GENOPT 2017: GENeralization-based challenge in global OPTimization Second edition

http://genopt.org/

Manifesto and Participation Instructions

Roberto Battiti *Yaroslav Sergeyev* *Mauro Brunato* *Dmitri Kvasov*

Abstract

While comparing results on benchmark functions is a widely used practice to demonstrate the competitiveness of global optimization algorithms, fixed benchmarks can lead to a negative *data mining* process. The motivated researcher can "persecute" the algorithm choices and parameters until the final designed algorithm "confesses" positive results for the specific benchmark.

To avoid the negative data mining effect, the GENOPT contest will be based on randomized function generators, designed for scientific experiments, with fixed statistical characteristics but individual variation of the generated instances.

The generators will be made available to the participants to test off-line and online tuning schemes, but the final competition will be based on random seeds communicated in the last phase through a cooperative process.

A dashboard will reflect the current ranking of the participants, who are encouraged to exchange preliminary results and opinions.

The final "generalization" ranking will be confirmed in the last competition phase. A scientific presentation and discussion of the methods and results (going deeper than the raw ranking) will be part of a dedicated workshop at LION 10 conference.

To summarize, the GENOPT challenge has the following goals:

Scientific learning and dynamic scoring The challenge is a joint learning effort continuing for some weeks. Non-experts are encouraged to enter, to see preliminary results, and learn from their better-performing peers.

Open results Partial results are visible on a public leaderboard, final results of the better techniques are presented in scientific publications, benchmark functions will remain available also for future experimentation.

Diversity of approach A dangerous habit of some communities is to encourage only internal comparisons (GA against GA, PSO versus PSO, global optimization schemes against other GO schemes). On the contrary, the larger the diversity of approach, the faster the scientific progress and the happier the final users of the optimization schemes.

Seamless participation A researcher minimizes the effort in participating in the challenge, so that he can concentrate on producing novel research. While sections 2 and 3 motivate and detail all competition choices and all actions, the actual instructions to be followed are summarized in Sec. 4.

1 Objectives of GENOPT

This document introduces the second edition of the GENOPT challenge; the first edition was held in 2016 [1].

In this short document we summarize the main motivations for organizing GENOPT with some representative papers (without any ambition of completeness) which can be useful as warmup exercises before participating.

Experimental analysis of algorithms is essential It is now clear that algorithm design for the global optimization of many problems can profit from *scientific experimentation* in a way similar to that of more traditional sciences like Physics.

In fact, the last twenty-thirty years of Computer Science research produced an abundance of *negative results* about the possibility to solve many problems exactly within acceptable (polynomial) computational times [2]. In some cases, even a guaranteed (worst-case) approximation for some problem is intractable.

Heuristics, methods without formal guarantees of convergence in acceptable CPU times, passed from being considered "social pariahs in the algorithmic society" to the only practical way of delivering improving solutions.

Most theoretical results about global optimization are of limited practical applicability, limited to special problems and specific - often unrealistic - assumptions. As an example, *asymptotic convergence* - the Holy Grail of global optimization - is easy to obtain for every method by adding some stochasticity[3] (like the generation of a random point with uniform probability from time to time) but it can be painfully slow and frankly irrelevant for the practitioners.

An experimental approach to algorithmics is advocated in [4]. Some software, tests and applications of global optimization are discussed for example in [5, 6].

Luckily, the hype surrounding magic general-purpose optimization strategies has disappeared. [7] demonstrates a "no free lunch" theorem: if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems. Only an appropriate matching between the characteristics of a problem (or of a specific instance) and the characteristics of an algorithm can lead to efficient

and effective results. If the structural characteristics of the instance are not known *a priori*, learning the relevant features in real-time and *reacting* by self-tuning algorithm parameters is a possible approach, as proposed in the Reactive Search Optimization (RSO) framework [8]. Examples of massive comparisons of complete global optimization solvers are [9, 10].

Scientific experimentation is different from pure horse racing Citing from [11] if "the whole affair is organized around an algorithmic race whose outcome determines the fame and fate of the contestants", researchers are wasting time. Scientific experimentation is much more than pure competition. According to[12], it requires a clear definition of the goals of the experiments, a careful design of the experiments and measures, an analysis leading to insight (understanding). Reproducibility is another essential scientific ingredient: the experiments must be reproducible without resorting to the mind and expertise of a specific researcher. In addition, suitably preprocessed and normalized results, like the performance profiles mentioned in [13], are critical to convey meaning. Tons of tables should be digested by the intelligent researcher to distill the novel discoveries. Make meaning!

Randomized function generators designed for experiments If test functions are fixed for a long time, motivated researchers will elicit reasonable results even from the poorest algorithm, though extensive hand-made parameter and algorithm tuning. To avoid this incestuous relationship between test problems, algorithms and researchers, randomized generators can help to distinguish between a tuning (learning) phase and a testing (generalization) phase. All contributors to GENOPT will be asked to set their solver meta-parameters just once for the entire class of test models to be solved as a batch. Self-tuning is of course allowed but without any human intervention.

Some papers discussing function generators intended for experiments are for example: [14] (with a tentative qualitative classification of test problems), [15] (focusing on the inspiring multi-dimensional scaling via stress minimization problem), [16] (simple functions with controllable smoothness, number and positions of local minima).

To start the GENOPT challenge, at least the following two generators will be provided as software with appropriate wrappers for different programming languages: [17] (functions with known minima and radius of attraction), [18] Radial Basis Functions to simulate structure at different scales.

Additional test functions will be added during the challenge. Motivated suggestions by participants are welcome.

Competition and collaboration can be combined Competition and prizes for the advancement of science have a long tradition. The Longitude reward was offered by the British government for a simple and practical method for the precise determination of a ship's longitude at sea in 1714. More recently,

a start-up company (Kaggle) exploits the concept of "crowdsourcing" in a novel way by organizing competitions in machine learning [19]. Prizes are even considered as a serious alternative to the patent system.

Our GENOPT challenge offers only symbolic prizes and can be considered as a sport event where all participants are gaining knowledge and advancing the state of the art, also by mutual collaboration and interaction. A deeper understanding of the relationship between problem/instance structure and innovative algorithms is the real prize we aim at.

2 Test functions

This section describes the final phase of the contest. Functions to be optimized are broadly divided into three *function families*: GKLS, conditioned transforms of classical benchmarks, and composition of classical benchmarks.

Every family is divided into (at least) six *function types*, which differ by their analytical definition and by the number of dimensions. Functions belonging to the same type share most properties and can be assumed to have the same difficulty.

Finally, every function type of every family is realized into an unlimited number of function *instances* which differ by some randomly generated parameters. In particular, every instance will have a randomly generated offset $c \in [-1, 1]$ added to the function's output in order to make the global minimum value unpredictable.

2.1 GKLS

The first family of functions is obtained by the GKLS generator [17], a procedure for generating three types (non-differentiable, continuously differentiable, and twice continuously differentiable) of classes of test functions with known local and global minima for multiextremal multidimensional box-constrained global optimization.

The six function types in this family are:

- Continuous, non differentiable, $D = 10, 30$;
- Continuously differentiable, $D = 10, 30$;
- Twice differentiable, $D = 10, 30$.

The number of minima depends on the domain dimensionality: $n = 5D$. The other parameters are fixed:

- radius of global minimum basin $r = 1/3$;
- distance of global minimizer $r = 2/3$;
- global minimum (before offsets are applied) $m = -1$.

Individual instances will differ by the randomly generated positions of minima and by the value of the output offset c.

2.2 Conditioned transforms of classical benchmarks

The second family of functions is based on classical continuous optimization benchmarks, of differing difficulty, with an initial mildly-conditioned transform that stretches individual directions by factors up to 3, or contracts them by factors up to $1/3$ with an overall contition number equal to 9. Six different function types are proposed:

- Rosenbrock (unimodal, narrowing bending valley), $D = 10, 30$;
- Rastrigin (strongly multimodal), $D = 10, 30$;
- Zakharov, (unimodal) $D = 10, 30$.

Input $\mathbf{x}$ is subject to a transform, different for every instance,

$$\mathbf{x}' = M\mathbf{x} + \mathbf{x}_0$$

where $\mathbf{x}_0 \in [-0.1, 0.1]^D$ is a small random translation (with the guarantee that the global minimum will be within the definition interval) and M is an orthogonal (not normal) matrix with condition number equal to 100 thus obtained:

$$M = X^T \Lambda X,$$

where X is an orthonormal random matrix (obtained by applying the Householder or Gram-Schmidt procedure to a set of random vectors), $\Lambda = \text{diag}(\lambda_1, \ldots \lambda_D)$ and

$$\lambda_i = 3^{2\frac{D-i}{D-1}-1},$$

so that $\lambda_1 = 10$, $\lambda_D = 0.1$, and all eigenvalues are (logarithmically) equally spaced.

Instances differ by the transform parameters M, $\mathbf{x}_0$ and by the output offset c.

2.3 Composite functions

The third function family is obtained by composing the same continuous optimization benchmarks, generating six different function types, three having $D = 10$, the other three having $D = 30$. Every function type is built by randomly selecting n classical functions $f_1, \ldots, f_n$ from the following set:

- Goldstein-Price, $D_f = 2$;
- Hartmann, $D_f = 3; 6$;
- Rosenbrock, $D_f = \text{rand}(3, D/2)$;
- Rastrigin, $D_f = \text{rand}(3, D/2)$;
- Sphere, $D_f = \text{rand}(3, D/2)$;

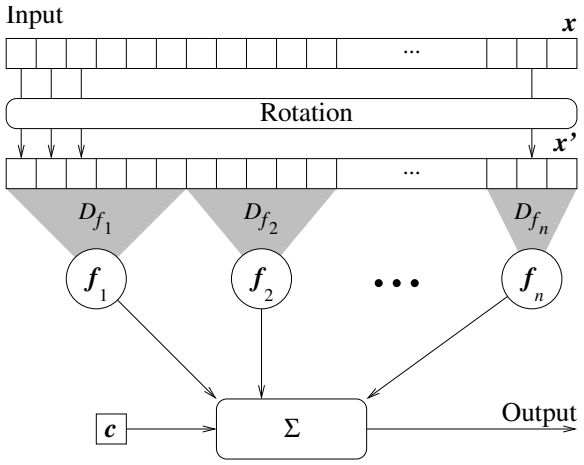

Fig. 1: Composition of functions

- Zakharov, $D_f = \text{rand}(3, D/2)$.

The sum of the function dimensionalities $D_{f_1}, \ldots, D_{f_n}$ must be such that $\sum_i D_{f_i} = D$. Every instance of the same type will correspond to the same choice of basis functions.

Input $\mathbf{x}$ is subject to a rototranslation, different for every instance:

$$\mathbf{x}' = U\mathbf{x} + \mathbf{x}_0$$

where U is a random orthonormal matrix (obtained by applying Householder or Gram-Schmidt to a random set of vectors) and $\mathbf{x}_0 \in [-0.1, 0.1]^D$ is a small random translation. The instance value is computed as

$$f(\mathbf{x}) = c + \sum_{i=1}^{n} f_i(x'_{b_i}, \ldots, x'_{b_i+D_{f_i}-1}),$$

where $b_i = \sum_{j=1}^{i-1} D_{f_j}$, as shown in Fig. 1.

3 Competition evaluation

The set of function types used in this phase of the competition is given in Table 1. A function instance is identified by a pair of numbers: its type index (first column in Table 1) and a positive integer seed. A submission will be generated by running the optimization algorithm on 100 instances (with seed from 1 to 100) for the first 18 function types (index from 0 to 17) and collecting the 1800 report files into a zip folder.

Tab. 1: List of function types used in the competition. Note that the Composite family is virtually infinite, with even type indexes $12, 14, \ldots$ corresponding to dimension 10 and odd indexes $13, 15, \ldots$ corresponding to dimension 30. Indexes $0, \ldots, 17$ will be used in the initial phase; the composite indexes for the final phase are defined in Section 3.3.

Index	Family	Type	Dimension
0	GKLS	non-differentiable	10
1			30
2		continuously differentiable	10
3			30
4		twice differentiable	10
5			30
6	High condition	Rosenbrock	10
7			30
8		Rastrigin	10
9			30
10		Zakharov	10
11			30
$12 + 2n$	Composite		10
$13 + 2n$			30

3.1 Use of the random number generator

The Mersenne Twister RNG, as implemented in the GKLS source, will be used to generate all parameters in the function library.

For the GKLS family, the RNG will be initialized with the provided seed, a GKLS instance will be created, and finally the value offset c is generated.

For the High-condition family, the RNG will be initialized with the provided seed, and the transform parameters M, $\mathbf{x}_0$ and c will be generated.

For the Composite family, the RNG will be initialized with the function index in order to generate the random selection of functions and dimensions. Then the RNG will be re-initialized with the provided seed to generate the rototranslation parameters U and $\mathbf{x}_0$ and the value offset c.

3.2 Evaluation criteria

The main problem of evaluating heuristics on different target functions is the risk of "comparing apples and oranges:" some functions could dominate the score system just because they are easier than expected (or harder, or with a more or less needle-like minimum). To avoid this, the decision was made to never compare values coming from different function types.

The GENOPT classification mechanism has been designed as a **ranked voting system**: many different rank orders among the various submissions are built ac-

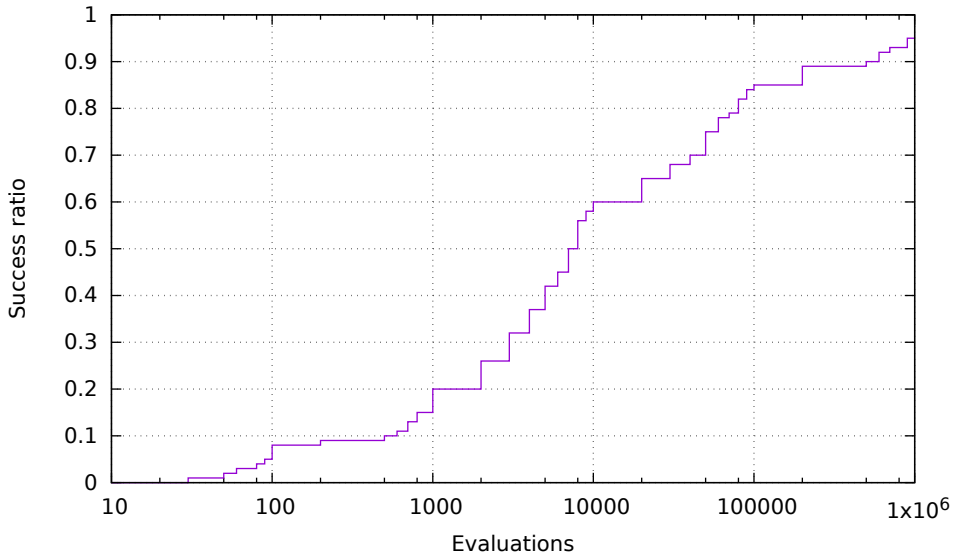

Fig. 2: Cumulative Performance profile sampled at checkpoints $i \cdot 10^j$.

cording to various criteria, with the condition that every criterion be internal to one function type. These "local" rank orders are then combined into the final, global ranking.

Ranking criteria are based on two measures: **record at checkpoints**, and **cumulative performance profiles**.

3.2.1 Record at Checkpoints

The first criterion compares different submissions based on their "typical" performance on a given function type within a given number of function evaluations.

For each of the 18 function types, 4 checkpoints are fixed at 1K, 10K, 100K, and 1M evaluations. For each checkpoint, submissions are ranked on the basis of their median record value across the 100 instances.

The median has been chosen for robustness: the occurrence of a very bad, although very rare, outlier might disrupt an arithmetic mean; moreover, the logarithmic nature of the problem (positive error minimization) does not match a linear scale very well, and a geometric mean would suffer from the opposite problem of having very good outliers condition an undeserved positive result.

3.2.2 Cumulative Performance Profile

Define a search to be *successful* when it finds a point in the domain whose value is within $\epsilon = 10^{-5}$ from the global minimum value. The Cumulative Performance Profile shows the ratio of searches that are successful after a given number of iterations. In detail:

- let f be one of the 18 function types,
- let f_i be its seeded instances ($i = 1, \ldots, 100$);
- let $\mathbf{x}_{ij}$ be the jth evaluation requested by the search algorithm during the optimization of function f_i;
- let flag s_{ij} be 1 iff the optimization of f_i is successful within evaluation j:

$$s_{ij} = \begin{cases} 1 & \text{if } \min_{k=1,\ldots,j} f_i(x_{ik}) < \min f_i + \epsilon \\ 0 & \text{otherwise;} \end{cases}$$

- finally, let the success ratio within the jth iteration over all instances be

$$r_j = \frac{\sum_{i=1}^{100} s_{ij}}{100}.$$

Figure 2 shows a possible Cumulative Performance diagram for a given function type. Success rate is sampled at evaluation numbers in the form $a \cdot 10^b$ with $a = 1, \ldots, 9$ and $b = 1, \ldots, 5$, and for 10^6. More formally, sampling points are

$$p_n = (1 + n\%9) \cdot 10^{1+\lfloor \frac{n}{9} \rfloor}, \qquad n = 0, \ldots, 45$$

We compute a first-order approximation of the area below the OC curve with a logarithmic horizontal coordinate:

$$A \approx \sum_{n=1}^{45} s_{p_n} \big(\log p_n - \log p_{n-1}\big),$$

which is precisely the area below the stepwise curve in Fig. 2

Submissions are ranked separately on the basis of the OC area for each function type.

3.2.3 Combination of the local criteria

Every function type, represented by 100 instances in each submission, provides five separate rankings (four based on the record value at each checkpoint, one on the Cumulative Performance diagram area), so a total number of 90 rank orders are provided.

In order to ensure a reasonably fair treatment of all such criteria, and for simplicity's sake, the final order is distilled by sorting submissions according to their average position across all local rank orders.

Please note that the chosen combination criterion (just like any rank-based system) can sometimes violate the "Independence of Irrelevant Alternatives" principle: in some cases, adding or removing a submission might change the order of existing ones.

3.3 Final phase

The current phase of the competition will end on March 31 at 24:00 GMT. The final phase of the competition will start at the same moment and will be open to all users who made at least one submission in the previous phase.

In this final phase, contestants will be asked not to fine tune their algorithms anymore, but to reapply the same configuration used in the previous phase. A random seed S_0 will be generated in a collaborative way described later, and will be published on the website. Contestants will be asked to run their optimization algorithms on 100 instances (with seeds $S_0, \ldots, S_0 + 99$) of function types $0, \ldots, 11$ and $S_0, \ldots, S_0 + 5$ (i.e., different function compositions will be used wrt the previous phase).

The same ranking criteria will be used as in the previous phase.

3.3.1 Final seed generation

To ensure fairness, the following collaborative, nothing-up-my-sleeves seed generation procedure will be followed:

- every user i will provide a positive integer of his choice $1 \leq p_i \leq 1000$, with the possibility of checking and/or modifying it anytime These values will be frozen on March 31 at 24:00 GMT;
- at the same moment, the base seed for the final test will be computed as

$$S_0 = \texttt{SHA-256}\left(\sum_i p_i\right) \%2^{31},$$

  i.e., the least-significant 31 bits of the `SHA-256` hash of the sum of the user-submitted values;
- the value of S_0 and the user-submitted numbers p_i will be published, so that every user will be able to check his own choice and replicate the generation process.

Since the organizers can in principle know everybody else's numbers, and could therefore manipulate the system with their own submitted values, the organizers won't submit a number (or, equivalently, they will submit zero).

4 Summary of rules for participating

To participate, contestants must:

- If not registered yet, send a message via the message box at

`http://genopt.org/`

asking to receive their account credentials.

- Download the latest version of the genopt library, available at the same page; link their optimization code to the provided API by following the examples and the documentation included in the package (various OSs and languages are supported).
- Execute one optimization run for 1,000,000 function evaluations for every function type from 0 to 17 inclusive and for every seed from 1 to 100 inclusive. Optionally, the 1,000,000 evaluations limit can be set by the appropriate function in the GenOpt library.
- Every run will generate a report file, for a total of $18 \times 100 = 1800$ files. Compress all report files as a ZIP file (the result should be between 1MB and 2MB).
- Login and upload the ZIP file.

At this moment, no limits are enforced on the number of submissions per user. Users can delete their own submissions at any time.

5 Awards

Contestants compete for the following award categories: **High Jump**, **Target Shooting**, and **Biathlon**. Winners will be celebrated at the LION Conference and the awards will remain publicly visible in the GENOPT website.

The **High Jump** prize will be awarded to the submission that typically jumps higher (lower, actually: this is a *minimization* contest) than the others, according to the "Record at Checkpoints" criterion described in Sec. 3.2.1.

The **Target Shooting** prize will be awarded to the submission whose runs typically achieve higher success rates in the quest for the true global minimum, according to the "Cumulative Performance Profile" criterion described in Sec. 3.2.2.

The **Biathlon** prize is awarded to the submission achieving the highest cumulate ranking in the two previous categories, as described in Sec. 3.2.3.

Acknowledgements

We thank Janos Pintér for sending comments and useful suggestions.

References

[1] R. Battiti, Y. Sergeyev, M. Brunato, and D. Kvasov, "GENOPT 2016: Design and results of a GENeralization-based challenge in global OPTimization," in *Proceedings of NUMTA 2016, Numerical Computations: Theory and Algorithms, Pizzo Calabro, Italy 19–25 June 2016*, The American Institute of Physics (AIP) Conference Proceedings, 2016.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability - A guide to the Theory of NP-Completeness*. W.H. Freeman and Company (San Francisco), 1979.

[3] A. Zhigljavsky and A. Žilinskas, *Stochastic global optimization*, vol. 9. Springer Science & Business Media, 2007.

[4] C. C. McGeoch, "Toward an experimental method for algorithm simulation," *INFORMS Journal on Computing*, vol. 8, no. 1, pp. 1–28, 1996.

[5] J. D. Pintér, "Global optimization: software, test problems, and applications," in *Handbook of global optimization*, pp. 515–569, Springer, 2002.

[6] J. D. Pintér and F. J. Kampas, "Benchmarking nonlinear optimization software in technical computing environments," *Top*, vol. 21, no. 1, pp. 133–162, 2013.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, April 1997.

[8] R. Battiti, "Reactive search: Toward self-tuning heuristics," in *Modern Heuristic Search Methods* (V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, eds.), ch. 4, pp. 61–83, John Wiley and Sons Ltd, 1996.

[9] A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinkó, "A comparison of complete global optimization solvers," *Mathematical programming*, vol. 103, no. 2, pp. 335–356, 2005.

[10] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[11] J. Hooker, "Testing heuristics: We have it all wrong," *Journal of Heuristics*, vol. 1, no. 1, pp. 33–42, 1995.

[12] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. Stewart, "Designing and reporting on computational experiments with heuristic methods," *Journal of Heuristics*, vol. 1, no. 1, pp. 9–32, 1995.

[13] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

[14] A. Törn, M. M. Ali, and S. Viitanen, "Stochastic global optimization: Problem classes and solution techniques," *Journal of Global Optimization*, vol. 14, no. 4, pp. 437–447, 1999.

[15] R. Mathar and A. Zilinskas, “A class of test functions for global optimization,” *Journal of Global Optimization*, vol. 5, no. 2, pp. 195–199, 1994.

[16] F. Schoen, “A wide class of test functions for global optimization,” *Journal of Global Optimization*, vol. 3, no. 2, pp. 133–137, 1993.

[17] M. Gaviano, D. E. Kvasov, D. Lera, and Y. D. Sergeyev, “Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 4, pp. 469–480, 2003.

[18] R. Battiti and G. Tecchiolli, “The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization,” *Annals of Operations Research – Metaheuristics in Combinatorial Optimization*, vol. 63, pp. 153–188, 1996.

[19] J. Carpenter, “May the best analyst win,” *Science*, vol. 331, pp. 698–699, 2011.