On the Empirical Evaluation of Las Vegas Algorithms — Position Paper

Holger Hoos[1]
Computer Science Department
University of British Columbia
Email: hoos@cs.ubc.ca

Thomas Stützle
IRIDIA
Université Libre de Bruxelles
Email: tstutzle@ulb.ac.be

Abstract

We advocate a new methodology for empirically analysing the behaviour of Las Vegas Algorithms, a large class of probabilistic algorithms comprising prominent methods such as local search algorithms for SAT and CSPs, like WalkSAT and the Min-Conflicts Heuristic, as well as more general metaheuristics like Genetic Algorithms, Simulated Annealing, Iterated Local Search, and Ant Colony Optimization. Our method is based on measuring and analysing run-time distributions (RTDs) for individual problem instances. We discuss this empirical methodology and its application to Las Vegas Algorithms for various problem domains. Our experience so far strongly suggests that using this approach for studying the behaviour of Las Vegas Algorithms can provide a basis for improving the understanding of these algorithms and thus facilitate further successes in their development and application.

1 Las Vegas Algorithms

Las Vegas Algorithms (LVAs) are nondeterministic algorithms for which, if a solution is found, its correctness is guaranteed. However, it is not guaranteed that for soluble instance of decision problems such an algorithm eventually finds a solution, or, for optimisation poblems, that an optimal or close-to-optimal solution will eventually be reached. Because of its nondeterministic nature, the run-time of a Las Vegas Algorithm is a random variable.

Las Vegas Algorithms are prominent not only in the field of Artificial Intelligence but also in other areas of computer science and Operations Research. Because of their inherent randomness, stochastic local search (SLS) algorithms are particular instances of LVAs. In recent years, SLS algorithms have become quite prominent for solving both NP-complete decision problems and NP-hard combinatorial optimisation problems. These

[1] Corresponding author; address: Computer Science Department, University of British Columbia, 2366 Main Mall, Vancouver, BC, Canada V6T 1Z4

algorithms, such as specific SLS algorithms for SAT and CSPs like WalkSAT [16] and the Min-Conflicts Heuristic [12], respectively, as well as more general metaheuristics like Tabu Search [2], Simulated Annealing [11], Genetic Algorithms [3], Evolution Strategies [14, 15], Ant Colony Optimisation [1], or Iterated Local Search [13, 18] have been found to be very successful on numerous problems from a broad range of domains. But also a number of systematic search methods, like some modern variants of the Davis Putnam algorithm for propositional satisfiability (SAT) problems, make use of non-deterministic decisions (like randomised tie-breaking rules) and can thus be characterised as Las Vegas Algorithms.

However, due to their stochastic nature, in analysing the behaviour of Las Vegas Algorithms one is mainly restricted to empirical methods, as theoretical results are difficult to obtain and often very limited in their practical applicability. The latter is, for example, the case for Simulated Annealing, which is proven to converge towards an optimal solution under certain conditions which, however, cannot be met in practice. On the other hand, theoretical results for algorithms which could be shown to be very effective in practice are usually very limited, as is the case for the most successful variants of Tabu Search. Often, the empirical methods that have been applied for the analysis of Las Vegas (and particularly SLS) algorithms in AI have been rather simplistic, like measuring and comparing average or median run-times over a test-set of instances sampled from a random problem distribution such as Random-3-SAT, Random Binary CSP, etc. It can be shown that these methods not only often give a very coarse description of algorithmic behaviour, but that they can also lead to misinterpretations and erroneous conclusions, e.g., when different sources of randomness (such as stochastic choices within the algorithm and probabilistic generation in the instance generation procedure, resp.) are not analytically separated [7].

2 Empirical Analysis using RTDs

We argue that a more sophisticated methodology for the empirical analysis of Las Vegas Algorithms is needed as a basis for their application, investigation, and further development; over the past three years, we developed such a refined empirical methodology and applied it extensively to Las Vegas Algorithms for various problem domains, foremost the well-known satisfiability problem in propositional logic (SAT). Our method is based on measuring and analysing run-time distributions (RTDs) for individual problem instances. For decision problems, this is done by running the given algorithm on the same problem instance for a number of times, where in each of these runs the time for finding a solution is recorded. From this data, an empirical run-time distribution can easily be estimated [7]. (For examples of empirical run-time distributions, see Figure 1.)

For optimisation problems additionally the solution quality has to be taken into ac-

count. In this case, we measure qualitative run-time distributions for different bounds on the required solution quality (which can be given, for example, as the percentage deviation from the best known solution or a lower bound on the optimal solution value). This can be effectively done by running the optimisation algorithm a number of times on the same problem instance; in each of these runs whenever a new best solution is found, the solution quality, the computation time needed to obtain it, and possibly some other statistic data for further analysis are recorded. This data is sufficient for estimating empirical run-time distributions for different solution quality bounds [18, 4]. (For examples of qualitative run-time distributions, see Figure 2.)

Based on our own empirical experience and more general considerations, we suggest the following guidelines for empirically analysing the run-time behaviour of Las Vegas algorithms.

1. Generally measure and compare RTDs, as opposed to basic descriptive statistics like mean, standard deviation, or percentiles. Enough runs of the algorithm should be performed to ensure that the estimates for the RTDs are sufficiently stable.

2. Try to approximate the empirical RTDs using parameterised functional models. Statistical goodness-of-fit tests like the χ^2-test should be used to evaluate these functional models. In our studies of SLS algorithms for various problem domains, we found that often approximations using exponential or generalised exponential distributions [4] were surprisingly accurate.

3. When comparing algorithms, check for cross-overs in their RTDs. If present, these typically indicate that by using portfolios or hybrid algorithms the problem can be solved more robustly and/or more efficiently.

4. When dealing with parameterised algorithms, it is often desirable and rewarding to study the impact of these parameters on the algorithm's RTD. When multiple parameters are used, care should be taken to avoid premature assumptions on the independence of their effects.

5. When using sets of problem instances (randomly generated or not), RTDs should be measured for individual instances to clearly detect differences in run-time behaviour across the test-set. Ideally, if the individual RTDs can be characterised using a parameterised model, the distribution of the model parameters across the test-set should be studied. If a parameterised model is not available, the distribution of basic descriptive statistics of the individual RTDs should be investigated. For sets of randomly generated instances this method ensures that the different sources of randomness (in the problem generation procedure and the LVA to be evaluated) are clearly separated.

6. Likewise, when comparing the performance of different algorithms on sets of instances, the comparison should be made on an individual instance basis. This allows to precisely analyse the correlation between the algortithms' performance across test-sets using standard statistical techniques.

Note that while we advocate to characterise the observed RTDs using parameterised functional models, many aspects of the RTD-based methodology do not rely on this so that they are still applicable when such models are not available. The advantage of parameterised approximations is twofold: first, they allow the compact and yet accurate characterisation of the observed run-time behaviour and secondly, by they often suggest generalisations and deeper explanations of the observed behaviour, in particular if the model parameters can be linked to algorithm parameters or properties of the given problem instances.

3 Applications of an RTD-based Methodology

Based on a classification of application scenarios for Las Vegas Algorithms, we have shown that in general, only RTDs provide all the information required to adequately describe the behaviour of these algorithms [4, 7]. We also demonstrated, how, based on functional approximations of RTDs for individual problem instances, interesting and novel characterisations for the run-time behaviour of some of the most popular stochastic local search algorithms in various areas of AI can be obtained, including SAT, Constraint Satisfaction, and various combinatorial optimisation problems [8, 10, 4, 18]. These characterisation results are of both, qualitative and quantitative nature and have a number of practical as well as theoretical implications.

The RTD-based methodology also provides a good basis for adequately comparing the performance of different Las Vegas Algorithms. Here, we additionally advocate the use of benchmark libraries comprising fundamentally different types of problems: instances from random problem distributions, such as Random-3-SAT; individual, application-relevant or otherwise interesting instances; and, where applicable, randomly generated, encoded instances from other domains (such as SAT-encoded graph colouring problems in random graphs). The latter have the advantage that they combine aspects of randomly generated and structured problem instances and are therefore, in our opinion, ideally suited for studying the impact of certain structural aspects of the given problem instances on algorithmic performance [6]. Based on these principles, we created and maintain SATLIB[2], a comprehensive public repository of SAT problem instances and algorithms; using our RTD-based methodology, we also conducted a large-scale empirical study comparing the

[2] `www.informatik.tu-darmstadt.de/AI/SATLIB`

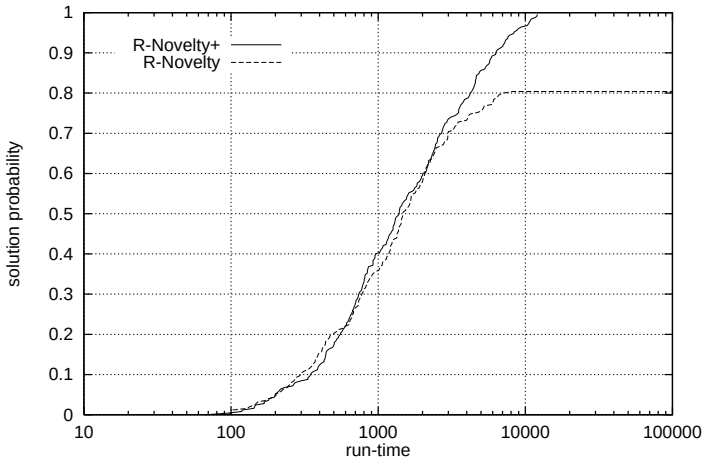

Figure 1: Run-time distributions for finding models of a hard, SAT-encoded graph colouring problem with 50 vertices using the R-Novelty algorithm and our improved version R-Novelty$^+$.

performance of various of the best known SLS algorithms for SAT on this benchmark suite [4, 9].

Finally, RTD-based analyses provide an important basis for tuning and improving the performance of SLS algorithms. Along these lines, e.g., it is easy to show under which conditions randomly restarting an SLS algorithm compromises its performance; also based on the methodology outlined above, we could recently develop significantly improved variants of some of the best known algorithms for SAT (Novelty$^+$ and R-Novelty$^+$, [4, 5]). The original algorithms suffered from premature stagnation of the search for a number of benchmark problem instances, which is clearly reflected in the respective RTDs (as exemplified in Fig. 1). Based on this observation, the algorithms could be slightly modified by allowing unconditional random walk steps with a small probability, which resulted in a significant improvement of their ability to solve the previously problematic problem instances (while for the other instances the performance remained essentially unaffected). This fact again can be easily and accurately established by an analysis of the empirical RTDs (see Fig. 1).

Similarly, we analysed the run-time behaviour of Iterated Local Search (ILS) algorithms which currently rank among the best performing approximation algorithms for large Travelling Salesman Problems. Our analysis revealed that ILS algorithms tend to quickly find good solutions for TSPs; yet, if very high solution qualities are required, ILS algorithms suffer from a type of stagnation which may strongly compromise their perfor-

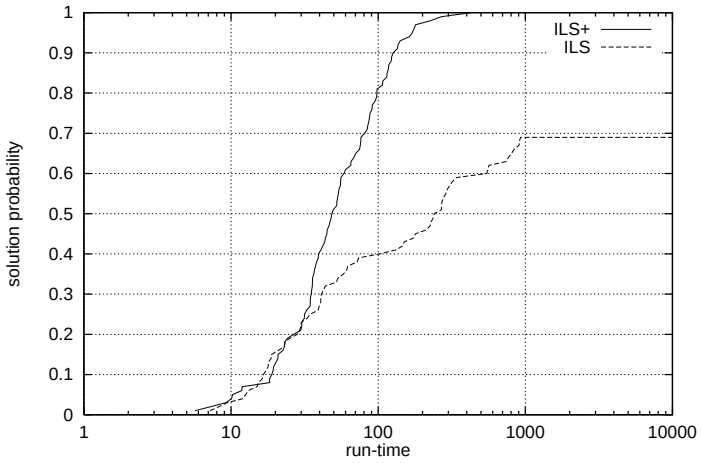

Figure 2: Run-time distributions for finding optimal solutions of TSP instance `lin318` using the basic ILS algorithm "ILS" and our improved version "ILS+".

mance. Based on this observation we could devise modifications to the ILS algorithms which, in practice, considerably increase their performance with respect to the detection of very high quality and even optimal solutions for TSP instances with several thousand cities (see Fig. 2; more details can be found in [18, 17]).

4 Conclusions

Although run-time distributions have been observed occasionally before, their use has been limited to concrete examples and specific application aspects. Our improved and refined empirical methodology is considerably more general and has already proven to be very useful for analysing the run-time behaviour of Las Vegas Algorithms in general, and SLS algorithms in particular. In the past, even the most successful applications of Las Vegas Algorithms in various areas of AI have been based on a fairly limited understanding of their behaviour. Our experience so far strongly suggests that using the RTD-based methodology for empirically studying the behaviour of Las Vegas Algorithms can provide a basis for improving the understanding of these algorithms and thus facilitate further successes in their development and application.

References

[1] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–42, 1996.

[2] F. Glover. Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[3] John H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[4] H.H. Hoos. *Stochastic Local Search - Methods, Models, Applications*. PhD thesis, TU Darmstadt, Germany, 1998.

[5] H.H. Hoos. On the Run-time Behaviour of Stochastic Local Search Algorithms for SAT. In *Proc. AAAI-99*, 1999 to appear.

[6] H.H. Hoos. SAT-Encodings, Search Space Structure, and Local Search Performance. In *Proc. IJCAI-99*, 1999 to appear.

[7] H.H. Hoos and T. Stützle. Evaluating Las Vegas Algorithms — Pitfalls and Remedies. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 238–245. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[8] H.H. Hoos and T. Stützle. Some Surprising Regularities in the Behaviour of Stochastic Local Search (poster abstract). In *Proc. CP'98*, volume 1520 of *LNAI*, page 470. Springer Verlag, 1998.

[9] H.H. Hoos and T. Stützle. Local Search Algorithms for SAT: An Empirical Evaluation. *Submitted to: J. Automated Reasoning, special Issue "SAT 2000"*, 1999.

[10] H.H. Hoos and T. Stützle. Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. *Submitted to: Artificial Intelligence*, 1999.

[11] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[12] S. Minton, M.D. Johnston, A.B. Philips, and P. Laird. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 52:161–205, 1992.

[13] O. Martin and S.W. Otto. Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research*, 63:57–75, 1996.

[14] I. Rechenberg. *Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Information*. Fromman Verlag, Freiburg, 1973.

[15] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Chichester, 1981.

[16] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise Strategies for Improving Local Search. In *Proceedings of AAAI'94*, pages 337–343. MIT Press, 1994.

[17] T. Stützle and H.H. Hoos. Analyzing the Run-time Behaviour of Iterated Local Search for the TSP. *Submitted to: III Metaheuristics International Conference*, 1999.

[18] T. Stützle. *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications*. PhD thesis, Darmstadt University of Technology, Department of Computer Science, 1998.