Statistical Models of Multistart Randomized Heuristic Search Performance

Vincent A. Cicirello
Computer Science and Information Systems
The Richard Stockton College of New Jersey
Pomona, NJ 08240

Technical Report: May 2008

Abstract

The complexity of many combinatorial optimization problems often preclude the application of exact problem solving techniques as the problem is scaled to real-world size. For example, for a problem that is NP-Hard in the strong sense, any algorithm that guarantees that its solutions are optimal is going to be limited in the size of the instance that it can solve given computational limitations. Rather than requiring "optimal" solutions, an alternative is to favor sufficiently good solutions that can be found efficiently. One approach to this uses a randomized heuristic algorithm. This paper specifically considers an algorithm known as Value Biased Stochastic Sampling (VBSS). VBSS uses a heuristic function to bias the random generation of a proposed solution to the problem. This process is repeated some number of times and retains only the best solution found over several trials. The key to the effectiveness of VBSS is the choice of the heuristic function. The job of the heuristic function is to provide problem dependent evaluation of the choices that can be made. This evaluation is used to bias the random decisions during the problem solving process. At the time of the design of such an algorithm, it may not be obvious which of several heuristics are best. During the past few years, we have been developing techniques for allowing VBSS to self-select which heuristic to use. Our approach relies on modeling the distribution of the quality of solutions obtained by VBSS over its allocated set of restarts. This paper presents some of our analysis of potential models, including goodness of fit tests for several possible assumptions we can make about the distribution of the quality of solutions. Our analysis leads us to the selection of the Generalized Extreme Value Distribution for our models of randomized heuristic performance.

1 Introduction

The complexity of many combinatorial optimization problems often preclude the application of exact problem solving techniques as the problem is scaled to real-world size. For example, for a problem that is NP-Hard in the strong sense, any algorithm that guarantees that its solutions are optimal is going to be limited in the size of the instance that it can solve given currently available computational resources.

Rather than requiring "optimal" solutions, an alternative is to favor sufficiently good solutions that can be found efficiently. For example, so-called anytime algorithms are algorithms that can provide some solution (though not necessarily a good one) very quickly, but which also provide solutions of non-decreasing quality as more computational time is available (Zilberstein and Russell, 1996; Zilberstein, 1996; Boddy and Dean, 1994). Many metaheuristic search algorithms often have the properties of an anytime algorithm. For example, most common variations of algorithms like simulated annealing (Kirkpatrick et al., 1983), genetic algorithms (Goldberg, 1989), tabu search (Glover, 1989; Glover, 1990), and the like can be considered anytime algorithms. These algorithms are often also largely problem independent, though some design decisions need be made that consider problem specific characteristics. For example, many of the algorithms in this class often rely on one or more heuristics to guide the problem solving process.

We have been exploring one particular type of metaheuristic, namely an algorithm known as *Value-Biased Stochastic Sampling (VBSS)* (Cicirello and Smith, 2005a) which is a variant of an algorithm known as Heuristic-Biased Stochastic Sampling (HBSS) (Bresina, 1996). VBSS uses a heuristic function to bias the random generation of a proposed solution to the problem. This process is repeated some number of times and retains only the best solution found over several trials. VBSS can clearly be used as an anytime algorithm. During its first iteration, it randomly samples the solution space which provides some solution very quickly. As VBSS continues to randomly sample the solution

space, only the best such solution that is found is retained. The quality of the final solution found is non-decreasing as more computational time and resources are allocated to the algorithm.

The key to the effectiveness of VBSS is the choice of the heuristic function used to guide the randomized search. The job of the heuristic function is to provide problem dependent evaluation of the choices that can be made. This evaluation is used to bias the random decisions during the problem solving process. At the time of the design of such an algorithm, it may not be obvious which of several heuristics are best.

During the past few years, we have been developing techniques for allowing VBSS to self-select which heuristic (or heuristics) to use. The general idea is that we can use feedback from the problem-solving process to learn about the performance of the available heuristics, to generate models of that performance, and to use those models to further guide the choice of heuristic for the problem. This approach is related to something called an algorithm portfolio which utilizes a collection of algorithms executed in parallel to minimize the risk (in terms of high variability of expected run length for problems that are NP-Hard) associated with relying on a single algorithm (Gomes and Selman, 2001; Gomes and Selman, 1997; Huberman et al., 1997). Our approach is related, but focuses more on problems for which it is inexpensive computationally to find some solution, but for which there is high variability in the quality of the solutions found. Our approach is to provide an algorithm like VBSS with a portfolio of heuristics that it can use. Our framework relies on modeling the distribution of the quality of solutions obtained by VBSS over its allocated set of restarts with a given heuristic. Models of the performance of each of the heuristics from the available set are then used to guide the selection of which heuristic to use on subsequent restarts.

This paper presents some of our analysis of potential models of heuristic performance, including goodness of fit tests for several possible assumptions we can make about the distribution of the quality of solutions. Our analysis leads us to the selection of the Generalized Extreme Value Distribution for our models of randomized heuristic performance. Previous experiments have provided empirical validation for the performance of this approach to automated heuristic selection (Cicirello and Smith, 2004) for a scheduling problem known as weighted tardiness scheduling with sequence-dependent setups, a problem that is NP-Hard in the strong sense. We have also developed an algorithm for using those models for effectively automating the selection of a heuristic by the algorithm itself (Cicirello and Smith, 2005b).

In this paper, we provide further statistical analysis of our selected model of heuristic performance. The remainder of the paper is oraganized as follows. In Section 2 we provide an overview of the VBSS algorithm. In Section 3 we discuss the motivation for our extremal value theory motivated model. Then, in Section 4 we provide statistical validation of that model. We conclude in Section 5 with a discussion of future work.

2 Value-Biased Stochastic Sampling

VBSS is an example of a type of algorithm known as stochastic sampling search. In a stochastic sampling search, the algorithm iteratively probes from the root of the search-tree down to a terminal node. Decisions are made at random and no memory of the search paths taken on previous probes is maintained. The simplest algorithm from this class is that of *iterative sampling* (Langley, 1992) which makes its random decisions uniformly from among all available choices. Other more sophisticated algorithms use domain heuristics to bias the randomized search. *Heuristic biased stochastic sampling* (HBSS) biases the random decisions by a function of a ranking determined by an ordering of the choices according to the heuristic (Bresina, 1996). Another related approach, referred to as *heuristic equivalency* (Gomes et al., 1997; Gomes et al., 1998; Gomes et al., 2000) by some and *acceptance bands* (Oddi and Smith, 1997; Cesta et al., 1999; Cesta et al., 2002) by others, considers all choices whose heuristic valuation is within some threshold of the heuristic's preferred choice to be equivalent. A random choice is then made uniformly from that set.

The particular stochastic sampling search algorithm that we employ in our work is that of *value biased stochastic sampling* (VBSS) (Cicirello and Smith, 2005a). Algorithm 1 shows pseudocode of VBSS. VBSS is closely related to HBSS. The key difference, however, is that it biases its random decisions according to a function of the heuristic values, rather than by a ranking over the choices. Like most other stochastic sampling search algorithms, it is a multistart algorithm where the best solution found over several runs is retained.

Although simplistic in nature, stochastic sampling search algorithms, especially those that use a heuristic for guidance, have been successful in a number of problem domains: telescope scheduling (Bresina, 1996), flow-shop scheduling (Watson et al., 1999), weighted tardiness scheduling (Cicirello and Smith, 2002; Cicirello and Smith, 2005a), oversubscribed scheduling (Kramer and Smith, 2004). They have also been incorporated as integral components of other types of algorithms, such as the ISES algorithm for resource constrained project scheduling (Cesta et al., 2002), as well as CSP search algorithms (Gomes et al., 2000), among others. Additionally, stochastic sampling

Algorithm 1: Value Biased Stochastic Sampling (VBSS)
Input: Number of iterations I; a "heuristic" function; a "bias" function; an "objective" function; and a search-tree T.
Output: A solution S.

```
VBSS(I, heuristic, bias, objective, T)
(1)     bestsofar ← solution S obtained if "heuristic" is followed from T
(2)     repeat I times
(3)         S ← root search-node of T
(4)         while S is a decision node of T
(5)             foreach choice C from S
(6)                 score[C] ← heuristic(C, S)
(7)             totalweight ← 0
(8)             foreach choice C from S
(9)                 weight[C] ← bias(score[C])
(10)                totalweight ← totalweight + weight[C]
(11)            foreach choice C from S
(12)                prob[C] ← weight[C] / totalweight
(13)            select randomly the choice C biased according to prob[C]
(14)            S ← Successor(S, C)
(15)        S ← OPTIONALPOSTPROCESSINGSTEP(S)
(16)        if objective(S) is superior to objective(bestsofar)
(17)            bestsofar ← S
(18)    return bestsofar
```

algorithms are sometimes used to generate starting configurations for local search algorithms. The pseudocode of Algorithm 1 allows for this through the OPTIONALPOSTPROCESSINGSTEP() of line 15, where one could potentially insert a call to some local search algorithm (e.g., simulated annealing or a hill climber).

3 A Model of Heuristic Performance Motivated by Extreme Value Theory

Algorithms such as VBSS require the selection of an appropriate domain search heuristic. The difficulty here is that for most problems of interest, there is no single clearly dominating choice. Our approach avoids restricting ourselves to any single search heuristic. Instead, we develop a framework for integrating several heuristics within VBSS (or similarly HBSS) (Cicirello, 2003; Cicirello and Smith, 2004; Cicirello and Smith, 2005b). The first required element of our framework is to model the distribution of the quality of solutions obtained over multiple runs of our stochastic sampler. During the search we can then estimate these models for each search heuristic in our set and use the models to guide the selection of a search heuristic for successive runs.

One might be tempted to assume a normal distribution is sufficient to model the quality of solutions obtained by biasing a stochastic sampler with a particular heuristic—e.g., by keeping track only of the mean quality obtained across the runs, and perhaps the standard deviation. There are examples that show such simplistic modeling assumptions can be quite effective. For example, Sadeh computes the expected improvement in the quality of solutions if one was to continue a run of simulated annealing below some threshold temperature; and then use such models to decide when to abandon a given run in favor of either restarting a fresh run or continuing a previously abandoned run (Sadeh et al., 1997). Similarly, Ruml computes the expected cost of solutions below a given node in a search tree and uses these models to bias a stochastic sampling algorithm—in a sense learning a search heuristic during the problem solving process (Ruml, 2001). Both of these algorithms produced very promising results.

We argue, however, that for our multistart stochastic sampling algorithm, abandoning normality assumptions can be advantageous. In fact, the quality of solutions across multiple runs of an algorithm like VBSS are not normally distributed as we will see shortly. We can better exploit problem solving feedback in the form of quality distributions if we consider distribution families that provide a better fit to this type of data. Specifically, we note that in a memoryless algorithm like VBSS, we maintain only the current best found solution. If an iteration produces a solution better, then we throw away the old solution, otherwise we throw away this new solution. Our goal in successive runs is to improve

upon the **best solution** we have yet found. We are thus most interested in the behavior at the extremum.

While exploring potential distribution families, we examined several problem instances of a weighted tardiness scheduling problem. We began by sampling the solution space uniformly at random and computing what is termed a *quality density function* (QDF) (Bresina et al., 1995). The QDF is just the probability density function of the quality of solutions if one was to sample the solution space uniformly. It can be quite useful in characterizing the difficulty of a problem instance; for example Bresina used it to define metrics for comparing the performance of search algorithms (Bresina et al., 1995). What we found was that for easy problem instances of our weighted tardiness scheduling problem that the optimal solutions were on average over 6.4 standard deviations better than the average feasible solution to the problem. We also found that the average solution obtained by VBSS using a strong domain heuristic on single iteration runs was also on average 6.4 standard deviations better than the average solution in the problem space (Cicirello and Smith, 2004). What this tells us is that both the optimal solutions, as well as the solutions typically found by our search algorithm, are actually rather rare events in the underlying problem space. Further examination of hard problem instances indicates that this is even more strongly the case. For hard problem instances, the optimal solutions are on average over 9.4 standard deviations better than the average solution in the problem space; while the average solution found by single iteration runs of VBSS is over 9.1 standard deviations better than the average random solution. VBSS guided by a strong domain search heuristic is essentially generating solutions that are generally quite good (though non-optimal) and which might be considered rare (or at least uncommon) when one considers the entire space of solutions for the problem instance.

For this reason, to model the quality of solutions generated by VBSS, we turn to extreme value theory, the study of rare or uncommon events (Coles, 2001). And more specifically, in our previous work (Cicirello, 2003; Cicirello and Smith, 2005b; Cicirello and Smith, 2004) we have utilized the distribution known as the Generalized Extreme Value (GEV) Distribution, a generalization of the extreme value distributions, types I (Gumbel), II (Fréchet), and III (Weibull), which are commonly reformulated as:

$$G(z) = \exp(-(1 + \xi(\frac{z-b}{a}))^{-1/\xi}) \tag{1}$$

where $\{z : 1 + \xi(\frac{z-b}{a}) > 0\}$, $-\infty < b < \infty$, $a > 0$, and $-\infty < \xi < \infty$. The case where $\xi = 0$ is treated as the limit of $G(z)$ as ξ approaches 0 to arrive at the Gumbel distribution.

4 Validating the Model of Heuristic Performance

In this Section, we consider our proposed GEV model of heuristic performance in more detail and compare it to other potential models. First, however, we need a method of estimating the GEV parameters. We use Hosking's maximum-likelihood estimator of the GEV parameters (Hosking, 1985). Also note that we use the term *Algorithm Quality Density Function (AQDF)* (Cicirello, 2003) to refer to an analog of Bresina's QDF but where we are specifically dealing with the probability density function of the quality of solutions generated by a specific stochastic search algorithm, such as VBSS, rather than simply the pdf of the underlying solution space.

Assuming that the distribution function behaves according to a GEV distribution, the probability P_i of finding a solution better than some threshold (τ) using heuristic i can be defined as:

$$P_i = 1 - G_i(-\tau) = 1 - \exp(-(1 + \xi_i(\frac{-\tau - b_i}{a_i}))^{-1/\xi_i}) \tag{2}$$

where the b_i, a_i, and ξ_i are estimated from the negative of the sample values (quality of solutions obtained by sampling with the heuristic i).

Our framework for integrating multiple heuristics into VBSS selects a heuristic to use for each restart randomly, but biases the random decision to favor heuristics with higher values of P_i, where τ is an algorithm parameter set based on the quality of solutions found so far. For complete details of how our framework uses the P_i to select which heuristic i to use for each restart of VBSS, see our prior work (Cicirello, 2003; Cicirello and Smith, 2004; Cicirello and Smith, 2005b). In the current paper, we focus solely on the appropriateness of our choice of the GEV as the model of the quality of solutions obtained with VBSS using a given heuristic.

Figure 1(a) provides an example of a GEV estimate for the AQDF for VBSS using a heuristic known as COVERT (Morton and Pentico, 1993) for an instance of a weighted tardiness scheduling problem. The GEV parameters were obtained using Hosking's maximum likelihood estimator. Note how well the long right-hand tail graphically fits

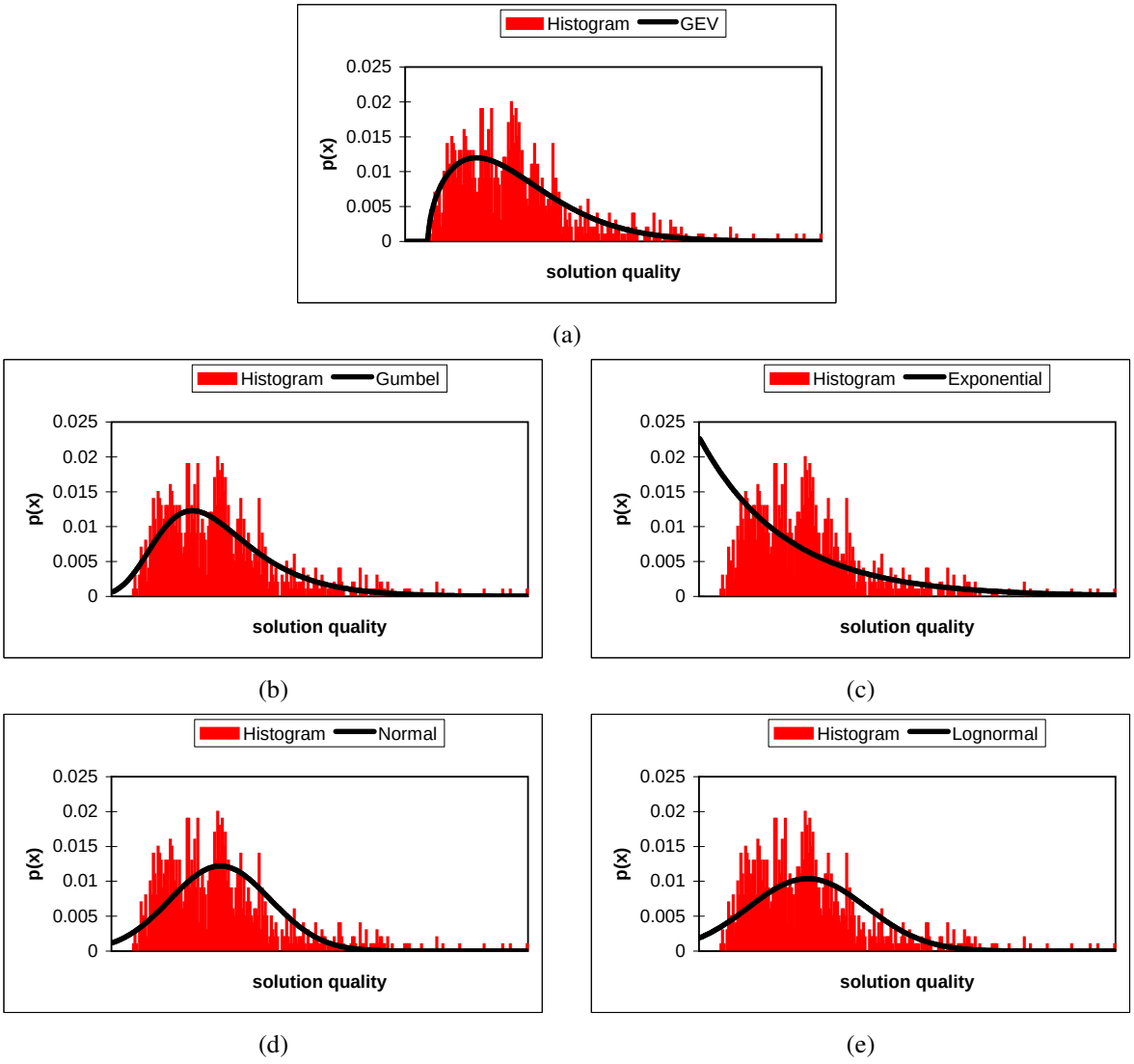

Figure 1: Examples of fits of several distribution families for the AQDF of VBSS using a strong domain heuristic for an instance of a weighted tardiness scheduling problem superimposed on a histogram estimate of this AQDF: (a) the GEV; (b) a Gumbel; (c) an Exponential; (d) a Normal; and (e) a Lognormal.

Table 1: Summary of the results of Chi-Squared Goodness of Fit tests.

Significance Level: $\alpha = 0.001$			
Distribution	χ^2	dof	upper critical value
GEV	17.32	18	42.31
Gumbel	45.82	19	43.82
Normal	84.26	19	43.82
Lognormal	84.26	19	43.82
Exponential	90.36	19	43.82

the solution quality data.[1] Also note how unlike the exponential, normal, and lognormal fits of Figure 1(c-e), the GEV estimate does not over predict the density of high quality solutions on the left-hand side of the distribution.[2] Figure 1(b) shows a fit of the Gumbel distribution (the Type I extreme value distribution). The Gumbel's long right-hand tail appears to fit well, but this model does over predict on the left (although not as severely as the normal, lognormal, and exponential fits).

From Figure 1(a) it appears that the GEV is a reasonable assumption for solution quality distributions. But, just how good a fit is it? We computed Chi-Squared Goodness of Fit tests for the distributions depicted in Figure 1(a-e). In the selection of the cell sizes for the Chi-Squared tests, we followed Kallenberg's recommendations for testing heavy-tailed distributions (Kallenberg et al., 1985)—we used 22 nonequiprobable cells (additional lower probability cells in the tails). This led to tests with 18 degrees of freedom (dof) for the GEV and 19 dof for the other distributions tested (the GEV has 3 parameters estimated from the data—location, scale, and shape—while the others tested have 2 parameters—location and scale). The tests were conducted at significance level $\alpha = 0.001$. The results are summarized in Table 1. The Gumbel, Normal, Lognormal, and Exponential distributions were all rejected by the Chi-Squared test at significance level 0.001. In fact, the normal, lognormal, and exponential all differ statistically (with extremely high significance) from the observed AQDF (P-values less than 0.000000001)—the Gumbel at level 0.0005. The value of the χ^2 statistic for the GEV ($\chi^2 = 17.32$ for a test with 18 dof) indicates a very good fit.

In Figure 2(a) we see the GEV estimates of the AQDFs using a strong domain heuristic (labeled as "Heuristic 1") and a weaker heuristic ("Heuristic 2") for a single instance of a weighted tardiness scheduling problem. This is the same problem instance used earlier to generate the AQDFs of Figure 1; and "heuristic 1" was that used to generate those AQDFs. Figure 2(b) shows their cumulative distribution functions (cdf). The GEV cdf for the solutions given by "heuristic 1"—the stronger heuristic—dominates that of "heuristic 2", as we would hope it would given that we already know that heuristic 1 is the stronger heuristic in this case. If we were to use these cdfs to compute an estimation of the probability of finding a solution of quality better than some threshold τ (i.e., using Equation 2) for each of these two heuristics, "heuristic 1" would be selected no matter what the value of τ. In this example, the GEV does an excellent job of predicting heuristic performance. Figure 2(c) and Figure 2(d) show the pdfs and cdfs for these same two heuristics for this same problem instance if we were to instead assume normality of the solution qualities. Under a normality assumption, for some values of τ we can be mislead into believing that heuristic 2 is the better heuristic in this situation.

5 Summary and Conclusions

We have been investigating approaches to integrating multiple search heuristics within a stochastic sampling search algorithm known as VBSS. In our previous research, we considered an approach where the distribution of the quality of solutions found by VBSS using each of the heuristics from some given set could be estimated, and then where those estimated quality distributions could in turn be used to guide the automated selection of heuristics to use on subsequent restarts of the VBSS algorithm. Our previous work argued for the use of the GEV distribution for the required heuristic performance model (Cicirello, 2003; Cicirello and Smith, 2004) and also considered how to use the model to guide the selection of heuristic on each restart (Cicirello, 2003; Cicirello and Smith, 2005b). This latter work specifically focused on how to balance the trade-off between choosing a heuristic to obtain additional data to refine the model of its performance versus choosing the heuristic predicted to be the best choice by the model.

[1] The histogram estimate of the AQDF was obtained via 1000 iterations of VBSS.

[2] This AQDF is for a minimization problem, so better qualities are to the left.

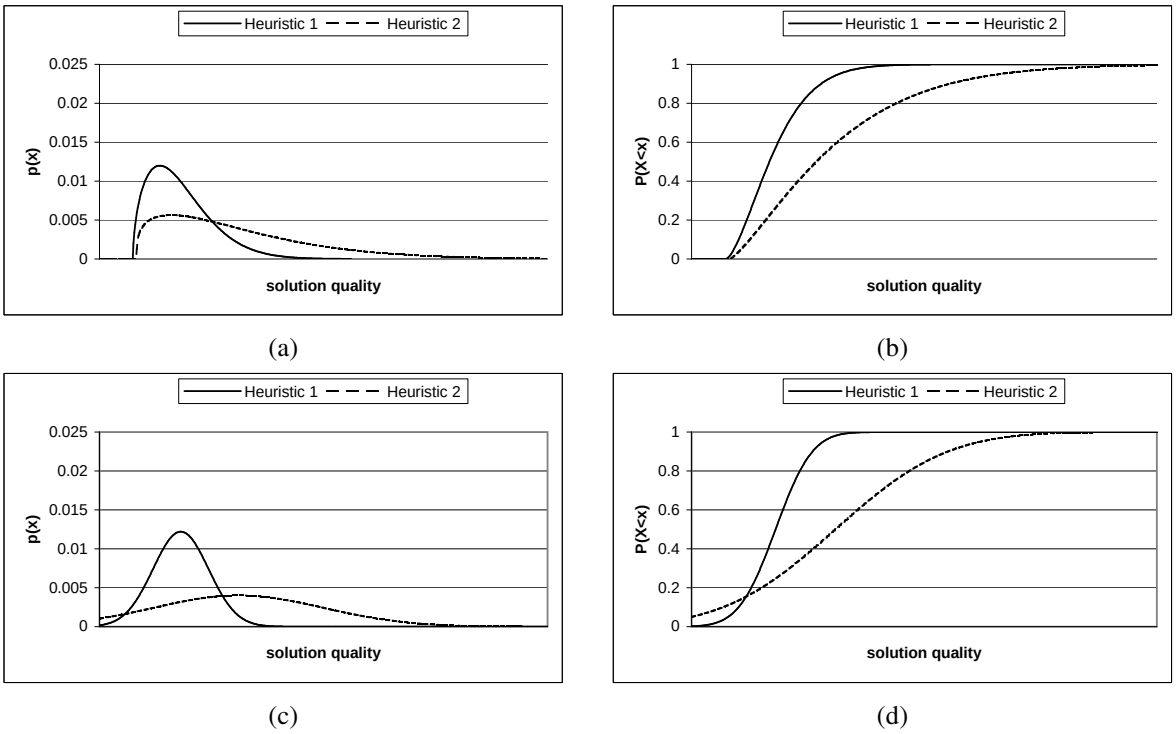

Figure 2: GEV estimates of the probability density functions (a) and cumulative distribution functions (b) for VBSS using two different heuristics for a single instance of a weighted tardiness scheduling problem. Parts (c) and (d) show the same assuming normally distributed AQDFs. Heuristic 1 is known to be quite strong; while heuristic 2 is somewhat weaker for this problem instance.

In this paper, we provided additional statistical validation of our choice of the GEV model for the performance of VBSS. Of the distribution families that we have considered, the GEV appears best as seen through Goodness of Fit tests. We have also provided an example where a poor choice of model (e.g., assuming the AQDF is normal) could potentially lead our framework for automatically selecting an appropriate heuristic from a portfolio of heuristics astray; while the GEV model better fits the solution quality data, especially the left-hand tail on which our approach heavily relies.

In the future, we have plans to further expand upon our work by investigating the applicability of our approach to multiheuristic search to other types of search algorithm. For example, the author has interests in genetic algorithms and other evolutionary computation approaches. These types of algorithms rely on operators inspired by models of natural genetics and evolutionary processes. One of the key design criteria of such an algorithm is deciding which specific operators to use for your problem. An alternative approach that we plan to investigate is whether we can instill the ability to self-select operators into a genetic algorithm.

References

Boddy, M. and Dean, T. (1994). Decision-theoretic deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–286.

Bresina, J., Drummond, M., and Swanson, K. (1995). Expected solution quality. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1583–1590. Morgan Kaufmann.

Bresina, J. L. (1996). Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Volume One*, pages 271–278. AAAI Press.

Cesta, A., Oddi, A., and Smith, S. F. (1999). An iterative sampling procedure for resource constrained project scheduling with time windows. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1022–1029. Morgan Kaufmann.

Cesta, A., Oddi, A., and Smith, S. F. (2002). A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8:109–136.

Cicirello, V. A. (2003). *Boosting Stochastic Problem Solvers Through Online Self-Analysis of Performance*. PhD thesis, The Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Also available as technical report CMU-RI-TR-03-27.

Cicirello, V. A. and Smith, S. F. (2002). Amplification of search performance through randomization of heuristics. In Van Hentenryck, P., editor, *Principles and Practice of Constraint Programming – CP 2002: 8th International Conference, Proceedings*, volume LNCS 2470 of *Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag. Ithaca, NY.

Cicirello, V. A. and Smith, S. F. (2004). Heuristic selection for stochastic search optimization: Modeling solution quality by extreme value theory. In Wallace, M., editor, *Principles and Practice of Constraint Programming – CP 2004: 10th International Conference, Proceedings*, volume LNCS 3258 of *Lecture Notes in Computer Science*, pages 197–211. Springer-Verlag. Toronto, Canada.

Cicirello, V. A. and Smith, S. F. (2005a). Enhancing stochastic search performance by value-biased randomization of heuristics. *Journal of Heuristics*, 11(1):5–34.

Cicirello, V. A. and Smith, S. F. (2005b). The max k-armed bandit: A new model of exploration applied to search heuristic selection. In Veloso, M. M. and Kambhampati, S., editors, *The Proceedings of the Twentieth National Conference on Artificial Intelligence*, volume 3, pages 1355–1361. AAAI Press. Pittsburgh, PA. Winner of the AAAI'05 Outstanding Paper Award.

Coles, S. (2001). *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag.

Glover, F. (1989). Tabu search – part I. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu search – part II. *ORSA Journal on Computing*, 2(1):4–32.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.

Gomes, C., Selman, B., and Kautz, H. (1998). Boosting combinatorial search through randomization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, pages 431–437. AAAI Press.

Gomes, C. P. and Selman, B. (1997). Algorithm portfolio design: Theory vs. practice. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*. Morgan Kaufmann.

Gomes, C. P. and Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, 126:43–62.

Gomes, C. P., Selman, B., and Crato, N. (1997). Heavy-tailed distributions in combinatorial search. In *Principles and Practices of Constraint Programming (CP-97)*, Lecture Notes in Computer Science, pages 121–135. Springer-Verlag.

Gomes, C. P., Selman, B., Crato, N., and Kautz, H. (2000). Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24:67–100.

Hosking, J. R. M. (1985). Algorithm AS 215: Maximum-likelihood estimation of the paramaters of the generalized extreme-value distribution. *Applied Statistics*, 34(3):301–310.

Huberman, B. A., Lukose, R. M., and Hogg, T. (1997). An economics approach to hard computational problems. *Science*, 275:51–54.

Kallenberg, W. C. M., Oosterhoff, J., and Schriever, B. F. (1985). The number of classes in chi-squared goodness-of-fit tests. *Journal of the American Statistical Association*, 80(392):959–968.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Kramer, L. and Smith, S. F. (2004). Task swapping for schedule improvement: A broader analysis. In *Proceedings 14th International Conference on Automated Planning and Scheduling*.

Langley, P. (1992). Systematic and nonsystematic search strategies. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, pages 145–152.

Morton, T. E. and Pentico, D. W. (1993). *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*. John Wiley and Sons.

Oddi, A. and Smith, S. F. (1997). Stochastic procedures for generating feasible schedules. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, pages 308–314. AAAI Press.

Ruml, W. (2001). Incomplete tree search using adaptive probing. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 235–241.

Sadeh, N. M., Nakakuki, Y., and Thangiah, S. R. (1997). Learning to recognize (un)promising simulated annealing runs: Efficient search procedures for job shop scheduling and vehicle routing. *Annals of Operations Research*, 75:189–208.

Watson, J. P., Barbulescu, L., Howe, A. E., and Whitley, L. D. (1999). Algorithm performance and problem structure for flow-shop scheduling. In *Proceedings, Sixteenth National Conference on Artificial Intelligence (AAAI-99), Eleventh Innovative Applications of Artificial Intelligence Conference (IAAI-99)*, pages 688–695. AAAI Press.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83.

Zilberstein, S. and Russell, S. (1996). Optimal composition of real-time systems. *Artificial Intelligence*, 82:181–213.