate the splitting concept in protein design, we divided real space into overlapping regions defined by a distance cut off from a central residue of a template structure. The sequence search space complexity of the small split design regions was low, and consequently the runtime of the entire process was reduced by almost 5-fold without any loss in design accuracy. Splitting the space into small pieces made it possible to reintroduce fine-grained conformational sampling, thus gaining speed without losing accuracy. Furthermore, the split tasks can be run independently on different processors with no inter-process communication burden.

In recombining the results of the split design tasks, conflicts appear where a residue in the overlapping region is assigned incompatible choices of amino acids. In this work, we explored ways of resolving these conflicts while preserving efficiency and accuracy. The results showed us that a second-pass rotamer search was necessary to resolve these conflicts, that the search space complexity for the second pass was significantly reduced from that of the combined first-pass design spaces and that a conservative approach to reducing the rotamer search space complexity was more accurate than an aggressive approach.

Piecemeal design is a general strategy for making computational protein design more efficient, accurate, and amenable to the distributed computing environment. The method wraps around any computational protein design method with minimal adaptation.

2 METHODS

2.1 Master search space generation

The master search space is a position-dependent amino acid subset that restricts the rotamer selection during protein design. The master search spaces used in this study were derived from a multiple sequence alignment that was generated by running three iterations of Position-Specific Iterated BLAST (Altschul *et al.*, 1997) using an E-value cutoff of 0.01. The multiple sequence alignments were converted into position-specific amino acid probability distributions (profiles), which were augmented with probability-weighted pseudocounts from the BLOSUM62 substitution matrix (Henikoff and Henikoff, 1996). The subset was then defined as all amino acids with $P \geq 0.04$.

2.2 Computational design algorithms and timing

Protein design is a search through the space of side-chain rotamer sequences, subject to the calculated energy. Side-chain rotamer libraries were obtained from the Richardson laboratory (Lovell *et al.*, 2000). The energy function was described previously (Huang and Bystroff, 2013).

Two design algorithms were tested in this study, namely, DEE and Monte Carlo (MC). Both of these methods use a precomputed one-dimensional array of energies for all rotamers at designable positions and a two-dimensional array of energies for all interacting pairs of these rotamers. DEE consists of several related algorithms, all of which eliminate rotamers from further consideration by the application of logical tests using energies. The algorithms used were 'original' DEE (Desmet *et al.*, 1992), Goldstein DEE (Goldstein, 1994), split DEE (Pierce *et al.*, 2000), magic bullet split DEE (Gordon and Mayo, 1998), magic bullet Goldstein DEE applied to rotamer pairings (Gordon and Mayo, 1998), 'original' DEE applied to rotamer pairings (Desmet *et al.*, 1992) and Goldstein DEE applied to rotamer pairings (Goldstein, 1994). Rotamers were eliminated by each of the DEE algorithms in turn until no more rotamers could be eliminated. Then, rotamer pairings were eliminated until no more eliminations were possible. The process was repeated until only one rotamer remained at each position, or until no more rotamers or rotamer pairings could be eliminated. If the minimum energy configuration (MEC), comprising a single rotamer per designable position, was not found, then a simulated annealing MC rotamer search was used to converge on the MEC, starting with all rotamers that have not been eliminated. DEE design tasks were timed on jobs of varying sizes, using wall-clock time on a dedicated cluster.

Alternatively, an MC search was performed using the same starting rotamer search space. An MC move consisted of a random rotamer swap selected from the starting search space or a subspace resulting from splitting. No rotamer elimination was done. Simulated annealing was used to establish convergence, which was defined as the point where 10 000 moves had passed without an accepted unfavorable mutation or where 1000 moves had passed without an accepted favorable mutation. MC design tasks were timed on jobs of varying sizes using wall-clock time to converge on a dedicated machine. The random variability in the MC timings was small, as determined by 10-fold replicate simulations.

The algorithms were implemented in Fortran 90 using MPI (Gabriel *et al.*, 2004), with GNU GCC gfortran v4.4.1 and Open MPI v1.4.2. Jobs were executed using two processes on each of nine machines running 64 bit, dual-core, 2.0 GHz processors with 2 GB of memory.

2.3 The energy function

A physics-based, pairwise energy function was developed to score and evaluate side-chain conformations and was characterized previously for its accuracy in recovering the sequence of known protein structure (Huang and Bystroff, 2013). In short, the energy function is a linear sum of a linear combination of weighted non-bonded pairwise atomic energy terms with cross-terms and correction terms. The component energy terms are as follows:

1. Van der Waals interactions (E_{Rep} and E_{Att})
2. Electrostatic interactions (E_{Elect})
3. Hydrogen bonding energies (E_{Hb} and E_{NonHb})
4. Solvation energies (E_{Solv1} and E_{Solv2})
5. Rotamer intrinsic energies (E_{Rot})
6. Void space energies (E_{Vol})
7. Amino acid reference energies (E_{Ref})

In the simplest version, the energy was a weighted sum of terms, as follows.

$$E_{total} = w_{rep}E_{rep} + w_{att}E_{att} + w_{elect}E_{elect} + w_{Hb}E_{Hb} + w_{NonHb}E_{NonHb} \\ + w_{Solv1}E_{Solv1} + w_{Solv2}E_{Solv2} + w_{rot}E_{rot} + w_{vol}E_{vol}$$

The relative weights were optimized by machine learning. Better results were obtained using cross-terms and amino acid-based reference terms. In this work, we used the final version of this energy function, which included no cross-terms but did include amino acid-based reference terms.

2.4 Profile expansion using simulated melting

The piecemeal approach sometimes produced suboptimal results, not selecting the lowest energy rotamer because it was not included in the search space, or because of 'edge effects', the failure to consider a certain interaction because a neighboring residue is not in the design region. To recover false-negative findings, we introduced a post-processing step called 'simulated melting', which is the algorithmic inverse of simulated annealing (Vanderbilt and Louie, 1984). A sequence profile was initialized to MEC; then MC rotamer swap moves were made with the Metropolis temperature set initially to a very low value ($T = 0.1$). If the search encountered any accepted mutations, the rotamer was added to the profile. The temperature was increased by a factor of 1.5 if no new mutations were accepted in 1000 tries. Mutations were summed to the profile until a specified sequence complexity was reached or until reaching a maximum temperature ($T = 1000$). The result of simulated melting is an expanded sequence profile that retains the MEC.

2.5 Full-length design versus the piecemeal approach

In 'full-length' design, all of the desired designable positions are designed at once, using a complete set of rotamers for each amino acid. In the 'piecemeal' approach, we divide the designable positions into subsets, or pieces, and we optionally split the sequence space (Fig. 1).

Real space was divided into regions surrounding design residues using a 7 Å distance cutoff. Each set of selected positions constitutes one 'piece', and the rotamer set for a piece consists of the rotamers for just those positions as listed in the master search space. All other positions were fixed to their native side-chain conformation.

Sequence space was divided by random elimination of amino acids from the master search space, stopping when a defined number of rotamers was reached. The search space could also be split by randomly eliminating rotamers rather than amino acids. By splitting sequence space at the rotamer level, we gained more control over the job size and complexity. Redundant copies of each downsampled piece were generated to ensure that all rotamers were included in at least one copy.

The result of running DEE or MC on each piece was a single rotamer sequence, the MEC. The resulting MECs were merged and reconciled as follows: at any sequence position that was designed in more than one piece, conflicting rotamer selections were reconciled by either (i) keeping the intersection of the rotamer sets or (ii) keeping the union of the sets. We refer to (i) as the 'aggressive' approach because it eliminates more of the search space and leads to a faster overall runtime. Approach (ii) is termed 'conservative' because it keeps more of the search space at the expense of runtime.

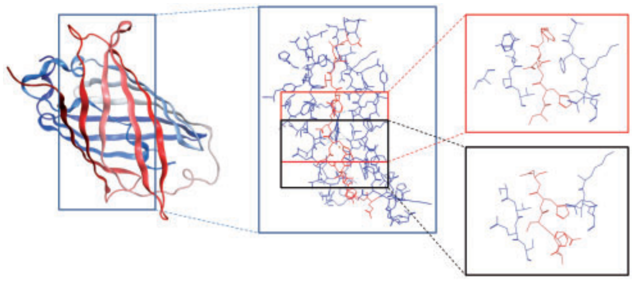
Fig. 1. Graphical representation of the piecemeal approach. Left: ribbon diagram of GFP (PDB code 2B3P), colored blue-to-red from N- to C-terminus. Center: wireframe representation of the 7.0 Å region (blue) surrounding beta Strand 7 (red). Right: 7.0 Å overlapping neighborhoods surrounding central residues 147 and 149 on beta Strand 7. The two neighborhoods share nine positions

In the one-pass piecemeal design, the MECs were expanded to profiles using simulated melting, and these profiles were merged and reconciled into one sequence profile (alternatively to a rotamer profile). This profile was the final result of the one-pass process.

In the two-pass piecemeal design, the merged profile from the one-pass design was converted to a new search space after applying a probability cutoff, $P > 0.1$. Design was carried out on this reduced search space using DEE or MC, and the final MEC was (optionally) expanded using simulated melting to produce the final result. Timings for the two-pass process include all of the design steps up to here; the merge and conversion operations have negligible runtime contribution. Runtime comparisons between methods were done using the two-pass process.

In the three-pass piecemeal design, the designed region was split both in real space and in rotamer space. After splitting into regions, multiple copies of each split search space were each reduced by random rotamer elimination to a specified target rotamer count. The number of copies depended on the target rotamer count and the desired redundancy. For example, if the rotamer sets were reduced by a factor of 10 and we wanted 5-fold redundancy, then a total of 50 randomly reduced copies would be generated. Redundancy is intended to ensure that the true rotamer is retained in at least one copy. Five-fold redundancy was found to be sufficient (n-fold = 5 in Fig. 2). After running DEE or MC design on each reduced copy, the resulting MECs were merged to form the rotamer set for a single piece, and a second round of DEE or MC design was carried out on that piece using the aforementioned rotamer set. The results of this second pass were merged as described above, using conservative or aggressive criteria, and this combined rotamer set was subject to a third round of DEE or MC.

2.6 Assessment

The quality of the protein design was assessed by comparing the all-atom energy of the final configuration and by comparing sequence profiles. Sequence profiles were visualized as Logos (Crooks *et al.*, 2004).

3 RESULTS

For the purposes of this study, we carried out designs using the leave-one-out (LOO) method, as described by Crone *et al.* (2012) and Huang *et al.* (2011). To create a LOO protein that binds a peptide sequence X at location Y in the circularly permuted and truncated template structure, the sequence segment Y is removed and is replaced with X, which must be of the same length. The sequence positions surrounding X are allowed to select any rotamer of any amino acid within a defined search space, called the master expression, while the positions within X are allowed to sample any rotamer of a single amino acid. Experimental confirmation of this method for generating biosensor proteins is ongoing and will be reported separately. The following are results for LOO designs using template proteins GFP (PDB ID: 2B3P) and streptavidin (PDB ID: 1STP). We first explored the runtime order as a function of the size of the search space, and then we explored the speedup and accuracy in design using various divide-and-conquer algorithms, collectively called 'piecemeal'. 'Full-length' refers to the algorithm in which the search space is not divided. One-pass, two-pass and three-pass piecemeal refer to algorithms in which the search space is split and

```
//*********************Variables*************************
// template = Atomic coordinates of backbone and side chains.
// residue = A template sequence position.
// rotamer_library = A list of discrete rotamers for each amino acid.
// space = A set of residue-specific rotamers, from a rotamer library.
// piece, subspace = A subset of a space.
// profile = A probability distribution of amino acids or rotamers.
// MEC = minimum energy configuration.
// nfold = redundancy value
// DEE = dead-end elimination algorithm
// MC = Monte Carlo rotamer search algorithm
// split_... = a subset of...
// central = a set of residues central to the design space.
// MSA = multiple sequence alignment
// BLOSUM = amino acid substitution matrix
//**********************Functions**************************
// Numsplit(space, rot_cutoff) = number of pieces of space after splitting
//   to size rot_cutoff; = 1 if space is already smaller than rot_cutoff.
// Split(space, [residue, range | rot_cutoff]) = a piece containing all
//   residues within range of residue, or rot_cutoff randomly
//   selected rotamers.
// Design(template, space) = The MEC of space, found by DEE or MC.
// Expand(MEC, template) = A profile expanded from MEC, generated
//   by "simulated melting".
// Union(profiles, cutoff) = A single profile from multiple profiles,
//   using the union operation, optionally applying a cutoff.
// Msa2space(template, central, range, MSA, BLOSUM,
//   rotamer_library) = a search space defined using a set of central
//   residues and a cutoff distance range, using all amino acids found
//   in MSA expanded by BLOSUM, all rotamers in rotamer_library

Piecemeal(template, central):
1. master_space = Msa2space(template, central, range, MSA,
   BLOSUM, rotamer_library)
2. select( Full-length, One-pass, Two-pass, Three-pass )

Full-length:
3.  MEC = Design(template, master_space)
4.  final_profile = Expand(MEC, template)

One-pass, Two-pass, Three-pass:
5. for i in central {
6.     piece[i] = Split(master_space, residue[i], range)
7.     if (Three-pass) {
8.        N = nfold * Numsplit(piece[i], rot_cutoff)
9.        for j in [1..N] {
10.          split_piece[j] = Split(piece[i], rot_cutoff)
11.          MEC = Design(template, split_piece[j])
12.          split_profile[j] = Expand(MEC, template)
13.       }
14.     subspace[i] = Union(split_profile[1..N], union_cutoff)
15.   } else {
16.       subspace[i] = piece[i]
17.   }
18.   MEC = Design(template, subspace[i])
19.   profile[i] = Expand(MEC, template)
20. }
21.  subspace = Union(profile[1..size(central)], union_cutoff)
22.  if (Two-pass or Three-pass) {
23.     MEC = Design(template, subspace)
24.     final_profile = Expand(MEC, template)
25.  } else {
26.     final_profile = subspace
27.  }
```

Fig. 2. Pseudocode description of the various design implementations used in this study

each sub-space is designed in one, two or three passes, with optional recombination and reconciliation of discrepancies.

3.1 Run-time order, DEE versus MC

Wall-clock runtimes were determined for design jobs of variable rotamer count using DEE or MC algorithms. The DEE runtime order was empirically estimated to be O($n^{3.5}$) both with respect to the rotamer count (Fig. 3a) and log of the combinatorial complexity of the search space (Fig. 3c). The MC runtime grew exponentially in both metrics (Fig. 3b and d). (Note that combinatorial complexity is not directly related to the rotamer count but also depends on how unevenly the rotamers are distributed in the sequence.) The choice of protein template, GFP or streptavidin, had no appreciable effect on the runtime order.

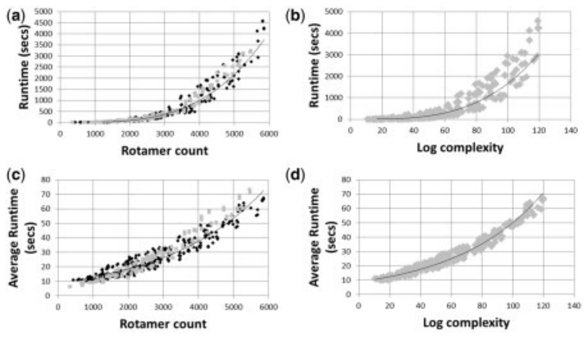

Fig. 3. Normalized timing data for MEC configuration searches as a function of algorithm used, DEE (**a** and **b**) or MC (**c** and **d**); rotamer count (a and c) and complexity of initial search spaces (b and d). Complexity, the product sum of rotamer counts at each position, is plotted as a logarithm. In figures a and c, GFP-based designs are in black, streptavidin designs are in gray; in figures b and d, only GFP-based designs are shown in gray. The lines on the graphs are regression fits to the GFP data only. For DEE, the fit is a power regression; for MC, it is exponential

Simple simulated annealing MC design was both faster than DEE design and scaled more slowly with design size. In fact, the average MC runtime for 6000 rotamers was only 2.5 times the average runtime for 3000 rotamers, a meager speedup. However, we found that the convergence criteria for the MC runtime (10 000 steps without an accepted move) was sufficient to give reproducible results for a 3000 rotamer set but not for a 6000 rotamer set. Results with 6000 rotamers were energetically poor and not reproducible. These observations reveal an energy landscape that is increasingly rugged with an increasing rotamer count. In a rugged landscape the starting configuration plays a strong role in determining the final result. A higher starting temperature and slower annealing schedule would have improved the reproducibility of the MC approach for larger rotamer sets but would do so at an unknown computational cost. Therefore, the true runtime dependency of MC design is underestimated in Figure 3. The task of optimizing the MC annealing rate as a function of the rotamer count was deemed impractical.

Given the need for reproducible and near optimal results, we selected the DEE algorithm for the purposes of assessing the piecemeal method. There is no reason to believe that the conclusions from DEE-based piecemeal design would not extend to MC-based piecemeal design, both in terms of speedup and accuracy. It should be mentioned that the optimality of the design algorithm (DEE is optimal, MC is heuristic) is not a reason for our selecting DEE. In fact, the protein design problem has many solutions, and finding the optimal one is generally not of critical importance. Because the energy landscape is rugged, it becomes prudent to use a deterministic approach such as DEE, which is insensitive to starting configuration.

3.2 Full-length design, without splitting

The sequence profiles and MEC energies produced by single design, in which the entire search space is explored at once, is the standard by which we measured the quality and efficiency of our proposed accelerated approaches, collectively called piecemeal design. The unsplit space of full-length design lacks the detrimental edge effects or any errors introduced by conflict resolution. To set this benchmark, we used the largest search space that would fit in memory, which is about 6000 rotamers. A coarse-grained rotamer library was used (Lovell *et al.*, 2000), allowing a relatively large region to fit in memory. The designed region included beta Strand 8 of GFP plus the 7 Å region surrounding it, consisting of 14 central positions surrounded by 54 positions—a total of 5800 rotamers. Full-length design (Fig. 2) was carried out using DEE. Results from piecemeal splitting of the same space were compared with this standard, both in terms of the runtime and optimality.

3.3 One-pass piecemeal design

The piecemeal approach attempts to recover the result of full-length design but faster and more efficiently. In 'one-pass' piecemeal, we divided the space surrounding a 14 residue beta strand into 14 overlapping regions, comprising a total of 54 designable positions. Of these positions, 37 were in overlap regions, potentially requiring conflict resolution. Each region was designed only once.

The amino acid selection at the 37 overlap positions differed between pieces in seven cases. To resolve conflicts, we simply selected the highest-frequency amino acid at each position. As a result, a total of six positions were mispredicted with respect to the full-length standard. Three of these six were in overlap positions, and three were in non-overlap positions. Note that the false-positive findings in the non-overlap regions could not have resulted from the naive approach to conflict resolution because there was no conflict to resolve. We conclude that errors in non-overlap regions were the result of edge effects that propagated inward, a domino effect that cannot be safely neglected.

3.3.1 One-pass piecemeal design with expansion In an attempt to recover the optimal amino acid selections, we applied the simulated melting algorithm, which produces an amino acid probability profile around the MEC. We set the ceiling value for sequence complexity to 1000 because this is the maximum number of sequences that could reasonably be screened experimentally. This translates to an average of 1.41 amino acids per position for 50 designed non-proline positions. One of the six false-positive findings, an overlap position, was eliminated by this simple procedure, but five were not and another two positions were incorrectly predicted, one in an overlap region and one in a non-overlapping region.

The surprising conclusion from this simple approach is that the majority of the designed positions, 48 out of 54, could be correctly predicted using only local information. We attributed the incorrect amino acid selections to edge effects, where the theoretical lowest energy rotamer requires moving a side-chain that is outside of the designed region and consequently is fixed to the wild-type rotamer.

3.4 Two-pass piecemeal design

Because the one-pass piecemeal design produced a significant number of errors, and even a single incorrect assignment can ruin protein stability, a second pass through DEE was required, starting with the results of the first pass. In the second pass, the search space was defined by merging the results of the first pass. The MEC from the first-pass design of each piece was expanded using the simulated melting algorithm to produce a profile. The profiles from all of the pieces were then merged into a single profile. The multiple profiles were merged one of two ways: (i) by keeping only the amino acids that occurred in all overlapping pieces or (ii) by keeping all amino acids of all overlapping pieces. We refer to these options as (i) intersection and (ii) union.

Intersection is a more aggressive approach, leading to faster runtimes. The full-length search space generated by intersecting all 14 pieces ran five times faster than the full-length design. However, the more aggressive approach has shortcomings. If an overlap position has low sequence diversity in the profile of one of the overlapping pieces, the optimal choice may be lost. A null intersection is also possible, although not observed in this case study. One of the designed positions in an overlap region was incorrectly resolved using this method.

3.4.1 Two-pass piecemeal design with union merging Using the less aggressive union approach when combining the piece profiles, we still observed a 4.8-fold speedup as compared with the full-length design with one incorrect assignment, a conservative and isosteric Q/E mutation. No errors were found in the non-overlap positions, showing that the errors in these positions had propagated from errors in the overlap regions. The union-merging approach was used for all further studies. The runtime for the second pass design was 12 times slower using union merging than the intersection approach because of the larger search space. But the time added in the second pass was insignificant when compared with the runtimes for the first pass.

3.4.2 Two-pass piecemeal design with rotamer profiles Using the two-pass method with expansion and union eliminated errors at the cost of runtime. We were able to improve efficiency without introducing errors by simply keeping the identities of the selected rotamers when expressing the output profile, instead of only keeping the identities of the amino acids. A selected amino acid may have any number of eliminated rotamers, as long as one of its rotamers was not eliminated. By keeping this information, we avoided the duplication of effort in the second pass. The simulated melting procedure was again used to augment the output rotamer profile, and these profiles were combined into a second-pass search space using the conservative union approach. The number of configurations to save during simulated melting was increased so that we retained a sufficient fraction of the search space.

The rotamer-level second-pass search space contains half as many rotamers as the sequence-level search space and finds the same MEC as the full-length search space (no errors) in one-twelfth of the time as compared with the sequence-level second pass, giving a total speedup of about five-fold for the rotamer-level two-pass process over the starting full-length search space. Furthermore, when the final MEC of each design process was diversified using simulated melting, the output profiles showed good similarity between the full-length search space and the two-pass piecemeal spaces (Fig. 4), demonstrating that the general energy landscape is not significantly perturbed by the initial division and reassembly of the search space.

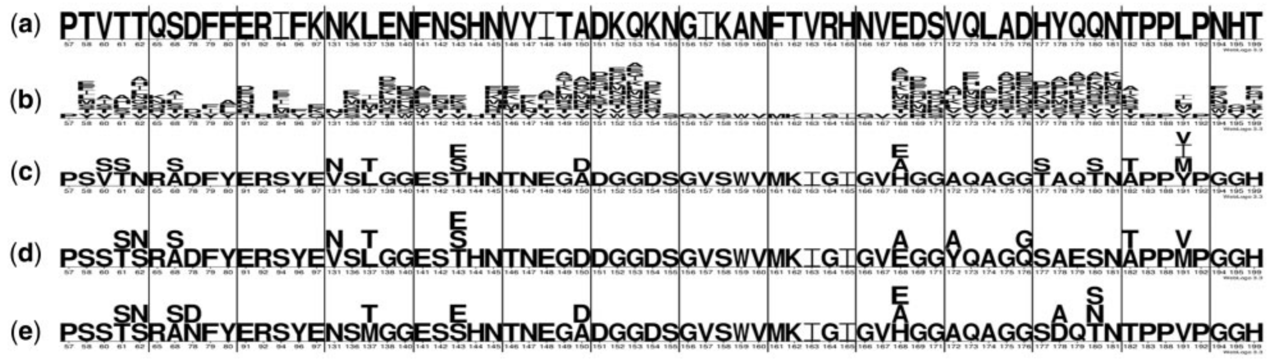

Fig. 4. Sequence logo comparison of the starting and ending design libraries for a GFP-based design. (**a**) Starting amino acid sequence at designable positions. (**b**) Master search space amino acids allowed at each designable position. (**c**) Final library after full-length design. (**d**) Final library after two-pass piecemeal using sequence-level union operation. (**e**) Final library after two-pass piecemeal using rotamer-level union operation

3.5 Three-pass piecemeal with fine-grained rotamers

Splitting the search space by regions greatly reduces the amount of memory required to run the DEE algorithm, which requires storage for pre-calculated rotamer pair interactions, and therefore scales quadratically with the number of rotamers. But this reduction may not be enough if we wish to use a fine-grained rotamer library. The fine-grained sampling of side-chain conformations improves the chances of finding the correct lowest energy design because packing interactions in the core of a protein are very sensitive to small adjustments in the side-chain dihedral angles. We considered two levels of rotamer expansion. Rotamers were expanded at the χ_1 angle or at both χ_1 and χ_2 by rotating the dihedral angle 10° in either direction. Expanding χ_3 and χ_4 angles affects only the longest amino acids and this was not done.

When the rotamer library is expanded at χ_1, effectively tripling the rotamer count, the full-length search space will no longer fit in memory, but the 14 pieces do. However, after running design on the 14 pieces and merging the resulting amino acid profiles, the search space is again too big to fit in memory. Using rotamer profiles instead of amino acid profiles (as described above) allowed the second-pass search space to fit in memory.

However, when the rotamer library was expanded out to χ_2, the first-pass search spaces for the 14 pieces no longer fit in memory. Therefore, a three-pass hierarchical design pipeline was developed, as shown in Figure 2. The additional step is added at the beginning of the pipeline, in which the search space for the pieces is downsampled until it fits in memory. To ensure full coverage, redundant copies were designed. The resulting profiles were then merged to create a new search space for the piece, smaller than the original piece. The pipeline then proceeds as for two-pass piecemeal design. The MEC from the three-pass design with rotamers expanded out to χ_2 had a slightly lower energy than the MEC from the χ_1 expanded rotamers, as one would expect for a more fine-grained rotamer search. The overall runtime for three-pass design with χ_2 expansion was 480-fold faster than the theoretical runtime for full-length design using the same search space. The timing for the full-length design job must remain theoretical, as it would require more memory than is available (over 9 Gb).

3.6 Validation studies

To validate the results described in the case study, three additional case studies were done, including two other LOO designs on GFP (β Strands 7 and 11) and a LOO design on β Strand 2 of streptavidin. In each case, the MEC from the full-length design is reproduced, much faster, using the two-pass piecemeal design with union merging and rotamer profiles. Additionally, a full redesign was carried out on nine other proteins using full-length and piecemeal design with neighborhood radii 7, 9 and 12 Å. The optimal design was reproduced with high accuracy using the two-pass piecemeal design in all but one case. The larger neighborhood radii did not have a significant effect on the design results. See Supplementary Data for more information.

4 DISCUSSION

We may conclude that protein design tasks may be effectively split into smaller pieces without loss of accuracy, provided that certain precautions are taken. Success was expected because the pairwise energetic interactions that govern the search are short ranged. However, when the search space was split into pieces, we observed edge effects where positions near the boundaries of pieces were incorrectly assigned, and we observed a domino effect where erroneous assignments near the boundaries propagated inward.

Both types of errors were ameliorated by two approaches to less aggressive rotamer elimination. First, the MEC produced by DEE or MC-based design was expanded to include the 1000 nearest and lowest-energy sequences, using an algorithm we called 'simulated melting'. Second, the sequences assigned to positions found in overlap regions between pieces were reconciled conservatively by keeping all observed amino acids at all positions, the union approach. Following this reconciliation, ambiguities were resolved with a second pass through the design algorithm.

Improved efficiency of the design process is gained using piecemeal with respect to memory usage as well. The LOO–GFP test case described above requires 548 MB of memory in the full-length design, but the largest single amino acid region only requires 49 MB. The memory savings are more pronounced in the case of the full redesigns: full-length redesign of dihydrofolate reductase requires 3 GB of memory, but 7.0 Å regions surrounding each position of the protein range from only 1.3 MB to 90 MB.

Computational efficiency was improved by expressing the MEC as a rotamer profile instead of a sequence profile, avoiding the duplication of effort in two-pass design. Fine-grained rotamer libraries were made accessible to the memory requirements of the DEE algorithm by splitting the search space at the configuration level and carrying out an additional design step prior to the two-pass process. This extra level of splitting did not introduce errors if we used at least five-fold redundancy.

Piecemeal design has an experimental parallel in directed evolution via DNA shuffling, where beneficial mutations from sub-optimal DNA sequences are combined and are subjected to screening to produce an improved functionality of interest. The piecemeal approach 'shuffles' sub-optimal MECs from pieces to achieve the global minimal energy conformation.

4.1 Distributed computing

The rationale for subdividing a design problem originally arose from the prospect of expanding computational design projects into a volunteer computing environment (Anderson, 2004). In such a computing paradigm, the time and resources available to a given job are not necessarily known or reliable; therefore, it is preferable to produce jobs with a minimal computational footprint both in runtime and in memory required. Robustness to failure is a must for volunteer computing, and this is easily handled in our approach by adding redundant jobs.

4.2 Applications

Because this approach to design is implementation-independent, it provides the most promise to design programs that rely on computationally expensive energy functions (Liang and Grishin, 2002; Peterson *et al*., 2004), complex configurational search algorithms (Georgiev *et al*., 2008; Kolodny *et al*., 2005; Mandell *et al*., 2009; Noonan *et al*., 2005; Smith *et al*., 2008) and fine-grained rotamer libraries (Dunbrack, 2002; Shapovalov and Dunbrack, 2011). The process of adapting piecemeal to other kinds of design problems depends only on how we define the central residues and the desired radius of interaction to consider. The application of this methodology to algorithmic extensions involving backbone movement or continuous rotamers on a single template (Hallen *et al*., 2013; Mandell and Kortemme, 2009) is unclear; however, a multi-template approach based on MC simulations has been developed, which demonstrates some compatibility with the piecemeal approach. This new approach, termed 'plastic design', enforces the constraint that each template must use the same amino acid and approximately the same side-chain dihedral angles for each move of the configurational search.

In conclusion, we present a simple and generally applicable approach to parallel processing of protein design that runs fast, introduces no errors and enables larger and finer-grained design tasks.

ACKNOWLEDGEMENTS

The authors are indebted to the Center for Computational Nanotechnology Innovations for use of computing resources. The authors thank Carlos Varela, Chris Carothers and Alisa Neeman for helpful discussions.

Funding: Research reported in this publication was supported by the National Institute of General Medical Sciences of the National Institutes of Health under Award Number (R01GM099827). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al*. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*., **25**, 3389–3402.

Ambroggio,X.I. and Kuhlman,B. (2006) Computational design of a single amino acid sequence that can switch between two distinct protein folds. *J. Am. Chem. Soc*., **128**, 1154–1161.

Anderson,D. (2004) BOINC: a system for public-resource computing and storage. In: *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID '04)*. IEEE Computer Society, Washington, DC, pp. 4–10.

Andronescu,M. *et al*. (2003) RNAsoft: a suite of RNA secondary structure prediction and design software tools. *Nucleic Acids Res*., **31**, 3416–3422.

Canutescu,A.A. *et al*. (2003) A graph-theory algorithm for rapid protein side-chain prediction. *Protein Sci*., **12**, 2001–2014.

Cochran,F.V. *et al*. (2005) Computational *de novo* design and characterization of a four-helix bundle protein that selectively binds a nonbiological cofactor. *J. Am. Chem. Soc*., **127**, 1346–1347.

Crone,D.E. *et al*. (2012) GFP-based biosensors. In: Rinken,T. (ed.) *State of the Art in Biosensors - General Aspects*. InTech, Rijeka, Croatia.

Crooks,G.E. *et al*. (2004) WebLogo: a sequence logo generator. *Genome Res*., **14**, 1188–1190.

Dallüge,R. *et al*. (2007) A tetrapeptide fragment-based design method results in highly stable artificial proteins. *Proteins*, **68**, 839–849.

Dantas,G. *et al*. (2003) A large scale test of computational protein design: folding and stability of nine completely redesigned globular proteins. *J. Mol. Biol*., **332**, 449–460.

Desmet,J. *et al*. (1992) The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, **356**, 539–542.

Dunbrack,R.L. (2002) Rotamer libraries in the 21st century. *Curr. Opin. Struc. Biol*., **12**, 431–440.

Gabriel,E. *et al*. (2004) Open MPI: goals, concept, and design of a next generation MPI implementation. In: *Proceedings, 11th European PVM/MPI Users' Group Meeting*. Budapest, Hungary.

Gao,J.Z.M. *et al*. (2010) Inverse folding of RNA pseudoknot structures. *Algorithms Mol. Biol*., **5**, 27.

Georgiev,I. *et al*. (2006) Improved pruning algorithms and divide-and-conquer strategies for dead-end elimination, with application to protein design. *Bioinformatics*, **22**, e174–e183.

Georgiev,I. *et al*. (2008) Algorithm for backrub motions in protein design. *Bioinformatics*, **24**, i196–i204.

Goldstein,R.F. (1994) Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophys. J*., **66**, 1335–1340.

Gordon,D.B. and Mayo,S.L. (1998) Radical performance enhancements for combinatorial optimization algorithms based on the dead-end elimination theorem. *J. Comput. Chem*., **19**, 1505–1514.

Guntas,G. *et al*. (2010) Engineering a protein–protein interface using a computationally designed library. *Proc. Natl Acad. Sci. USA*, **107**, 19296–19301.

Hallen,M.A. *et al*. (2013) Dead-end elimination with perturbations (DEEPer): a provable protein design algorithm with continuous sidechain and backbone flexibility. *Proteins*, **81**, 18–39.

Henikoff,J.G. and Henikoff,S. (1996) Using substitution probabilities to improve position-specific scoring matrices. *Comput. Appl. Biosci*., **12**, 135–143.

Huang,Y.M. *et al.* (2011) Quantitative *in vivo* solubility and reconstitution of truncated circular permutants of green fluorescent protein. *Protein Sci.*, **20**, 1775–1780.

Huang,Y.M. and Bystroff,C. (2013) Expanded explorations into the optimization of an energy function for protein design. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **PP**, 1.

Jiang,L. *et al.* (2008) *De novo* computational design of retro-aldol enzymes. *Science*, **319**, 1387–1391.

Joachimiak,L.A. *et al.* (2006) Computational design of a new hydrogen bond network and at least a 300-fold specificity switch at a protein−protein interface. *J. Mol. Biol.*, **361**, 195–208.

Karanicolas,J. *et al.* (2011) A *de novo* protein binding pair by computational design and directed evolution. *Mol. Cell.*, **42**, 250–260.

Kolodny,R. *et al.* (2005) Inverse kinematics in biology: the protein loop closure problem. *Int. J. Robot. Res.*, **24**, 151–163.

Kortemme,T. *et al.* (2004) Computational redesign of protein-protein interaction specificity. *Nat. Struct. Mol. Biol.*, **11**, 371–379.

Kuhlman,B. *et al.* (2003) Design of a novel globular protein fold with atomic-level accuracy. *Science*, **302**, 1364–1368.

Krivov,G.G. *et al.* (2009) Improved prediction of protein side-chain conformations with SCWRL4. *Proteins*, **77**, 778–795.

Liang,S. and Grishin,N.V. (2002) Side-chain modeling with an optimized scoring function. *Protein Sci.*, **11**, 322–331.

Lovell,S.C. *et al.* (2000) The penultimate rotamer library. *Proteins*, **40**, 389–408.

Malakauskas,S.M. and Mayo,S.L. (1998) Design, structure and stability of a hyperthermophilic protein variant. *Nat. Struct. Biol.*, **5**, 470–475.

Mandell,D.J. and Kortemme,T. (2009) Backbone flexibility in computational protein design. *Curr. Opin. Biotech.*, **20**, 420–428.

Mandell,D.J. *et al.* (2009) Sub-angstrom accuracy in protein loop reconstruction by robotics-inspired conformational sampling. *Nat. Methods*, **6**, 551–552.

Moltó,G. *et al.* (2009) Protein design based on parallel dimensional reduction. *J. Chem. Inf. Model.*, **49**, 1261–1271.

Noonan,K. *et al.* (2005) Probik: protein backbone motion by inverse kinematics. *Int. J. Robot. Res.*, **24**, 971–982.

Peterson,R.W. *et al.* (2004) Improved side-chain prediction accuracy using an *ab initio* potential energy function and a very large rotamer library. *Protein Sci.*, **13**, 735-–751.

Pierce,N.A. *et al.* (2000) Conformational splitting: a more powerful criterion for dead-end elimination. *J. Comput. Chem.*, **21**, 999–1009.

Röthlisberger,D. *et al.* (2008) Kemp elimination catalysts by computational enzyme design. *Nature*, **453**, 190–195.

Samish,I. *et al.* (2011) Theoretical and computational protein design. *Annu. Rev. Phys. Chem.*, **62**, 129–149.

Shapovalov,M. and Dunbrack,R.L. (2011) A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*, **19**, 844–858.

Smith,C.A. and Kortemme,T. (2008) Backrub-like backbone simulation recapitulates natural protein conformational variability and improves mutant side-chain prediction. *J. Mol. Biol.*, **380**, 742–756.

Street,A.G. and Mayo,S.L. (1999) Computational protein design. *Structure*, **7**, R105–R109.

Vanderbilt,D. and Louie,S.G. (1984) A Monte carlo simulated annealing approach to optimization over continuous variables. *J. Comput. Phys.*, **56**, 259–271.