Published in final edited form as:

Nat Biotechnol. 2013 February ; 31(2): 108–111. doi:10.1038/nbt.2495.

Prize-Based Contests Can Provide Solutions to Computational Biology Problems

Karim R Lakhani^{1,2,*}, Kevin J Boudreau^{2,3,*}, Po-Ru Loh⁴, Lars Backstrom⁵, Carliss Baldwin¹, Eric Lonstein¹, Mike Lydon⁵, Alan MacCormack¹, Ramy A Arnaout^{6,7,#}, and Eva C Guinan^{7,8,#}

¹Harvard Business School, Boston, Massachusetts, USA

²Harvard-NASA Tournament Lab, Institute for Quantitative Social Science

³London Business School, London, UK

⁴Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

⁵TopCoder.com, Glastonbury, Connecticut, USA

⁶Department of Pathology and Division of Clinical Informatics, Department of Medicine, Beth Israel Deaconess Medical Center, Boston, Massachusetts, USA

⁷Harvard Medical School, Boston, Massachusetts, USA

⁸Department of Radiation Oncology, Dana-Farber Cancer Institute, Boston, Massachusetts, USA

Advances in biotechnology have fuelled the generation of unprecedented quantities of data across the life sciences. However, finding individuals who can address such “big data” problems effectively has become a significant research bottleneck. Historically, prize-based contests have had striking success in attracting unconventional individuals who can solve difficult challenges. To determine whether this approach could solve a real “big data” biologic algorithm problem, we used a complex immunogenomics problem as the basis for a two-week online contest broadcast to participants outside academia and biomedical disciplines. Participants in our contest generated over 600 submissions containing 89 novel computational approaches to the problem. Thirty submissions exceeded the benchmark performance of NIH’s MegaBLAST. The best achieved both greater accuracy and speed (x1000). Here we show the potential of using online prize-based contests to access individuals without domain-specific backgrounds to address big data challenges in life sciences.

Corresponding Author: Eva C. Guinan, MD, Dana-Farber Cancer Institute, 450 Brookline Avenue, Boston, MA 02215, eva_guinan@dfci.harvard.edu.

*these authors contributed equally,

#these authors contributed equally

Author Contributions: KRL, KJB and ECG conceived, designed and executed the experiment, analyzed data and wrote the manuscript. CB and AM contributed to experimental design. EL executed the experiment and collected data. RAA identified and co-developed the immunogenomics problem, tested the submissions and helped write the manuscript. ML and LB co-developed the problem statement and test data. PRL analyzed and categorized all submission data and helped write the manuscript.

The advent of high-throughput biology has resulted in studies that routinely generate gigabytes of data. Zerbino *et al.* (2012)¹ estimate that genome data available for analysis will grow from petabytes (10^{15}) to exabytes (10^{18}). Finding individuals with the necessary training to analyze such “big data” has become a bottleneck in moving from sequencing to discovery². This supply-demand problem extends beyond genomics to other life science areas^{3, 4, 5} and to many diverse industries. It has been projected that by 2018 there will be shortage of approximately 200,000 data scientists and 1.5 million individuals, in the US Economy, with sufficient training and skills to conceptualize and manage big data analyses⁶.

In academia and elsewhere, this bottleneck is more than just a personnel shortage. There may be a mismatch between a specific computational research need and available local expertise, academic incentives often encourage experts to focus on their own research rather than collaborative projects, and available personnel may lack experience with the specific approach or techniques required. In light of this bottleneck, as an alternative to engaging in an extensive search to identify and contract with potentially suitable individuals, prize-based contests^{7, 8, 9, 10, 11} have emerged as a novel approach to find solutions to challenging problems in settings as diverse as industrial R&D, software development, and internet commerce. Such contests are one part of a decade-long trend toward solving science problems through large-scale mobilization of individuals by what the popular press refers to as “crowdsourcing”¹². In general, crowdsourcing has come to imply a strategy that relies on external, unaffiliated actors to resolve a particular problem. It encompasses a range of approaches intended to accomplish tasks from rote mechanical to highly intellectual problem solving. Strategies can enable cooperation and knowledge sharing amongst participants or create competitions, may limit entry to specified communities or allow universal participation, and may offer pecuniary and/or non-pecuniary incentives. Lastly, strategies may necessitate creation and utilization of complex infrastructure such as specialized scientific gaming websites or simply require internet access.

Computational biology has become a fertile ground for experimentation with various crowdsourcing approaches. One approach is to transform the problem into a game that non-scientists can play without significant knowledge of the underlying scientific principles. For example, visual representation of molecular interactions in the Foldit game format has attracted participation by individuals without training in molecular biology or biochemistry to solve protein structure prediction problems that have eluded resolution by systematic, expert research programs^{13, 14, 15}. Additional tools permit players to generate protein-folding algorithms, the best of which is equivalent to those emerging from academia¹⁶. Other efforts have focused on soliciting solutions from a larger expert community by moving questions beyond one laboratory or institution. For example, CAGI (critical assessment of genome interpretation)¹⁷ focuses on analyses of real data to predict biologically-relevant information and the CLARITY¹⁸ challenge asks scientific teams to sequence 3 patient genomes (i.e. teams must have access to sequencing infrastructure), identify the variants and develop clinical reporting formats.

Over the last ten years, online prize-based contest platforms have emerged to solve specific scientific and computational problems for the commercial sector. These platforms, with solvers in the range of tens to hundreds of thousands have achieved considerable success by

exposing thousands of problems to larger numbers of heterogeneous problem-solvers and by appealing to a wide range of motivations to exert effort and create innovative solutions^{19, 20}. The large number of entrants in prize-based contests increases the probability that an “extreme-value” (or maximally performing) solution can be found through multiple independent trials; also known as a parallel-search process²⁰. In contrast to traditional approaches, where experts are predefined and pre-selected, contest participants self-select to address problems and typically represent a diversity in knowledge, skills and experience that would be virtually impossible to duplicate locally¹⁹. Thus, the contest sponsor can identify an appropriate solution by allowing many individuals to participate and observing the best performance. This is particularly salient for highly uncertain innovation problems where prediction of the best solver or approach may be difficult and the best person to solve one problem may be unsuitable for another²⁰.

The prize-based contest approach used here differs from other academic crowdsourcing efforts in multiple ways: 1) it uses an existing commercial platform with a built-in membership base of computer-science solvers able to immediately attack the problem and able to deliver submissions in the multiple hundreds as compared to tens for DREAM and CLARITY; 2) sponsors do not need to develop specialized problem-solving infrastructure (e.g. online games interfaces); 3) it is generalizable to any life sciences problem that can be translated into generic computer-science terms (e.g., Foldit is specifically designed to address spatial protein-folding problems); and 4) it delivers working algorithms rapidly (weeks from launch). Yet it remains to be determined if academic biomedicine problems are amenable to solving by these ready-made platforms.

Therefore, we experimented with the application of a prize-based contest to solve a data-rich biological problem related to immune repertoire profiling, in which short, recombined and mutated stretches of genetic sequence had to be annotated according to their constituent gene components^{5, 21, 22, 23}. The specificity of immune cells for particular antigens depends on which antibodies B cells secrete and which T cell receptors (TCRs) T cells express, which in turn depends on the sequence of their antibody or TCR genes. Unlike other genes, those for antibodies and TCRs are not encoded as single genes. Instead, they are built up combinatorially in each cell from gene segments; thus each new cell can have a gene with a different DNA sequence²⁴. Diversity is increased further by insertion or deletion of nucleotides at the junctions (joins) between segments, by mutation in the resulting gene, and/or by combinatorial pairing of the encoded protein product. Thus a relatively small number of gene segments (<100 for antibody heavy chains in humans) can lead to an extraordinary number of different molecules (~10³⁰). An obligate step toward making sense of this diversity is annotating sequence according to which gene segments contribute to each recombined gene. This exercise is particularly challenging because the segments are often short, and recombined genes routinely exhibit numerous insertions, deletions, and substitutions (mutations).

Our goal in the prize contest was the creation of an algorithmic solution with better performance characteristics than sequence annotation approaches using established methods such as BLAST²⁵ or IMGT/V-QUEST²⁶. As a typical sequencing run for these genes produces on the order of 10⁵ sequences,^{5, 21, 22, 23} we sought solutions that could annotate

this many sequences, with at least comparable accuracy to existing solutions, in much less time, which we defined as 30 seconds on an off-the-shelf desktop computer with 1GB memory.

Three crucial steps were taken to transform the highly domain-specific immunogenomics problem into a challenge that would be of interest and attractive to non-life science solvers. First, we rephrased our problem in generic terminology: given 10^5 strings (gene sequences), each generated as the union of three substrings (gene segments; one from each of three sets A, B, and C of known substrings), with multi-letter (polynucleotide) insertions and deletions at the junctions and substitutions in the final string, write and implement an algorithm that determines which three substrings contribute to each string. This produced a problem statement devoid of biological concepts and presented an information theory and string processing task that a computational expert could tackle. The crucial element was removal of all context specific information and requirements for using existing, preferred approaches so that solvers with heterogeneous backgrounds had the freedom to apply their own diverse perspectives and heuristics to create their solutions²⁷. Second, we assembled the required test data for solution generation and scoring. Standard practice in this contest platform is to generate three independent test suites: a public training set for contestants, a private set to enable real-time algorithm evaluation by contestants and, to prevent over-fitting, a private set for final submission scoring by the contest administrators. Our test sets used all known antibody heavy-chain V, D, and J gene segments and the nucleotide insertion, deletion, and substitution (mutation) frequencies observed in actual antibody heavy-chain sequences⁵. The development of test suites requires significant care as contestants will carefully examine the data's nuances and characteristics, and artifacts in the data that may be spuriously correlated with results will be discovered and likely exploited for competitive advantage. Third, we devised a linear function scoring metric that supported our goal of achieving both improved accuracy and computational efficiency (speed), and disclosed this metric and its components to the contestants. The score was the only metric used to award prizes. All information made available to participants is detailed in Supplementary Materials.

We ran our contest on the TopCoder.com online programming competition website, a commercial platform that had the advantage of providing us with an existing community of "solvers". Established in 2001, TopCoder currently has a community of over 400,000 software developers that compete regularly to solve programming challenges¹³. Our contest ran for two weeks and offered a \$6000 prize pool, with top ranking players receiving cash prizes of up to \$500 each week. Our challenge drew 733 participants, of whom 122 (17%) submitted software code. This group of submitters, drawn from 69 countries, included roughly half (44%) professionals with the remainder being students at various levels. Most participants were between 18 and 44 years old. None were academic or industrial computational biologists, and only five described themselves as coming from either R&D or life sciences in any capacity.

Consistent with usual practices in algorithm and software development contests, participants were able to make multiple code submissions to enable testing of solutions and participant learning and improvement. Collectively, participants submitted 654 solutions, averaging to 5.4 submissions per participant. Participants reported spending an average of 22 hours each

developing solutions, for a total of 2,684 hours of development time. Final submissions that received cash awards are available for download under an open source license (see Supplementary Materials).

Apart from the use of test suites to rank order the solutions, we also evaluated the accuracy and speed of each participant's final submission by testing it on an *in silico* set of 10^5 antibody heavy chain sequences (using the same seed sequences as in the contest test suite) on a desktop computer (2009 iMac Intel Core i5 2.66 GHz with 8GB RAM; Figure 1). We tested these submissions directly against existing "industry standard" algorithmic solutions. An approach based on NCBI's MegaBLAST (Figure 1, square) and our co-author's (RAA) custom annotation software, idAb (Figure 1, dark circle) served as benchmarks (see also Supplementary Material)⁵. Sixteen of the 122 submissions outperformed the accuracy (77%) of the idAb solution and 30 outperformed the MegaBLAST benchmark for accuracy (72%). Furthermore, eight submissions achieved an 80% accuracy score, which is very near the theoretical maximum for the dataset. The remaining error corresponds to sequences that cannot be correctly annotated, either due to removal of D-segment sequence, truncation, or mutation (see scoring, Supplementary Material).

There was also a notable improvement in speed over both benchmark algorithms. Submissions that were at least as accurate as the benchmarks ran on average 30 times faster than idAb on an expanded test suite of 1 million sequences (89 seconds vs. 2,845 seconds) and 175 times faster than MegaBLAST (89 seconds vs. 15,623 seconds). The three fastest submissions ran in 16 seconds—178 times faster than idAb and nearly 1,000 times faster than MegaBLAST. Like idAb and MegaBLAST, algorithms from these top performers have runtimes that scale linearly with the number of sequences. Thus, these submissions can annotate 10 million sequences in under three minutes and nearly a quarter-billion sequences per hour on a typical desktop machine, demonstrating their potential to scale with constantly improving sequencing technologies.

To investigate the specific technical approaches developed by contestants, we commissioned three independent computer science PhD researchers to review all submissions and determine what techniques were implemented. Their analyses determined that 10 distinct elemental methods (Table 1) were used in 89 combinations in the 654 submissions. As the number of elemental methods in a submission increased so did its performance (Figure 2) with leaderboard scores increasing by 85.3 points for each additional method employed ($p < 0.01$). Analysis of the benchmark algorithms showed that methods numbered 2, 3, 5 and 8 were implemented in the MegaBLAST algorithm while methods numbered 2, 4 and 7 were implemented in the idAb code.

Thus, the results achieved by contest participants in only fourteen days improved significantly upon the existing solutions available to academic researchers, decreasing processing time by up to three orders of magnitude with accuracy reaching the theoretical maximum. Moreover, thirty different solutions improved upon the state of the art exemplified by both an off-the-shelf, general purpose tool widely used by the academic community (MegaBLAST) and software developed by a single team addressing the identical problem (idAb), suggesting that prize-contests are a robust, reliable approach to efficiently

generate desired solutions. While the solvers were virtually devoid of domain-specific knowledge, abstracting the problem into general algorithmic and mathematical terms allowed a wide range of non-domain experts to address an important, complex problem. These contestants brought to the problem whatever skills and expertise they had or could find, likely yielding a far more diverse toolkit than would be available locally, and generated significant diversity in technical approaches. Accessing such diversity may be particularly important, as big data biomedical analytics is a rapidly evolving field in which it is difficult to know a priori the kind, quality and breadth of expertise needed to generate an effective solution.

In summary, we show that a prize-based contest on a commercial platform can effectively recruit skilled individuals to apply their knowledge to a big-data biomedical problem. Deconstruction and transformation of problems for a heterogeneous solver community coupled with adequate data to produce and validate results can support solution diversity and minimize the risk of suboptimal solutions that may arise from limited searches. In addition to the benefits of generating new knowledge, this strategy may be particularly useful in situations where the computational or algorithmic problem, or potentially any science problem, represents a barrier to rapid progress but where finding the solution is not itself the major thrust of the investigator's scientific effort. The America Competes Act passed by Congress provides funding agencies with the authority to administer their own prize-based contests and paves the way for establishing how grant recipients might access commercial prize platforms to accelerate their own research.

Methods

Test Set Generation

Each test set, a V, D, and J segment were chosen at random from human or mouse antibody or TCR gene segments. (This was done to encourage solutions that would work well across species and receptor types.) A random number of bases was removed at the VD and DJ junctions and a random number of bases inserted at these junctions. These deletions and insertions simulate biological processes involved in antibody and TCR generation; in all cases the random numbers were chosen according to known distributions. (Note that because D segments are often short, this process sometimes removes all trace of a constituent D segment.) Next, a fraction of bases were mutated by changing them at random to other bases. This process was repeated 10^6 times to produce a data set representative of what a real-world immune-profiling experiment might produce.

Submission Scoring

To select for submissions that were both fast and accurate, we scored them according to both quality and time (see Problem Statement, Supplementary Materials). Finally, we evaluated the best performers according to a real-world test. In an actual experiment, the necessary first step is usually to annotate the recombined, mutated sequences according to what V, D, and J segments gave rise to them. The best annotation algorithm will correctly identify the greatest possible number of constituent V, D, and J sequences. Therefore we tested winners according to the percent of V, D, and J annotations that matched the V, D, and J segments

used as input. So, for each sequence, possible scores are 0, 33, 66, and 100% depending on whether 0, 1, 2, or all 3 segments are correctly identified. We calculated the cumulative score for 10^6 sequences in this way. Note we expected the best possible score to be less than 100%, for several reasons: because of the effect of deletions on the D segment; because truncation sometimes removes the only part of a segment that makes it detectably differ from another, similar segment; and because mutation can sometimes convert one segment into another or else result in the sequence being equally similar to multiple segments (leading to several equally good annotations, all but one of which are, nonetheless, the wrong input segment). Based on previous work⁵ we expected a maximum score no lower than 75%.

MegaBLAST Implementation

We implemented our MegaBLAST approach using the simplest possible procedure. First, we used formatdb to build three blast databases, one for each segment family (V, D, J or, generically, a, b, c). Next, we ran MegaBLAST on the unknowns using each database in turn. We scored the results as described.

An alternative simple approach was to build a single database including all possible combinations; however N-nucleotide addition results in potentially extensive insertions in that case and was therefore considered suboptimal. A third approach, of building a database of all possible *sequences*, would have resulted in a very large database and so was not considered.

Scoring details for real-world test

Performance was scored as the total percent of correct segments, as follows. For each sequence, if a segment assignment matched the identity of the segment used to create the sequence, it was counted as “correct.” Thus, the maximum score for a single sequence was $3/3 = 100\%$. Note that because mutation can convert one segment into another, it is possible that the best match is nonetheless incorrect by this scoring system. For example, if two starting segments were $a1=ATGCC$ and $a2=ATTGC$, and $a1$ was used to create a sequence, but then mutation changed the G to a T, an algorithm that identified this fragment as $a2$ would be incorrect. Thus, the maximum possible score under mutation will be less than 100%.

Acknowledgments

Harvard Business School’s Division of Research and Faculty Development funded the prize money and supported KRL, EL, CB and AM. KRL and KJB were also supported by the NASA Tournament Lab which is funded by the NASA Human Exploration and Operations Mission Directorate. ECG was supported by the Harvard Clinical and Translational Science Center (NCRR 5UL1RR025758 and 5UL1RR025758-S and RAA was supported by the Klarman Family Foundation. PRL was supported by NDSEG and NSF graduate research fellowships

References

1. Zerbino DR, et al. Science. 2012; 336(6078):179–182. [PubMed: 22499938]
2. Tan TW, et al. BMC Genomics. 2009; 10(Suppl 3):S36. [PubMed: 19958501]
3. Wollman R, Stuurman N. J Cell Sci. 2007; 120(Pt 21):3715–3722. [PubMed: 17959627]
4. Joyce A, Palsson B. Nat Rev Mol Cell Biol. 2006; 7:198–210. [PubMed: 16496022]

5. Arnaout R, et al. PLoS One. 2011; 6(8):e22365. [PubMed: 21829618]
6. Brown B, Chui M, Manyika J. McKinsey Quarterly. 2011; 4:24–35.
7. Taylor CR. Amer Econom Rev. 1995; 85(4):872–890.
8. Scotchmer, S. Innovation and Incentives. MIT Press; Cambridge, MA: 2004.
9. Levine DK. Eyes on the Prize? Science. 2009; 323(5919):1296–1297. [PubMed: 19265006]
10. Terwiesch C, Xu Y. Manage Sci. 2008; 54(9):1529–1543.
11. Travis J. Science. 2008; 319(5871):1750–1752. [PubMed: 18369115]
12. Howe, J. Crowdsourcing. Crown Books; New York: 2008.
13. Eiben CB, et al. Nat Biotechnol. 2012; 30(2):190–192. [PubMed: 22267011]
14. Khatib F, et al. PNAS. 2011; 108(47):18949–18953. [PubMed: 22065763]
15. Cooper S, et al. Nature. 2010; 466(7307):756–760. [PubMed: 20686574]
16. Khatib F, et al. Nature Struct & Molec Bio. 2011; 18(10):1175–7. [PubMed: 21926992]
17. <https://genomeinterpretation.org>
18. Scudellari M. Nature Medicine. 2012; 18(326)
19. Jeppesen L, Lakhani KR. Organ Sci. 2010; 19:1016–1033.
20. Boudreau KJ, Lacetera N, Lakhani KR. Manage Sci. 2011; 57(5):843–863.
21. Boyd SD, et al. Sci Transl Med. 2009; 1:12ra23.
22. Weinstein JA, et al. Science. 2009; 324(5928):807–810. [PubMed: 19423829]
23. Robins HS, et al. Sci Transl Med. 2010; 2(47):47ra64.
24. Jung D, et al. Annu Rev Immunol. 2006; 24:541–570. [PubMed: 16551259]
25. Altschuler SF, et al. J Mol Biol. 1990; 215(3):403–410. [PubMed: 2231712]
26. Brochet X, et al. Nucleic Acids Res. 2008; 36(Web Server issue):W503–508. [PubMed: 18503082]
27. Hong L, Page S. PNAS. 2004; 101(46):16385–16389. [PubMed: 15534225]

Figure 1.

Accuracy score vs speed of contest-commissioned immunoglobulin sequence annotation code. Shown are accuracy vs speed (computational time) for 70 final submissions (circles), MegaBLAST (triangle), and in-house code (square) processing 100,000 in silico-generated recombined VDJ sequences. Accuracy score is the fraction of gene segment annotations that matched the gene segment used in generating the sequence. Note that because sequence contains mutations, the best annotation might differ from the segment used, resulting in a maximum-possible accuracy of <1.0.

Figure 2. Solution Quality versus Number of Techniques. The unit of observation is the individual submission (654 solutions submission by 122 submitters). The problem-solving score is an amalgam of solution quality and time to execute used to generate the leaderboard results during the contest (details in Methods, Scoring). The curve is a locally-weighted polynomial fitted curve.

Table 1

Elemental techniques used (in combinations) by contestant participants.

Method	Description
1	Filtering by ungapped alignment score (Hamming distance): Compare the query string against strings from sets A, B, and/or C, trying various possible offsets.
2	Filtering by comparing frequencies of hashed chunks: For both the query string and strings from A, B, and/or C, move a sliding window across the string and make a frequency table of the chunks that appear in the window, optionally after hashing the chunks. Select the best match(es) between the frequency table obtained from the query and those from the corpus.
3	Dynamic programming: Compute the actual Levenshtein distance between a portion of the query and strings from sets A, B, and/or C.
4	Dynamic programming extended to more than one section (A, B, C) at once: Extend the dynamic programming Levenshtein distance computation to find the optimal edit distance between (a portion of) the query and all possible A+B, B+C, or A+B+C combinations.
5	Bit optimizations: Use bitwise arithmetic to operate on multiple characters at a time.
6	SSE optimizations: Use Streaming SIMD Extensions (a CPU instruction set enabling single-instruction multiple- data “SIMD” parallelization) to process up to 16 characters or strings at once.
7	Refinement of choices after finding initial solution: As a post-processing step, hold two of the three selections fixed and re-optimize the third.
8	Fast approximation of edit distance in well-matched regions: Use restricted dynamic programming, Hamming distance, or variants thereof to speed up the computation.
9	Precomputation of statistics on the string corpus: Perform offline analysis of the provided sets A, B, and C and use the data obtained for decision-making in the algorithm.
10	Explicitly prefer shorter B strings: In heuristic approaches, give bonuses to shorter strings from set B (which empirically have greater likelihood of producing high scores).