OSPREY 3: Open-Source Protein Redesign for You, Refactored, with Powerful New Features

Donald Lab

October 28, 2017

1 Abstract

We present OSPREY 3, a new and greatly improved release of the OSPREY protein design software. OSPREY 3 features a convenient Python interface, greatly improving OSPREY's ease of use. OSPREY 3 also includes several new algorithms, which introduce substantial speedups as well as improved biophysical modeling. *BBK** is a much more efficient version of OSPREY's K^* free energy-based protein design algorithm. LUTE uses machine learning to frame designs with advanced modeling assumptions and continuous flexibility as very efficiently solvable discrete optimization problems without significant loss of accuracy. BWM*exploits sparsity in residue interaction graphs to perform protein design in provably subexponential time, and CATS improves OSPREY's modeling of backbone flexibility. Even without these improved algorithms, OSPREY 3 is over two orders of magnitude faster than previous versions of OSPREY when run on the same hardware. It also introduces GPU support, which provides an additional speedup of over an order of magnitude when run on desktop-class GPUs. OSPREY has always offered a unique package of advantages over other design software: provable design algorithms, awareness of continuous flexibility during design, and modeling of conformational entropy. With this new version, we believe it empowers significantly improved computational protein design compared to the previous state of the art.

2 Introduction

For over a decade, the OSPREY software package [7, 9, 10] has offered the protein design community a unique combination of continuous flexibility modeling, ensemble modeling, and algorithms with provable guarantees. Having begun as a software release for the K^* algorithm, which approximates binding constants using ensemble modeling, it now boasts a wide array of algorithms found in no other software. OSPREY has been used in many designs that were empirically successful—*in vitro* [2, 5, 8, 12, 21, 23, 26] and *in vivo* [5, 12, 21, 23] as

well as in non-human primates [23]. OSPREY's predictions have been validated by a wide range of experimental methods, including binding assays, enzyme kinetics and activity assays, in cell assays (MICs, fitness) and viral neutralization, *in vivo* studies, and crystal structures [20, 23].

However, as OSPREY grew to include more algorithms and features, the code became increasingly complicated and difficult to maintain. The growing complexity of the software also hindered its ease-of-use. OSPERY 3 represents a complete refactoring of the code, and presents a simpler and more intuitive interface that makes protein redesign much easier than before. The new code organization also facilitates adding new features to the open-source project, both by ourselves and by other contributors. We have introduced a convenient Python scripting interface and added support for GPU acceleration of the bulk of the computation, allowing designs to be completed much more quickly and easily than in previous versions of OSPREY. We believe OSPREY 3 will be a very useful tool for both developers and users of provably accurate protein design algorithms.

2.1 Past successes of OSPREY

OSPREY has been used for an impressive number of empirically successful designs, ranging from enzyme design to antibody design to prediction of antibiotic resistance mutations. Notably, OSPREY has been successful in many *prospective* experimental studies, i.e., studies in which our designed sequences are tested experimentally, thus providing more realistic validation than a retrospective comparison of OSPREY calculations to previous experimental results. OSPREY is most applicable to problems that can be posed in terms of binding, allowing the K^* algorithm and its variants to select the optimal sequence based on an estimate of binding free energy. But most protein design problems can be posed in this way, sometimes in terms of binding to more than one ligand.

For example, we have successfully predicted novel resistance mutations to new inhibitors in MRSA (methicillin-resistant *Staphylococcus aureus*), by searching for sequences that have impaired drug binding compared to wild-type DHFR, but still form the enzyme-substrate complex as usual, allowing catalysis [5, 20]. Our predictions were validated not only biochemically and structurally, but also at an organismal level [18, 20]. Similarly, we have successfully changed the preferred substrate of an enzyme—the phenylalanine adenylation domain of gramicidin S synthetase—from phenylalanine to leucine by modeling of the two enzyme-substrate complexes, searching for sequences with improved binding to leucine and less to phenylalanine.

Still other successes of OSPREY have involved improving a single binding interaction, like the interaction of the antibody VRC07 with its antigen, the gp120 surface protein of HIV. Using this approach, we designed a broadly neutralizing antibody VRC07-523LS against HIV with unprecedented breadth and potency that is now in clinical trials [1, 23]. Likewise, we have used OSPREY to develop peptide inhibitors of CAL, a protein involved in cystic fibrosis [21]. This is a protein design problem of direct therapeutic significance

that consists of optimizing a protein-protein binding interaction.

In addition, a number of other research groups have successfully used the OSPREY algorithms and software (by themselves) to perform biomedically important protein designs, *e.g.,* to design anti-HIV antibodies that are easier to induce [11]; to design a soluble prefusion closed HIV-1-Env trimer with reduced CD4 affinity and improved immunogenicity [3]; to design a transmembrane Zn^{2+}-transporting four-helix bundle [16]; to optimize stability and immunogenicity of therapeutic proteins [19, 24, 28]; and to design sequence diversity in a virus panel and predict the epitope specificities of antibody responses to HIV-1 infection [4].

We believe OSPREY 3 will enable an even greater range of successful designs.

3 New protein design algorithms in version 3

3.1 LUTE: Putting advanced modeling into a form suitable for efficient, discrete design calculations

OSPREY 3 comes with LUTE [15], a new algorithm that addresses two issues with previous versions of OSPREY.

First, previous versions modeled continuous flexibility by enumerating conformations in order of a *lower bound* on minimized conformational energy [6,9]. This approach is often inefficient in that many conformations—possibly even a number exponential in the number of mutable residues—can have lower bounds below the GMEC energy, and thus will all have to be enumerated. Only a small gain in efficiency is obtained by minimizing the energies of the partial conformations corresponding to nodes of the A* tree [14], again because of the gap between lower bounds and actual minimized energies. LUTE addresses this problem by directly optimizing the minimized energies of full conformations, which are estimated using an expansion in low-order tuples of residue conformations. Thus, the burden of modeling continuous flexibility is shifted from the combinatorial optimization (A*) step, which has unfavorable asymptotic scaling, to a precomputation step that only scales quadratically with the number of residues. This precomputation step consists of sampling a "training set" of conformations, computing their minimized energies, and then inferring the coefficients of the expansion. These coefficients can then be used as residue interaction energies in combinatorial search, whether single- or multistate. The combinatorial search will have the form of a discrete search and thus achieve high efficiency, but will accurately match the results of a continuously flexible search.

Second, all previous combinatorial protein design algorithms have relied on an explicit decomposition of the energy as a sum of local (e.g., pairwise) terms. This made design with energy functions that do not have this form difficult. For example, previous use of the Poisson-Boltzmann [25] energy function, the gold standard of implicit solvent modeling, in design has relied either on *post-hoc* reranking of a limited number of favorable designs from a calculation based on pairwise energies, which would cause all other designs favored by

the Poisson-Boltzmann energetics to be missed, or on a decomposition that is incompatible with continuous flexibility [27]. However, LUTE need only calculate the energies of entire conformations in order to infer its coefficients—explicit pairwise energies are not part of this calculation. Thus LUTE can straightforwardly support general energy functions, and as shown in [15] it can obtain good fits at least in the case of Poisson-Boltzmann energies.

OSPREY users can now turn on LUTE for continuously flexible calculations simply by setting the configuration "useTupExp" to true. OSPREY 3 also supports design with Poisson-Boltzmann solvation energy calculations, which use the DelPhi [17,22] software for the single-point Poisson-Boltzmann calculations (we ask the user to download DelPhi separately for licensing reasons). But as an algorithm, LUTE's abilities go well beyond these features—it is a general tool for taking advanced modeling of a single voxel in a system's conformation space and putting into a suitable form for efficient, discrete combinatorial optimization calculations yielding the best design sequence. As mentioned in [15], we are currently working on adding other capabilities like continuous entropy modeling this way. Moreover, any other researchers who would like to model some phenomenon in protein design, but find it difficult to fit into the usual discrete pairwise framework used in design calculations, are encouraged to try LUTE and OSPREY 3 as a framework for their modeling. Such improved modeling is essential to increasing the reliability of and range of feasible uses for computational protein design.

3.2 CATS: Local backbone flexibility in all biophysically feasible dimensions

OSPREY pioneered protein design calculations that model local continuous flexibility of sidechains in the vicinity of rotamers in all biophysically feasible dimensions (i.e., the sidechain dihedrals). This continuous flexibility was often critical in finding optimal sequences [6], and especially in eliminating artificial steric problems for ideal rotameric conformations that are chosen without consideration of protein context. In OSPREY 3, we now extend this ability to the backbone: allowing local continuous backbone flexibility in the vicinity of the native backbone in all biophysically feasible dimensions.

This flexibility is enabled by the CATS algorithm [13]. CATS uses a new parameterization of backbone conformational space, along with the voxel framework that OSPREY has always included. CATS is equivalent to searching over all changes in backbone dihedrals (ϕ and ψ) subject to keeping the protein conformation constant outside of a specified flexible region. This constraint is necessary to keep larger backbone dihedral changes from propagating down the backbone and unfolding the protein. CATS includes an efficient Taylor series-based algorithm for computing atomic coordinates from its new degrees of freedom, enabling efficient energy minimization. CATS is intended to be run along with OSPREY's other algorithms, yielding efficient calculations with continuous flexibility in both the sidechains and the backbone. In Ref 13, we have shown that backbone flexibility as modeled by CATS is sometimes critical for resolving artificial steric problems and often

affects energetics significantly, just as has previously been shown for continuous sidechain flexibility [6].

3.3 *BBK**: Efficiently computing the tightest binding sequences from a combinatorially large number of binding partners

(JJ: This was an attempt to describe K^*. It's more of a placeholder to show the tone and points I think we should get across.) Although many GMEC-based designs predict sequences which fold and even bind the desired target, proteins do not exist in nature as a single static structure, but instead as thermodynamic ensemble of structures. Protein design algorithms that optimize binding affinity search for sequences whose thermodynamic ensemble energetically favor the desired bound or unbound states over other undesirable states. In doing so, these algorithms search for sequences whose conformational ensemble may contain multiple low-energy conformations in the desired state. Algorithms whose input model accounts for the ensemble-nature of proteins more accurately represent protein flexibility, and can identify sequences with multiple low-energy conformations which GMEC-based algorithms would overlook. The K^* algorithm [] models an ensemble of Boltzmann-weighted conformations to approximate the Boltzmann-weighted partition function Z. It combines dead-end elimination pruning [] with A^* [] gap-free conformation enumeration to compute provable ε-approximations Z for the protein states of interest. K^* combines these Z scores to approximate the association constant, K_a, as the ratio of ε-approximate partition functions between the bound and unbound states of a protein-ligand complex. Notably, each partition function ratio, called a K^* *score*, is provably accurate with respect to the the biophysical *input model* []. *(JJ: I suspect an explanation of the input model belongs elsewhere in this paper. It wouldn't make sense to introduce it after describing LUTE, etc. I'm keeping it in the source file so we can move it where it needs to go.)* K^* efficiently approximates K_a by provably enumerating a gap-free list of low-energy conformations without exploring infrequently observed high-energy conformations in the protein or ligand. By doing so K^* is considerably more efficient than exhaustive conformation enumeration, enumerating as little as 3% of the space of all possible conformations for a given sequence. Notably, K^* is able to efficiently search over conformations for a *single sequence*. A long-standing area for potential improvement has therefore been to develop algorithms that efficiently search over *multiple sequences*. All provable ensemble-based algorithms, as well as many heuristic algorithms which optimize binding affinity, are *single-sequence* algorithms, which must compute or bound the binding affinity for each possible sequence. The asymptotic runtime complexity of single-sequence algorithms is therefore linear in the number of possible sequences, and exponential in the number of mutable residues. Therefore, designs with many mutable residues rapidly become intractable when using single-sequence algorithms. To manage the combinatorial explosion of the sequence space, K^* uses its inter-mutation pruning filter to prune sequences whose K^* scores provably cannot be within a user-specified factor of the best sequence encountered thus

far. Nevertheless, inter-mutation pruning is applied only *after* K^* initiates binding affinity computation, meaning that K^* must compute the K^* score for each possible sequence.

OSPREY 3 provides a new algorithm, *BBK**, which overcomes this challenge. *BBK** [**?**] builds on K^*, and is the first provable, ensemble-based protein design algorithm to run in time sublinear in the number of sequences. The key innovation in *BBK** that enables this improvement is the *multi-sequence bound* (MS). Rather than compute binding affinity separately for each possible sequence, as single-sequence methods do, *BBK** efficiently computes a single provable K^* score upper bound for a combinatorial number of sequences. *BBK** uses MS bounds to prune a combinatorial number of sequences during the search, entirely avoiding single-sequence computation for all pruned sequences. Indeed, this combinatorial pruning produces a significant empirical runtime speedup: *BBK** runs in time sub-linear in the number of sequences. In our experiments, *BBK** provably computed the tightest binding sequences while computing K^* scores for up to 10^5 fewer sequences than any single-sequence algorithm.

Importantly, *BBK** also contains many other powerful algorithmic improvements and implementation optimizations: the parallel architecture of *BBK**, which enables concurrent energy minimization, and a novel two-pass partition function bound, which minimizes far fewer conformations while still computing a provable ε-approximation to the partition function. Combined with the combinatorial pruning power of the MS bound, *BBK** is able to search over sequence spaces [X] orders of magnitude larger than previously possible with single-sequence K^*. Not only is *BBK** able to provably bound and prune a combinatorial number of suboptimal sequences, *BBK** also provably approximates K^* scores for individual sequences [Y] times faster than single-sequence K^*.

(Jeff: Do we want to mention MPLP here?)

4 Performance enhancements in version 3

4.1 Further optimizing code yields large single-threaded speedups

OSPREY 3's code has been heavily optimized to improve single-threaded performance. Two main areas have received the most attention, and the most improvement in performance, so far: A* search speed, and conformation minimization speed.

The performance of A* search in OSPREY depends mostly on the size of the conformation space of the design. More mutable and flexibile residues create a larger conformation space which must be searched systematically to find the lowest-energy conformations while perserving guarantees on solution quality, and hence the search requires more time. Search time is also dependent on the speed at which we can execute the scoring functions on A* nodes. Optimizations in OSPREY 3 have dramatically increased the A* node scoring speed mainly through the re-use of computed values between different nodes. Many intermediate values used by the A* scoring functions need only be computed once per design and can be cached throughout the rest of the search. This reduces the cost of node scoring by

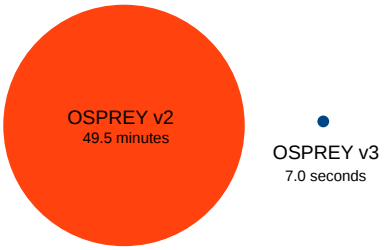

Figure 1: On a small benchmark sidechain packing problem involving a 114-residue fragment of PDZ3 domain of PSD-95 protein complexed with a 6-residue peptide ligand (PDB ID: 1TP5) and consisting of six continuously flexible residues, OSPREY 3 (blue) finished the design 425-fold faster than OSPREY 2 (red). Runtimes for the benchmark design are represented as the areas of the discs. A smaller disc represents a faster runtime. This benchmark was performed in a single thread running on an Intel Xeon E5-2640 v4 CPU.

roughly an order of magnitude. We can also score child nodes differentially against their parent nodes to speed up node scoring. Caching intermediate values during the parent node scoring and using them to simplify child node scoring yields roughly another order of magnitude speedup in A* node scoring. *(Jeff: trying to be concise here, hopefully we don't need the mathematical details of these A* scoring functions?)*

OSPREY 3 also includes optimizations to improve the performance of forcefield evaluation and conformation minimization. The code in OSPREY 3 which evaluates forcefield energies for a protein conformation has been heavily optimized, although speed gains here over OSPREY 2 are modest (roughly two-fold), since the original code was already well-optimized in this area. Much larger performance increases were gained by caching forcefield parameters and lists of atom pairs between different conformations to be minimized yielded roughly a 10-fold increase in speed. *(Jeff: I'm going off of memory for the impact of these optimizations. Hopefully rough estimates are good enough here? Or maybe this is too much detail altogether?)*

Combined together, these optimizations to single-threaded performance made OSPREY 3 about 425-fold faster than OSPREY 2 on a small benchmark sidechain packing problem (See Figure 4.1).

4.2 GPU acceleration reduces design runtimes

A major performance bottleneck of OSPREY designs with continuous flexibility is minimizing protein conformations and conformation fragments. Since the space of possible conformations in a design grows exponentially with the number of flexible and mutable residues, the number of conformations that must (in the worst-case) be enumerated to find the GMEC or to approximate the Boltzmann-weighted partition function grows exponentially as well. Since all enumerated conformations must be subsequently minimized to

compute their energies, and minimization is a relatively expensive operation, the bulk of a design's runtime can be spent on energy minimization of conformations. Therefore, improvements to the speed of energy minimization (an operation that is called an exponential number of times) can have a dramatic impact on OSPREY runtimes.

Much work has been done in the past to optimize OSPREY for execution on CPUs, particularly highly multi-core CPUs and even networked clusters of CPU-powered servers *(JJ: cite DACS (Georgiev bioinformatics 2006), cOsprey (Pan JCB 2016))*. However, modern GPU hardware enables high-performance computation at a fraction of the cost of large CPU clusters, mainly due to the huge video game industry that propels innovation in hardware design and drives down costs. OSPREY 3 includes GPU programs (called *kernels*) built using the CUDA framework that implements the forcefield calculations and local minimization algorithms used in protein redesign. We present performance results of these GPU kernels on various hardware platforms in Figure 4.2. Overall, on desktop-class machines, GPU hardware improves minimizations speeds over multi-core CPUs by roughly 12-fold for sufficiently large molecules. On server-class machines, a single GPU is merely about 2.6-fold faster than dual CPUs. These large machines can house more GPUs than CPUs though, so a fully-provisioned server with 4 or even 8 GPUs would likely greatly outperform 2 or even 4 CPUs. Laptop-class GPUs, however, do not appear to be powerful enough to offer any advantages over traditional CPU computing in OSPREY 3.

Modern GPU architectures offer thousands of parallel hardware units for calculations, compared to the tens of parallel hardware units in modern CPU architectures. The performance results of the current incarnation of OSPREY's GPU kernels indicate that current minimization speeds on GPUs have only begun to scratch the surface of what is possible, particularly for molecules with a small number of atom pairs. It is incredibly likely that future versions of these GPU kernels will offer significantly higher performance on the same hardware – perhaps allowing minimization speeds many times faster than today's GPU kernels.

5 Python scripting improves ease-of-use

One of the most visible additions to OSPREY 3 is the Python application programming interface (API), which allows fine-grained control over design parameters in a streamlined and easy-to-use experience. OSPREY 3 still supports a command-line interface with configuration files for backwards compatibility, but new development will be focused mostly on the new Python interface. *(Jeff: Is this even true? Not entirely sure...) (JJ: I'm not sure myself, since for day-to-day uses I wound up writing python scripts to produce a modified command line interface. It's worth discussing among the OSPREY developers. I'd say that a Python interface is friendlier than Java, and doesn't require compilation, which does save protein designers time.)*

The OSPREY 3 distribution contains a Python module which can be installed using the

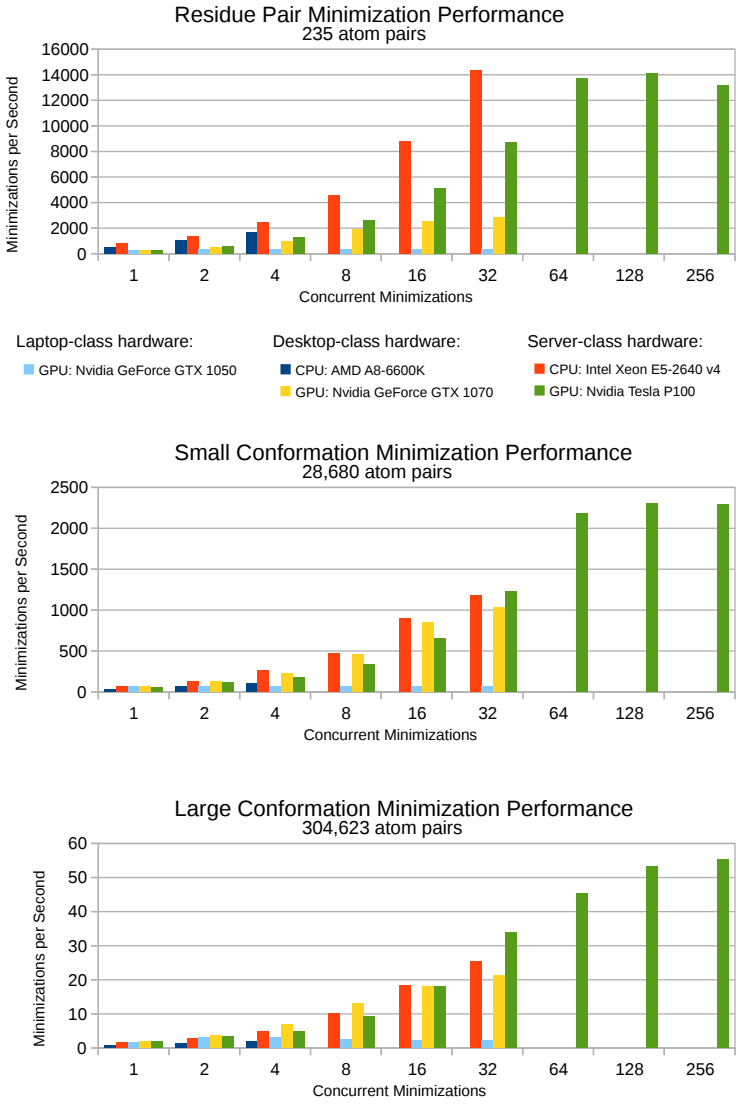

Figure 2: Benchmarks for protein conformation minimization in OSPREY 3 for various hardware platforms and for molecules of varying size. From smallest to largest: **(top)** a single residue pair used in energy matrix computations, **(middle)** a full protein conformation with a single flexible residue, **(bottom)** a full protein conformation with 20 flexible residues. For CPU hardware, concurrent minimizations correspond to CPU threads. For GPU hardware, concurrent minimizations correspond to *streams* defined by the CUDA framework. Faster minimization speeds correspond with faster OSPREY runtimes.

```
import osprey

osprey.start()

# define a strand
strand = osprey.Strand('1CC8.ss.pdb')
strand.flexibility[2].setLibraryRotamers('ALA', 'GLY')
strand.flexibility[3].setLibraryRotamers(osprey.WILD_TYPE, 'VAL')
strand.flexibility[4].setLibraryRotamers(osprey.WILD_TYPE)

# make the conf space
confSpace = osprey.ConfSpace(strand)

# choose a forcefield
ffparams = osprey.ForcefieldParams()

# how should we compute energies of molecules?
ecalc = osprey.EnergyCalculator(confSpace, ffparams)

# how should we define energies of conformations?
confEcalc = osprey.ConfEnergyCalculator(confSpace, ecalc)

# how should confs be ordered and searched?
emat = osprey.EnergyMatrix(confEcalc)
astar = osprey.AStarMPLP(emat, confSpace)

# find the best sequence and rotamers
gmec = osprey.GMECFinder(astar, confEcalc).find()
```

Figure 3: A Python script that performs a very simple design in OSPREY 3.

popular package manager PIP. Once installed, using OSPREY 3 is as easy as writing a python script. High-performance computations are still performed in the Java virtual machine to give the fastest runtimes, so Java is still required to run OSPREY 3, but communication between the Python enviroment and the Java environment is handled behind-the-scenes, and OSPREY 3 still looks and feels like a regular Python application.

See Figure 5 for a complete example of a Python script that performs a very simple design using OSPREY 3.

References

[1] VRC 605: A Phase 1 Dose-Escalation Study of the Safety and Pharmacokinetics of a Human Monoclonal Antibody, VRC07-523LS, Administered Intravenously or Subcutaneously to Healthy Adults. ClinicalTrials.gov Identifier:

NCT03015181. NIAID And National Institutes of Health Clinical Center. January (2017). https://clinicaltrials.gov/ct2/show/NCT03015181.

[2] Cheng-Yu Chen, Ivelin Georgiev, Amy C. Anderson, and Bruce R. Donald. Computational structure-based redesign of enzyme activity. *Proceedings of the National Academy of Sciences of the USA*, 106(10):3764–3769, 2009.

[3] G. Y. Chuang, H. Geng, M. Pancera, K. Xu, C. Cheng, P. Acharya, M. Chambers, A. Druz, Y. Tsybovsky, T. G. Wanninger, Y. Yang, N. A. Doria-Rose, I. S. Georgiev, J. Gorman, M. G. Joyce, S. O'Dell, T. Zhou, A. B. McDermott, J. R. Mascola, and P. D. Kwong. Structure-Based Design of a Soluble Prefusion-Closed HIV-1 Env Trimer with Reduced CD4 Affinity and Improved Immunogenicity. *J. Virol.*, 91(10), May 2017.

[4] N. A. Doria-Rose, H. R. Altae-Tran, R. S. Roark, S. D. Schmidt, M. S. Sutton, M. K. Louder, G. Y. Chuang, R. T. Bailer, V. Cortez, R. Kong, K. McKee, S. O'Dell, F. Wang, S. S. Abdool Karim, J. M. Binley, M. Connors, B. F. Haynes, M. A. Martin, D. C. Montefiori, L. Morris, J. Overbaugh, P. D. Kwong, J. R. Mascola, and I. S. Georgiev. Mapping Polyclonal HIV-1 Antibody Responses via Next-Generation Neutralization Fingerprinting. *PLoS Pathog.*, 13(1):e1006148, Jan 2017.

[5] Kathleen M. Frey, Ivelin Georgiev, Bruce R. Donald, and Amy C. Anderson. Predicting resistance mutations using protein design algorithms. *Proceedings of the National Academy of Sciences of the USA*, 107(31):13707–13712, 2010.

[6] Pablo Gainza, Kyle Roberts, and Bruce R. Donald. Protein design using continuous rotamers. *PLoS Computational Biology*, 8(1):e1002335, 2012.

[7] Pablo Gainza, Kyle E. Roberts, Ivelin Georgiev, Ryan H. Lilien, Daniel A. Keedy, Cheng-Yu Chen, Faisal Reza, Amy C. Anderson, David C. Richardson, Jane S. Richardson, and Bruce R. Donald. OSPREY: Protein design with ensembles, flexibility, and provable algorithms. *Methods in Enzymology*, 523:87–107, 2013.

[8] I. Georgiev, P. Acharya, S. Schmidt, Y. Li, D. Wycuff, G. Ofek, N. Doria-Rose, T. Luongo, Y. Yang, T. Zhou, B. R. Donald, J. Mascola, and P. Kwong. Design of epitope-specific probes for sera analysis and antibody isolation. *Retrovirology*, 9(Suppl. 2):P50, 2012.

[9] Ivelin Georgiev, Ryan H. Lilien, and Bruce R. Donald. The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. *Journal of Computational Chemistry*, 29(10):1527–1542, 2008.

[10] Ivelin Georgiev, Kyle E. Roberts, Pablo Gainza, Mark A. Hallen, and Bruce R. Donald. OSPREY (Open Source Protein Redesign for You) user manual. Available online: `www.cs.duke.edu/donaldlab/software.php`. Updated, 2015. 94 pages., 2009.

[11] Ivelin S. Georgiev, Rebecca S. Rudicell, Kevin O. Saunders, Wei Shi, Tatsiana Kirys, Krisha McKee, Sijy ODell, Gwo-Yu Chuang, Zhi-Yong Yang, Gilad Ofek, Mark Connors, John R. Mascola, Gary J. Nabel, and Peter D. Kwong. Antibodies VRC01 and 10e8 neutralize HIV-1 with high breadth and potency even with ig-framework regions substantially reverted to germline. *The Journal of Immunology*, 192(3):1100–1106, Feb 2014.

[12] Michael J. Gorczynski, Jolanta Grembecka, Yunpeng Zhou, Yali Kong, Liya Roudaia, Michael G. Douvas, Miki Newman, Izabela Bielnicka, Gwen Baber, Takeshi Corpora, Jianxia Shi, Mohini Sridharan, Ryan Lilien, Bruce R. Donald, Nancy A. Speck, Milton L. Brown, and John H. Bushweller. Allosteric inhibition of the protein-protein interaction between the leukemia-associated proteins Runx1 and CBFβ. *Chemistry and Biology*, 14:1186–1197, 2007.

[13] Mark A. Hallen and Bruce R. Donald. CATS (Coordinates of Atoms by Taylor Series): protein design with backbone flexibility in all locally feasible directions. *Bioinformatics*, 33(14):i5–i12, 2017.

[14] Mark A. Hallen, Pablo Gainza, and Bruce R. Donald. A compact representation of continuous energy surfaces for more efficient protein design. *Journal of Chemical Theory and Computation*, 11(5):2292–2306, 2015.

[15] Mark A. Hallen, Jonathan D. Jou, and Bruce R. Donald. LUTE (Local Unpruned Tuple Expansion): Accurate continuously flexible protein design with general energy functions and rigid-rotamer-like efficiency. In *International Conference on Research in Computational Molecular Biology*, pages 122–136. Springer, 2016.

[16] Nathan H Joh, Tuo Wang, Manasi P Bhate, Rudresh Acharya, Yibing Wu, Michael Grabe, Mei Hong, Gevorg Grigoryan, and William F DeGrado. De novo design of a transmembrane zn-transporting four-helix bundle. *Science*, 346(6216):1520–4, Dec 2014.

[17] Anthony Nicholls and Barry Honig. A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. *Journal of Computational Chemistry*, 12(4):435–445, 1991.

[18] Adegoke Ojewole, Anna Lowegard, Pablo Gainza, Stephanie M. Reeve, Ivelin Georgiev, Amy C. Anderson, and Bruce R. Donald. OSPREY predicts resistance mutations using positive and negative computational protein design. In *Computational*

Protein Design, volume 1529 of *Methods in Molecular Biology*. Humana Press, New York, 2017. In press.

[19] Andrew S Parker, Yoonjoo Choi, Karl E Griswold, and Chris Bailey-Kellogg. Structure-guided deimmunization of therapeutic proteins. *J Comput Biol*, 20(2):152–65, Feb 2013.

[20] Stephanie M. Reeve, Pablo Gainza, Kathleen M. Frey, Ivelin Georgiev, Bruce R. Donald, and Amy C. Anderson. Protein design algorithms predict viable resistance to an experimental antifolate. *Proceedings of the National Academy of Sciences of the USA*, 112(3):749–754, 2015.

[21] Kyle E. Roberts, Patrick R. Cushing, Prisca Boisguerin, Dean R. Madden, and Bruce R. Donald. Computational design of a PDZ domain peptide inhibitor that rescues CFTR activity. *PLoS Computational Biology*, 8(4):e1002477, 2012.

[22] Walter Rochia, Sundaram Sridharan, Anthony Nicholls, Emil Alexov, Alessandro Chiabrera, and Barry Honig. Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: Applications to the molecular systems and geometric objects. *Journal of Computational Chemistry*, 23(1):128–137, 2002.

[23] Rebecca S. Rudicell, Young Do Kwon, Sung-Youl Ko, Amarendra Pegu, Mark K. Louder, Ivelin S. Georgiev, Xueling Wu, Jiang Zhu, Jeffrey C. Boyington, Xuejun Chen, Wei Shi, Zhi-Yong Yang, Nicole A. Doria-Rose, Krisha McKee, Sijy O'Dell, Stephen D. Schmidt, Gwo-Yu Chuang, Aliaksandr Druz, Cinque Soto, Yongping Yang, Baoshan Zhang, Tongqing Zhou, John-Paul Todd, Krissey E. Lloyd, Joshua Eudailey, Kyle E. Roberts, Bruce R. Donald, Robert T. Bailer, Julie Ledgerwood, NISC Comparative Sequencing Program, James C. Mullikin, Lawrence Shapiro, Richard A. Koup, Barney S. Graham, Martha C. Nason, Mark Connors, Barton F. Haynes, Srinivas S. Rao, Mario Roederer, Peter D. Kwong, John R. Mascola, and Gary J. Nabel. Enhanced potency of a broadly neutralizing HIV-1 antibody *in vitro* improves protection against lentiviral infection *in vivo*. *Journal of Virology*, 88(21):12669–12682, 2014.

[24] Regina S Salvat, Yoonjoo Choi, Alexandra Bishop, Chris Bailey-Kellogg, and Karl E Griswold. Protein deimmunization via structure-based design enables efficient epitope deletion at high mutational loads. *Biotechnol Bioeng*, Feb 2015.

[25] Doree Sitkoff, Kim A. Sharp, and Barry Honig. Accurate calculation of hydration free energies using macroscopic solvent models. *Journal of Physical Chemistry*, 98:1978–1988, 1994.

[26] Brian W. Stevens, Ryan H. Lilien, Ivelin Georgiev, Bruce R. Donald, and Amy C. Anderson. Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme's mechanism and selectivity. *Biochemistry*, 45(51):15495–15504, 2006.

[27] Christina L. Vizcarra, Naigong Zhang, Shannon A. Marshall, Ned S. Wingreen, Chen Zeng, and Stephen L. Mayo. An improved pairwise decomposable finite-difference Poisson-Boltzmann method for computational protein design. *Journal of Computational Chemistry*, 29(7):1153–1162, 2008.

[28] Hongliang Zhao, Deeptak Verma, Wen Li, Yoonjoo Choi, Christian Ndong, Steven N Fiering, Chris Bailey-Kellogg, and Karl E Griswold. Depletion of T cell epitopes in lysostaphin mitigates anti-drug antibody response and enhances antibacterial efficacy in vivo. *Chem Biol*, 22(5):629–39, May 2015.