Multi-heuristic Stochastic Sampling Search: Extreme Value Theory and the Max K-Armed Bandit

Vincent A. Cicirello

Computer Science
Stockton University
Galloway, NJ 08205 USA
https://www.cicirello.org/

Technical Report AI-09-003

March 2009

Multi-heuristic Stochastic Sampling Search: Extreme Value Theory and the Max K-Armed Bandit

Vincent A. Cicirello

Computer Science
Stockton University
Galloway, NJ 08205 USA
https://www.cicirello.org/

Technical Report AI-09-003

Abstract

In this paper, we develop theoretical foundations for integrating multiple heuristics within a stochastic sampling search procedure. Such a capability is important in problem domains where several heuristics are available but none dominate across all problem instances. In order to utilize multiple heuristics, a search algorithm needs a mechanism for selecting from among the alternatives. We can view this selection problem as an online learning problem where the search framework builds models of the distributions of the quality of solutions given by the heuristics and uses those models as guidance for selecting heuristics throughout the search process. One key observation is that the distribution of solution values obtained across multiple runs of a stochastic sampling search algorithm using a strong domain heuristic is heavy-tailed. Modeling these solution quality distributions requires turning to extremal value theory. In particular, we motivate the use of a distribution known as the Generalized Extreme Value (GEV) distribution for our model of the performance of a search heuristic. We show specifically that it can be quite advantageous to abandon the more typical normality assumptions in favor of this GEV model. Secondly, we develop an exploration policy for allocating trials of a stochastic sampling algorithm amongst the set of component heuristics. This policy follows from an analysis of a new variation of the well known multiarmed bandit problem that we call the Max K-Armed Bandit. The analysis indicates that under our GEV model, a double exponentially increasing rate of trials of the stochastic sampling algorithm should be allocated to the observed best heuristic relative to the number of trials allocated to the alternative heuristics. We validate our search framework—which we call *Quality Distribution Based sEArch CONtrol (QD-BEACON)*—empirically for two NP-Hard scheduling problems: (1) weighted tardiness scheduling and (2) resource constrained project scheduling with time windows (RCPSP/max). Our approach leads to an algorithm that is competitive with the current best heuristic algorithms for each of these problems, including finding new best solutions to some difficult instances of RCPSP/max.

Keywords: bandit problems, modeling heuristic performance, multiarmed bandit, stochastic sampling, multi-heuristic search, scheduling

ACM Classes: F.2.2; G.2.1; G.3; I.2.6; I.2.8

MSC Classes: 68T20; 68W05; 68W20; 68W50; 60G70

https://orcid.org/0000-0003-1072-8559

1 Introduction

Employing heuristics for guidance is at the very heart of much of AI search. For some types of search algorithm, there is theoretical insight one can turn to when faced with the problem of selecting an appropriate search heuristic for a given problem domain. For example, when implementing A* issues such as admissibility and informedness can be brought into play to decide among alternative search heuristics. For other types of search algorithm, however, there is little theory to draw on and the choice of what heuristic to use remains somewhat of a black-art. For example, what really makes for a good choice of a local search operator in a particular application context? Typically, this decision is made in the algorithm design phase after gathering some empirical data on the problem; or in some cases a heuristic may be selected arbitrarily, justifying the choice with one's past experience in algorithm implementation.

The particular type of search algorithm that we focus on in this paper is that of heuristically biased stochastic sampling search. Heuristically biased stochastic sampling search algorithms choose alternative branches of the search-space randomly, but use a domain-dependent heuristic to evaluate the choices to provide a bias for those random decisions. This class of search algorithm includes variants such as *heuristic biased stochastic sampling (HBSS)* [8] and *value biased stochastic sampling (VBSS)* [15]. This type of heuristic search algorithm has been successful in a variety of scheduling domains [8, 40, 10, 15].

The performance of a stochastic sampling algorithm depends heavily on the quality of the search heuristic that is used. The difficulty in selecting an appropriate domain-dependent heuristic is that there is rarely, if ever, a single dominating heuristic for a given problem domain. Consider the problem domain of scheduling, for example. In this domain a large collection of "dispatch scheduling heuristics" have been developed for sequencing jobs that must compete for a shared resource [30]. Specific heuristics tend to work well in certain situations and are less than satisfactory in others. Figure 1 illustrates one well-known tradeoff with respect to the weighted tardiness scheduling problem, where features of the problem instance at hand can be used to guide the choice of heuristic. Specifically, in cases where due dates are very loose, the relatively simple heuristic, earliest due date first (EDD), is typically very effective. The top half of Figure 1 illustrates this very example by showing the distribution of the quality of solutions produced across multiple runs of a stochastic sampling algorithm using the EDD heuristic as well as using a heuristic known as WSPT on a "loose duedate" instance of the problem. The EDD heuristic clearly dominates for this instance of the problem (lower values of quality are better). Contrast this with the bottom half of this figure which shows the quality distributions for an instance with tighter duedates. There, the WSPT heuristic performs better—in very heavily loaded situations, the WSPT heuristic is provably optimal [30]. For many problems of interest, however, these special cases are atypical. A poor choice of search heuristic a priori can lead to ineffective problem solving later on.

In this paper, we develop theoretical foundations for integrating multiple heuristics within a stochastic sampling search procedure. Such a capability is important in problem domains where several heuristics could be exploited but none dominate across all problem instances. In order to utilize multiple heuristics, a search algorithm needs a mechanism for selecting from among the alternatives. We view this search heuristic selection problem as an online learning problem where the search framework builds models of the distributions of the quality of solutions given by the heuristics and uses those models as guidance for selecting heuristics on each subsequent run during the problem solving process. First, we consider the characteristics of distributions of the quality of solutions obtained across multiple runs of such a stochastic sampling search using a strong domain heuristic. Specifically, we observe that solution quality distributions are often heavy-tailed, which leads us to models from the field of extreme value theory. Namely, we motivate the use of a distribution known as the *Generalized Extreme Value (GEV) Distribution* for our model of the

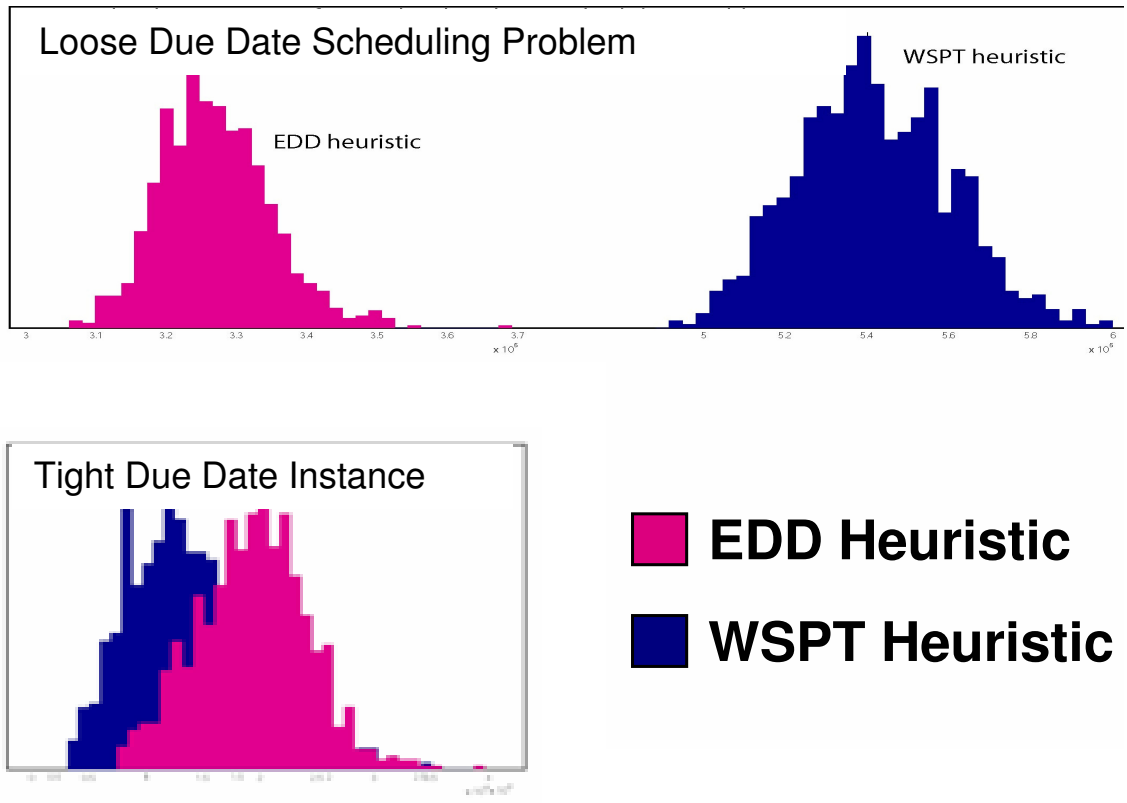

Figure 1: Comparison of the distribution of the quality of solutions given across multiple runs of a stochastic sampling algorithm using two different heuristics for two different problem instances of the weighted tardiness scheduling problem.

performance of a search heuristic within a stochastic sampling algorithm. Our search framework must allocate trials of the stochastic sampling algorithm amongst the set of component heuristics. To do this, our framework requires an exploration policy for balancing the tradeoff between executing trials for the purpose of improving our performance models with exploiting those models for search guidance. Our exploration policy is based upon an analysis of a new variation of the well known multiarmed bandit problem that we call the *Max K-Armed Bandit* [14]. We call our framework *Quality Distribution Based sEArch CONtrol (QD-BEACON)*. Just as beacons of light guide sailors through murky waters in the darkness of night, QD-BEACON can guide search algorithms through the murky selection of a heuristic.

The paper proceeds as follows. In Section 2, we begin with some background on stochastic sampling search in general; and in particular describe the VBSS algorithm which we use in our experiments. In Section 3 we then present our extreme value theory motivated approach to modeling the performance of search heuristics. Our exploration policy for allocating trials of amongst the set of component heuristics as well as the underlying theory motivating this policy is detailed in Section 4. In Section 5 we present the results of experiments on two NP-Hard scheduling problems: (1) weighted tardiness scheduling and (2) resource constrained project scheduling with time windows (RCPSP/max). We conclude the paper in Section 6.

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
```

2 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* [29] which makes its random decisions uniformly from among all available choices. Other more sophisticated algorithms use domain heuristics to bias the randomized search. Bresina's *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 [8]. Another related approach, referred to as *heuristic equivalency* [20, 19, 21] by some and *acceptance bands* [32, 9, 10] 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. Note here that Gomes' heuristic equivalency was not used for iterative sampling, but rather to randomize decisions made within a systematic backtracking search.

The particular stochastic sampling search algorithm that we employ later in this paper as part of our empirical study is that of *value biased stochastic sampling* (VBSS) [15]. 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 [8], flow-shop

scheduling [40], weighted tardiness scheduling [12, 15], oversubscribed scheduling [28]. They have also been incorporated as integral components of other types of algorithms, such as the ISES algorithm for resource constrained project scheduling [10], as well as CSP search algorithms (e.g., [21]), among others. Additionally, stochastic sampling 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). Our experiments with the weighted tardiness scheduling problem later in Section 5.1 applies such a local search to the solutions generated by the runs of VBSS.

3 Modeling a Solution Quality Distribution

Algorithms such as HBSS and 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).

Perhaps the simplest possible approach to integrating multiple search heuristics within a stochastic sampling search might be to take a round-robin approach where the set of heuristics is simply cycled—i.e., using each heuristic in rotation, allocating the same number of runs to each. One of the drawbacks of this round-robin approach is that it blindly treats all available heuristics the same throughout the problem solving process. Before the search begins, we may have little if any information regarding the differential performance of the set of heuristics for the particular problem instance. Thus, at the start we really cannot do much more than treat the heuristics as equivalent. The round-robin approach is typically employed in algorithm portfolio approaches where several highly variable algorithms are either executed in parallel or interleaved on a single processor [25]. But if we assume that there may be some heuristics that are intrinsically better-suited to the given problem instance (even if we do not actually know which they are beforehand) then this round-robin approach can waste a good deal of runs on the less well suited heuristics from the set.

To address the problem, the QD-BEACON framework for multi-heuristic stochastic sampling search combines multiple search heuristics by using feedback from the search to learn performance models of the heuristics for the given problem instance. 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 et al compute 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 [34]. 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 [33]—in a sense learning a search heuristic during the problem solving process. 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 in Section 3.3. 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 tail behavior of our solution quality distribution. To be able to say anything particularly meaningful about the tail, we need to make assumptions about the distribution family.

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) by Bresina [7]. The QDF is just the probability density function of the quality of solutions if one were 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 [7]. 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 [13]. 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 sampling from the tail of the solution-space. For this reason, we turn to extreme value theory, the study of rare or uncommon events [16].

3.1 Generalized Extreme Value Distribution

Upon noting that we are essentially dealing with behavior at the extremes of the solution space, we consider the extreme value theory analog to the central limit theory. Let $M_n = \max\{X_1, \ldots, X_n\}$, where $X_1, \ldots, X_n$ is a sequence of independent random variables having a common distribution function F. For example, perhaps the X_i are the mean temperatures for each of the 365 days in the year, then M_n would correspond to the annual maximum temperature. To model M_n, extreme value theorists turn to the *extremal types theorem* [16]:

Theorem 1 *If there exists sequences of constants $\{a_n > 0\}$ and $\{b_n\}$ such that $P((M_n - b_n)/a_n \leq z) \rightarrow G(z)$ as $n \rightarrow \infty$, where G is a non-degenerate distribution function, then G belongs to one of the following families:*

I: $G(z) = \exp(-\exp(-(\frac{z-b}{a}))),\ -\infty < z < \infty$

II: $G(z) = \exp(-(\frac{z-b}{a})^{-\alpha})$ *if* $z > b$ *and otherwise* $G(z) = 0$

III: $G(z) = \exp((\frac{z-b}{a})^{\alpha})$ *if* $z < b$ *and otherwise* $G(z) = 1$

for parameters $a > 0$, b and in the latter two cases $\alpha > 0$.

These are the extreme value distributions, types I (Gumbel), II (Fréchet), and III (Weibull). The types II and III distributions are heavy-tailed. The Gumbel distribution is medium-tailed. These distributions are

commonly reformulated into the generalization known as the generalized extreme value distribution (GEV):

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

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. Under the assumption of Theorem 1, $P((M_n - b_n)/a_n \leq z) \approx G(z)$ for large enough n which is equivalent to $P(M_n \leq z) \approx G((z - b_n)/a_n) = G^*(z)$ where $G^*(z)$ is some other member of the generalized extreme value distribution family.

The main point here is that to model the distribution of the maximum element of a fixed-length sequence (or block) of identically distributed random variables (i.e., the distribution of "block maxima"), one needs simply to turn to the GEV distribution regardless of the underlying distribution of the individual elements of the sequence.

3.2 Modeling Solution Quality via the GEV

Theorem 1 only explicitly applies to modeling the distribution of "block maxima". The assumption we now make is that the quality distribution for a heuristically-biased stochastic sampling algorithm behaves the same as the distribution of "block maxima". The rationale for this assumption is that since the solutions produced by an algorithm such as VBSS even on single iteration runs are so rare (i.e., between 6 and 10 standard deviations better than the mean of the solution space), executing a single iteration of VBSS is roughly equivalent to sampling the "block maxima" for uniformly sampled solutions from the underlying problem space for a significantly large block size. That is, the quality of the solution given by a single run of VBSS is equivalent to the best solution given across a significantly large number of uniform samples taken from the overall solution space.

Thus, we use a GEV distribution to model the algorithm quality density function (AQDF) [11] of a given heuristic. The AQDF represents the probability density function of the quality of solutions obtained by a particular stochastic search algorithm across multiple runs. You can obtain the AQDF by first modeling its cumulative distribution function as a GEV distribution. Algorithm 2 shows how our QD-BEACON framework exploits this idea to integrate multiple search heuristics into VBSS.

To use the GEV as our model, we first note that the particular combinatorial optimization problems we are interested in are minimization problems. So we actually need the "block minima" analog to Equation 1. Let $M'_n = \min\{X_1, \ldots, X_n\}$. We want $P(M'_n < z)$. Let $M''_n = \max\{-X_1, \ldots, -X_n\}$. Therefore, $M'_n = -M''_n$ and $P(M'_n < z) = P(-M''_n < z) = P(M''_n > -z) = 1 - P(M''_n \leq -z)$. Therefore, 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). The call to UPDATEAQDF() in line 21 of Algorithm 2 updates our current estimate of the AQDF—its pseudocode is shown in Algorithm 3.

To compute these parameters, we use Hosking's maximum-likelihood estimator of the GEV parameters [24]. In estimating the GEV parameters, Hosking's algorithm is called at least once. Hosking's algorithm is an iterative numerical algorithm which requires an initial "guess" of the GEV parameters. The first

Algorithm 2: QD-BEACON for Integrating Multiple Heuristics within VBSS

Input: Number of iterations I; a set of heuristics H; an "objective" function; the threshold parameter τ, and a search-tree T.

Output: A solution S.

QD-BEACON/VBSS(I, H, objective, τ, T)

```
(1)   bestsofar ← solution S obtained if heuristic h1 ∈ H followed from T
(2)   foreach heuristic h ∈ H − h1
(3)     S ← solution obtained if heuristic h is followed from T
(4)     if objective(S) is superior to objective(bestsofar)
(5)       bestsofar ← S
(6)   repeat I times
(7)     heuristic ← SELECTHEURISTIC()
(8)     S ← root search-node of T
(9)     while S is a decision node of T
(10)      foreach choice C from S
(11)        score[C] ← heuristic(C, S)
(12)      totalweight ← 0
(13)      foreach choice C from S
(14)        weight[C] ← bias(score[C])
(15)        totalweight ← totalweight + weight[C]
(16)      foreach choice C from S
(17)        prob[C] ← weight[C] / totalweight
(18)      select randomly the choice C biased according to prob[C]
(19)      S ← Successor(S, C)
(20)    S ← OPTIONALPOSTPROCESSINGSTEP(S)
(21)    UPDATEAQDF(heuristic, objective(S), τ)
(22)    if objective(S) is superior to objective(bestsofar)
(23)      bestsofar ← S
(24)  return bestsofar
```

call (Algorithm 4 line 4) uses an initial "guess" of the parameters as recommended by Hosking (set assuming a Gumbel distribution—lines 1-3 of Algorithm 4). If Hosking's algorithm fails to converge, then a fixed number of additional calls are made with random initial guesses of the parameters (Algorithm 4 lines 8-14). If convergence still fails, we use the values of the parameters as estimated by assuming a type I extreme value distribution (the Gumbel distribution)—Algorithm 4 lines 15-18. It should be noted that this fallback condition appears to rarely occur, if ever, in our experiments. Much of the time, the first call to Hosking's algorithm converges upon a maximum-likelihood estimate of the GEV parameters.

3.3 Analysis of Fits of the GEV

Figure 2(a) provides an example of a GEV estimate for the AQDF for VBSS using a heuristic known as COVERT [30] 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 the solution quality data.[1] Also note how unlike the exponential, normal, and lognormal fits of Fig-

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

Algorithm 3: Using the GEV Distribution within QD-BEACON

Input: A heuristic, the threshold parameter τ, and the quality of the solution just obtained

Output:

Description: Updates the N_i, X_i, Y_i, P_i, $S_{i,j}$, and the GEV parameters appropriately. Note that it assumes smaller objective values, or qualities, are better.

UPDATEAQDF(Heuristic,Quality,τ)

(1) $i \leftarrow$ **index of** Heuristic **in** H

(2) $N_i \leftarrow N_i + 1$

(3) $X_i \leftarrow X_i +$ Quality

(4) $Y_i \leftarrow Y_i +$ Quality^2

(5) Insert Quality into $S_{i,j}$

(6) $\mu_i \leftarrow \frac{X_i}{N_i}$

(7) $\sigma_i \leftarrow \sqrt{\frac{Y_i - N_i \mu_i^2}{N_i - 1}}$

(8) $\{b_i, a_i, \xi_i\} \leftarrow$ MLEOFGEV(μ_i,σ_i,$-1 * S_{i,j}$)

(9) $P_i \leftarrow 1 - \exp(-(1 + \xi_i(\frac{-\tau - b_i}{a_i}))^{-1/\xi_i})$

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

ure 2(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 2(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 2(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 2(a-e). In the selection of the cell sizes for the Chi-Squared tests, we followed Kallenberg's recommendations for testing heavy-tailed distributions [27]—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.

Figure 3(a) shows 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 weighted tardiness scheduling—the

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

Algorithm 4: Maximum Likelihood Estimation of the GEV parameters
Input: The sample mean and standard deviation. The list of data-points S to use for the estimation
Output: The maximum likelihood estimates of b, a, and ξ

```
MLEofGEV(μ,σ,S)
(1)   ã ← σ√6 / π
(2)   b̃ ← μ − 0.5772ã
(3)   ξ̃ ← 0
(4)   {ã, b̃, ξ̃} ← HOSKING(S, ã, b̃, ξ̃, ResultCode)
(5)   if ResultCode is Success
(6)      return {ã, b̃, ξ̃}
(7)   else
(8)      for 1 to 5
(9)         ã ← random value
(10)        b̃ ← random value
(11)        ξ̃ ← random value
(12)        {ã, b̃, ξ̃} ← HOSKING(S, ã, b̃, ξ̃, ResultCode)
(13)        if ResultCode is Success
(14)           return {ã, b̃, ξ̃}
(15)     ã ← σ√6 / π
(16)     b̃ ← μ − 0.5772ã
(17)     ξ̃ ← 0
(18)     return {ã, b̃, ξ̃}
```

same instance used earlier to generate the AQDFs of Figure 2, and "heuristic 1" was that used to generate those AQDFs. Figure 3(b) shows the cumulative distribution functions (cdf). The GEV cdf for "heuristic 1"—the stronger heuristic—dominates that of "heuristic 2", as expected given that we already know that heuristic 1 is the stronger heuristic in this case. If we use Equation 2 to compute the probability of finding a solution of quality better than some threshold τ for each of these two heuristics, "heuristic 1" would be selected no matter the value of τ. In this example, the GEV does an excellent job of predicting heuristic performance. Figures 3(c-d) show the pdfs and cdfs for these same two heuristics for this same problem instance but assuming normality of the solution qualities. Under a normality assumption, some values of τ can mislead us into believing that heuristic 2 is the better heuristic.

4 The Exploration-Exploitation Tradeoff

In Section 3 we presented an approach to modeling the distribution of the quality of solutions produced by a stochastic sampling search algorithm using a particular search heuristic. If we would like to use such models to inform the selection of an appropriate search heuristic for each run of the stochastic sampler, then we need to balance the tradeoff between exploration (executing runs to gather data about the quality of solutions obtained with a given heuristic) and exploitation (using the current models to guide the search).

The *K-Armed Bandit* [6] often serves as an analogy for balancing exploration and exploitation in search domains. The problem is to allocate trials to the arms of a k-armed bandit (i.e., slot machine with k arms, each with a different pay-out distribution) with the goal of maximizing expected total reward. Many have analyzed variations of the bandit problem (e.g., [1, 2, 3, 4, 6]). Others have used bandits as inspiration for,

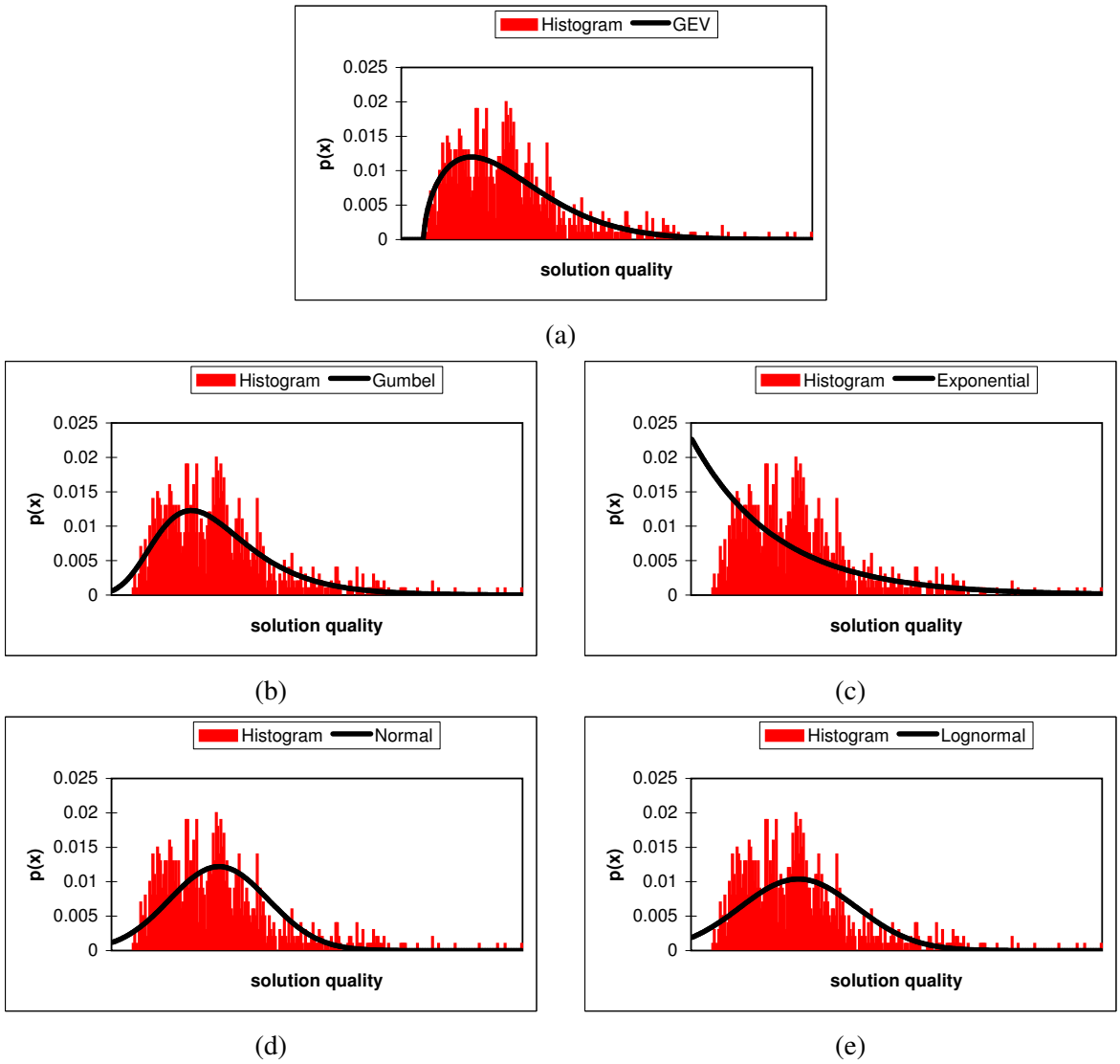

Figure 2: Examples of fits of several different distribution families for the AQDF of VBSS using a strong domain heuristic for an instance of a weighted tardiness scheduling problem superimposed on a histogram estimate of this AQDF: (a) the GEV; (b) a Gumbel; (c) an Exponential; (d) a Normal; and (e) a Lognormal.

or justification of, particular exploration strategies—e.g., for genetic algorithms [22, 23] and reinforcement learning [26, 39].

Given that the basic multi-armed bandit problem is concerned with maximizing the expected sum of rewards, it is not a direct match to our problem of stochastic search control. In the stochastic search case, we have our current reward (i.e., the best solution found so far) and need to find some reward that is better yet. In this Section, a variation of the multiarmed bandit is posed—the *Max K-Armed Bandit Problem*.[3] The problem, simply stated, is to allocate trials among the k arms so as to maximize the expected best single sample reward. As mentioned earlier, a stochastic sampling algorithm retains only the best solution found

[3]The Max K-Armed Bandit Problem, and much of the analysis that appears below, was first presented at AAAI'05 [14].

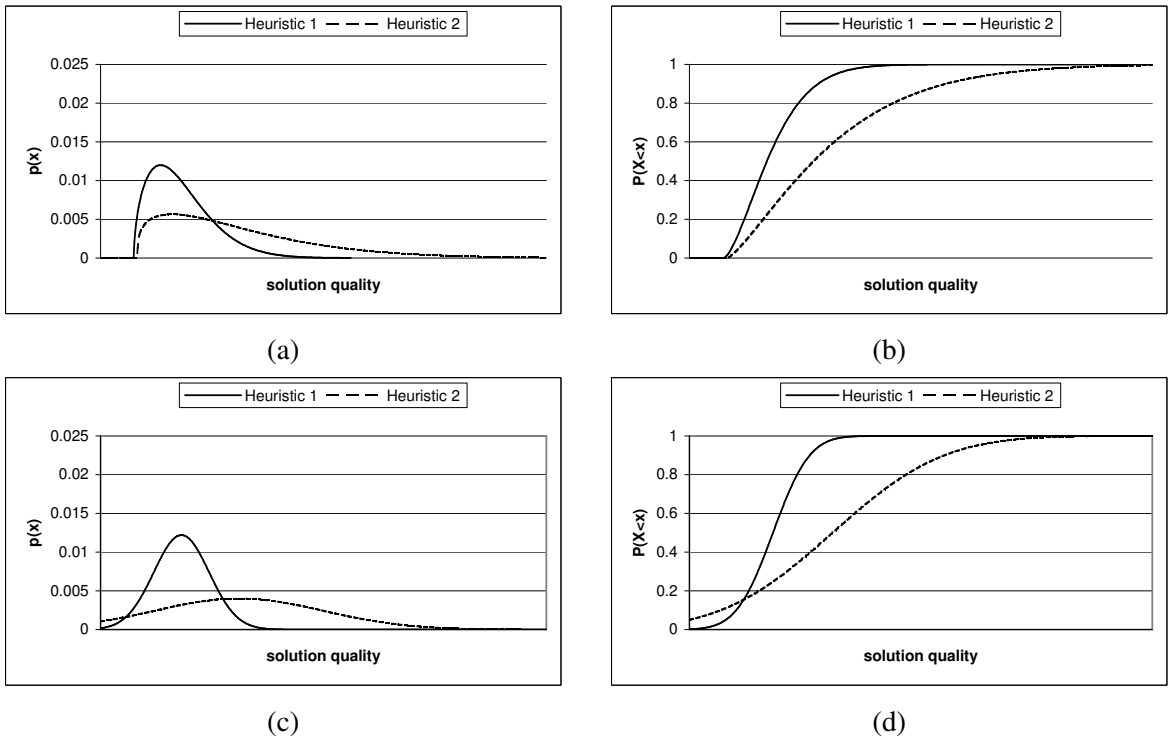

Figure 3: GEV estimates of the probability density functions (a) and cumulative distribution functions (b) for VBSS using two different heuristics for a single instance of a weighted tardiness scheduling problem. Parts (c) and (d) show the same assuming normally distributed AQDFs. Heuristic 1 is known to be quite strong; while heuristic 2 is somewhat weaker for this instance.

during its multiple runs. Thus, the goal of future runs is to find a solution better than the best we have already found—and not to optimize the expected sum of solution qualities across all runs.

In the subsections that follow, we begin by overviewing the result of Holland's analysis of the original multiarmed bandit problem [22, 23]. The structure of Holland's analysis serves as a model for our subsequent analysis of the Max K-Armed Bandit. We then formally define the Max K-Armed Bandit problem and present our initial analysis of it including our distributional assumptions. This analysis then leads directly to a recommended exploration policy for multi-heuristic stochastic sampling search.

4.1 The K-Armed Bandit: Holland's Analysis

The *k-armed bandit* problem plays a major role in Holland's original theoretical analysis of the Genetic Algorithm (GA). Holland uses the k-armed bandit analogy to show that the GA achieves a near-optimal tradeoff of exploration and exploitation [22, 23].

For the two-armed bandit, the expected reward for arm one is μ_1 with variance σ_1^2 (μ_2 and σ_2^2 for arm two). Furthermore, $\mu_1 \geq \mu_2$, but it is not known which arm is which. The problem is to maximize expected reward for a series of trials. One must determine the optimal tradeoff of exploratory actions (i.e., to discover the payoffs) and exploitation actions (i.e., playing the apparent best). The k-armed bandit is the obvious generalization. Let $R(\mu, \sigma)$ be a reward function that samples a normal distribution with mean μ and standard deviation σ, and let n_i be the number of samples given the i-th arm. The objective is to allocate

the n_i to optimize:

$$\max \sum_{i=1}^{k} n_i R(\mu_i, \sigma_i). \tag{3}$$

If the μ_i are known, then all N trials should be allocated to the arm with the largest μ_i to maximize the expected value of this objective. Without knowledge of the μ_i, it is necessary to perform some exploration to solve the problem.

For the two-armed bandit, Holland showed the optimal policy (to minimize expected loss from trials of the worst arm) allocates n^* trials to the worst arm, and $N - n^*$ to the best arm where in the limit:

$$N - n^* \sim \Theta(\exp(cn^*)), \tag{4}$$

where c is a constant [22, 23]. The trials allocated the observed best arm should increase exponentially relative to the allocation to the observed worst arm. Holland generalized this to the k-arm case, showing the worst-case expected loss for the problem occurs when $\mu_2 = \mu_3 = \ldots = \mu_k$ and $\sigma_2 = \sigma_3 = \ldots = \sigma_k$; and further showing that the best arm should be allocated $N - (k-1)m^*$ trials where N is the total number of trials and where each of the other $k - 1$ arms are allocated m^* trials. The optimal number of trials, in the limit, is:

$$N - (k-1)m^* \sim \Theta(\exp(cm^*)). \tag{5}$$

Thus, the number of trials allocated to the observed best arm in the optimal allocation should increase exponentially with the number of trials allocated to each of the other $k - 1$ arms.

4.2 The Max K-Armed Bandit

With the above analysis of the k-armed bandit in mind, let us now consider the *Max K-Armed Bandit*. In the Max K-Armed Bandit Problem, we are faced with a series of N trials. In any given trial, we can choose any of the k arms. For each of the arms there is an expected payoff according to some probability distribution. The goal is to maximize the value of the *best* single reward received over the N trials. This new objective is to allocate N trials among the arms to optimize:

$$\max \max_{i=1}^{k} \max_{j=1}^{n_i} R_j(D_i), \tag{6}$$

where $R_j(D_i)$ is the reward of the j-th trial of arm i with reward distribution D_i and where arm i is sampled n_i times.

Under assumptions motivated by extreme value theory, we show that to maximize the expected max single sample reward over N trials, the number of samples taken from the observed best arm should grow double exponentially in the number of samples taken from the observed second best. We proceed in three steps. First, we make some assumptions about the payoff distributions associated with each arm. Then we consider the special case of two arms. Finally, we generalize this solution to K arms.

4.2.1 Payoff Distribution Assumptions

To analyze the Max K-Armed Bandit, it is necessary to specify the type of distribution that each of the arms follow. In the classic version of the bandit problem, this is not necessary. Since the classic problem concerns the maximization of the expected sum of rewards, it is sufficient to make assumptions about the means and standard deviations of the arms. In the Max K-Armed Bandit case, however, we require an expression for

the expected max of a series of N trials as a function of N. This necessitates an assumption about the form of the underlying distribution of trials. The extremal types theorem tells us that the distribution of the max of a series of independent and identically distributed trials (as the length of the series grows large) belongs to one of three distribution families independent of the underlying distribution of the trials: the Gumbel, the Fréchet, or the Weibull [16]. However, which of those distribution families depends on the form of the underlying distribution of the samples.

To make appropriate assumptions we consider the target application—allocating runs among a set of multistart stochastic search heuristics for combinatorial optimization. Our Section 3 motivation indicates that we should assume that the trials of an arm follow a GEV distribution. Since an assumption of the most general GEV distribution prevents a closed form analysis, let us instead assume that each of the arms samples from a type I extreme value distribution (or the Gumbel distribution). This assumption is actually quite reasonable. We have found that MLE estimates of the GEV parameters for solution quality distributions are most often either a Gumbel ($\xi = 0$) or a Weibull.[4] When Weibull, we have found that the value of ξ is often quite small (e.g., between 0 and -1)—i.e., approaching the shape of a Gumbel. The Gumbel distribution has a cumulative probability of:

$$P(Z \leq z) = G(z) = \exp\left(-\exp\left(-\left(\frac{z-b}{a}\right)\right)\right), \tag{7}$$

where b is the location parameter and a the scale parameter.

4.2.2 The Max 2-Armed Bandit Case

Theorem 2 *The Two-Armed Double Exponential Sampling Theorem:*

To optimize the Max 2-Armed Bandit, where the arm samples are drawn from Gumbel distributions, the observed best arm should be sampled at a rate that increases double exponentially relative to the observed second best. That is, if the observed second best arm is sampled n times out of a total number of trials N. The optimal sampling policy allocates $N - n \sim \Theta(\exp(\exp(cn)))$ trials to the observed best arm.

Proof: Let there be two arms, M_1 and M_2, with the rewards of M_i drawn from a Gumbel distribution $G_i(x)$ with location parameter b_i and scale parameter a_i. The mean reward of a single sample of M_i is therefore: $\mu_i = b_i + 0.5772a_i$, where 0.5772 is Euler's number, and the standard deviation is: $\sigma_i = \frac{a_i\pi}{\sqrt{6}}$.

Given that the expected largest sample of a series of trials must be maximized, an expression is needed for the distribution of the maximum of a series of samples. Given N samples $\{X_1, \ldots, X_N\}$ from a distribution, the distribution of the maximum of these samples is:

$$P(\max(X_i) \leq x) = P(X \leq x)^N. \tag{8}$$

With the assumption of samples drawn from a Gumbel distribution, we have:

$$P(\max(X_i) \leq x) = \exp(-N\exp(-\frac{x-b}{a})). \tag{9}$$

This simplifies to:

$$P(\max(X_i) \leq x) = \exp(-\exp(-\frac{x-b-a\ln N}{a})). \tag{10}$$

[4] We have been unable to confirm the existence of any Type II (or Fréchet) AQDFs.

From this we see that the distribution of the max of N samples drawn from a Gumbel distribution with location parameter b and scale parameter a is also a Gumbel distribution with location parameter, $b_{\max} = b + a \ln N$ and scale parameter $a_{\max} = a$. Thus, the expected max reward of N samples from each of the two arms in the problem is:

$$b_i + 0.5772a_i + a_i \ln N. \tag{11}$$

Consider that M_1 is the better of the two arms in the problem. This necessitates a definition for "better". Let:

$$b_1 + 0.5772a_1 + a_1 \ln N > b_2 + 0.5772a_2 + a_2 \ln N \tag{12}$$

which implies that $a_1 \geq a_2$. Otherwise, for great enough N this inequality would fail to hold.

In the two-armed problem, where we do not know with certainty which arm is M_1 and which is M_2, the expected max reward if we had access to an omniscient oracle is clearly $b_1 + 0.5772a_1 + a_1 \ln N$ —the expected max reward of giving all N trials to the better arm. However, given that we cannot know with certainty which arm is which, some exploration is necessary. Consider that we draw n samples from the observed second best arm, and $N - n$ samples from the observed best arm. Now consider the loss of reward associated with sampling from the second best arm. There are two cases to consider:

1. The observed best is really the best. In this case, the loss comes from giving n less samples to the best arm—with an expected loss equal to: $a_1(\ln N - \ln(N - n))$.

2. The observed best arm is really second best. The loss in this case depends on whether the expected value of giving $N - n$ samples to the second best arm is greater than giving n samples to the best arm. That is, the expected loss is: $\min\{a_1(\ln N - \ln n), (b_1 - b_2) + 0.5772(a_1 - a_2) + a_1 \ln N - a_2 \ln(N - n)\}$. This form of loss is maximized when the expected value of the max of n samples of the best arm equals that of $N-n$ samples of the second best. This allows us to consider a simplification of the expected loss in this case: $a_1(\ln N - \ln n)$.[5]

Let q be the probability that the observed best arm is really second best. Therefore, $(1 - q)$ is the probability that the observed best really is the best. The expected loss of sampling n times from the observed second best and $N - n$ times from the observed best arm, as a function of n is therefore:

$$l(N) = q(a_1(\ln N - \ln n)) + (1 - q)(a_1(\ln N - \ln(N - n))). \tag{13}$$

This can be simplified to:

$$l(N) = q(a_1(\ln(N - n) - \ln n)) + a_1(\ln N - \ln(N - n)). \tag{14}$$

To select a value for n that minimizes the expected loss, we need to define q as a function of n. Let M_b be the arm that is perceived as best (i.e., the arm perceived to have the highest expected max single sample reward over a series of N trials) and M_w be the arm that is perceived as second best. The probability q can be stated as the probability that the expected max value of N samples of M_w is greater than the expected max value of N samples of M_b. If we note that the parameters of a Gumbel distribution can be estimated (e.g., [31]) from the data by $\tilde{a} = \frac{s\sqrt{6}}{\pi}$ and $\tilde{b} = \bar{X} - 0.5772\tilde{a}$, where $\bar{X}$ and s are the sample mean and sample

[5] Alternatively, we could also consider the simplification $(b_1 - b_2) + 0.5772(a_1 - a_2) + a_1 \ln N - a_2 \ln(N - n)$, but this would unnecessarily complicate the analysis.

standard deviation, then we can define:

$$q(n) = P\left((\tilde{b}_b + 0.5772\tilde{a}_b + \tilde{a}_b \ln N) - (\tilde{b}_w + 0.5772\tilde{a}_w + \tilde{a}_w \ln N) < 0\right) \quad (15)$$

$$= P\left((\bar{X}_b + \frac{s_b\sqrt{6}}{\pi} \ln N) - (\bar{X}_w + \frac{s_w\sqrt{6}}{\pi} \ln N) < 0\right) \quad (16)$$

$$= P\left(\bar{X}_b - \bar{X}_w < (s_w - s_b)\frac{\sqrt{6}}{\pi} \ln N\right). \quad (17)$$

The central limit theorem says that $\bar{X}_b$ approaches a normal distribution with mean μ_b and variance $\frac{\sigma_b^2}{N-n}$. Similarly, $\bar{X}_w$ approaches a normal distribution with mean μ_w and variance $\frac{\sigma_w^2}{n}$. The distribution of $\bar{X}_b - \bar{X}_w$ is the convolution of the distributions $\bar{X}_b$ and $-\bar{X}_w$. The convolution of these distributions is by definition a normal distribution with mean $\mu_b - \mu_w$ and variance $\frac{\sigma_b^2}{N-n} + \frac{\sigma_w^2}{n}$. Using an approximation for the tail of a normal distribution, we can define $q(n)$ as:

$$q(n) \lesssim \frac{1}{\sqrt{2\pi}} \frac{\exp(-x^2/2)}{x} \quad (18)$$

where

$$x = \frac{(\mu_b - \mu_w) + \frac{\sqrt{6}}{\pi} \ln(N)(\frac{\sigma_b}{\sqrt{N-n}} - \frac{\sigma_w}{\sqrt{n}})}{\sqrt{\frac{\sigma_b^2}{N-n} + \frac{\sigma_w^2}{n}}}. \quad (19)$$

Given the expressions for $q(n)$ and x, note that $q(n)$ decreases exponentially in n. Using the same simplification made by Holland [23], note that no matter the value for σ_b, there is a large enough N such that for n close to its optimal value, $\frac{\sigma_b^2}{N-n} \ll \frac{\sigma_w^2}{n}$. This leads to:

$$x \lesssim \frac{(\mu_b - \mu_w)\sqrt{n} - \frac{\sigma_w\sqrt{6}}{\pi} \ln N}{\sigma_w} \quad (20)$$

To select the value of n that will minimize the loss $l(n)$ we begin by taking the derivative of $l(n)$ with respect to n:

$$\frac{dl}{dn} = \frac{dq}{dn}(a_1(\ln(N-n) - \ln n)) - q(n)(\frac{a_1}{N-n} + \frac{a_1}{n}) + \frac{a_1}{N-n}, \quad (21)$$

where

$$\frac{dq}{dn} \lesssim -q(n)\frac{x^2+1}{x}\frac{dx}{dn}, \quad (22)$$

and

$$\frac{dx}{dn} \lesssim \frac{\mu_b - \mu_w}{2\sigma_w\sqrt{n}}. \quad (23)$$

The optimal value of n occurs when $\frac{dl}{dn} = 0$ so we can get a bound on the optimal n by solving the following inequality:

$$0 \lesssim \frac{a_1}{N-n} - q(n)\frac{a_1 N}{N-n} - q(n)\frac{x^2+1}{x}\frac{dx}{dn}(a_1(\ln(N-n) - \ln n)). \quad (24)$$

We can collect the logarithmic terms on the left to obtain

$$\ln(N-n) - \ln n \lesssim \frac{(1-q(n)N)x}{(N-n)q(n)\frac{dx}{dn}(x^2+1)}. \tag{25}$$

Recalling that $q(n)$ decreases exponentially in n, $(1-q(n)N)$ rapidly approaches 1. Noting $\frac{x}{x^2+1} \lesssim \frac{1}{x}$, obtain:

$$\ln(N-n) - \ln n \lesssim \frac{1}{(N-n)q(n)\frac{dx}{dn}x}. \tag{26}$$

Substituting expressions for $q(n)$ and $\frac{dx}{dn}$ we get:

$$\ln(N-n) - \ln n \lesssim \frac{\sigma_w\sqrt{8\pi}\sqrt{n}}{(N-n)(\mu_b-\mu_w)} \exp\left(\frac{(\mu_b-\mu_w-\frac{\sigma_w\sqrt{6}}{\pi\sqrt{n}}\ln(N))^2\, n}{2\sigma_w^2}\right) \tag{27}$$

Finally, exponentiate both sides of the inequality, to obtain:

$$N-n \lesssim \exp\left(\ln(n) + \frac{\sigma_w\sqrt{8\pi}\sqrt{n}}{(N-n)(\mu_b-\mu_w)} \exp\left(\frac{(\mu_b-\mu_w-\frac{\sigma_w\sqrt{6}}{\pi\sqrt{n}}\ln(N))^2\, n}{2\sigma_w^2}\right)\right). \tag{28}$$

The question that remains is which term in the exponential dominates the expression. We can take the fraction involving $N-n$ up into the exponential and gain insight into the answer to this question:

$$N-n \lesssim \exp\left(\ln(n) + \exp\left(\frac{(\mu_b-\mu_w-\frac{\sigma_w\sqrt{6}}{\pi\sqrt{n}}\ln(N))^2\, n}{2\sigma_w^2} + \ln\left(\frac{\sigma_w\sqrt{8\pi}\sqrt{n}}{(N-n)(\mu_b-\mu_w)}\right)\right)\right). \tag{29}$$

We must now determine which part of this double exponential dominates as the total number of samples N grows large. Consider the following limits:

$$\lim_{N\to\infty} \frac{\left(\mu_b-\mu_w-\frac{\sigma_w\sqrt{6}}{\pi\sqrt{n}}\ln(N)\right)^2 n}{2\sigma_w^2} = \infty \tag{30}$$

$$\lim_{N\to\infty} \ln\left(\frac{\sigma_w\sqrt{8\pi}\sqrt{n}}{(N-n)(\mu_b-\mu_w)}\right) = -\infty \tag{31}$$

Note that the first expression is dominated by the $(\ln N)^2$ and that within the logarithm of the second expression, the N in the denominator dominates. For large enough N, it is sufficient to consider which of $(\ln N)^2$ and $\ln(1/N)$ dominates. Consider the following:

$$\begin{aligned}
\lim_{N\to\infty}\{(\ln N)^2 + \ln(1/N)\} &= \lim_{N\to\infty}\{(\ln N)^2 + \ln 1 - \ln N\} && (32)\\
&= \lim_{N\to\infty}\{\ln N\,(\ln N - 1)\} && (33)\\
&= \infty && (34)
\end{aligned}$$

Taking this into account and making a few other obvious simplifications, we can arrive at:

$$N-n \sim \Theta(\exp(\exp(cn))) \tag{35}$$

This shows that the number of trials $N-n$ given to the observed best arm should grow double exponentially in n to maximize the expected max single sample reward.

4.2.3 Generalization to the K-Armed Case

Theorem 3 *The Multiarmed Double Exponential Sampling Theorem: To optimize the Max K-Armed Bandit (samples drawn from Gumbel distributions), the observed best arm should be sampled at a rate increasing double exponentially relative to the number of samples given the other $k-1$ arms.*

Proof: To make this inductive leap from the result of the two-arm case to the k-arm case, observe the following. The worst case loss in the k-armed case occurs when the $k-1$ worst arms are identical (as is the case in Holland's analysis of the original k-armed bandit). If these $k-1$ arms are identical then it doesn't matter how we allocate trials among them—the result is equally poor. But, if any of these $k-1$ arms is better than any of the other $k-2$ arms, then we can improve our expected reward by allocating more trials to it. Assume the worst case that the $k-1$ arms are identical. With m^* trials given to each of these $k-1$ worst arms, the analysis of the k-armed case can be considered a special case of the analysis of the two-armed problem. Specifically, we have the observed best arm and a meta-arm comprised of the aggregation of the other $k-1$ arms. The meta-arm is given $n^* = m^*(k-1)$ trials uniformly distributed across the $k-1$ arms. Since the $k-1$ arms are identical in the worst case, the meta-arm behaves identically to the second best arm in the two-arm case. Thus, the number of samples $N - m^*(k-1)$ given the observed best arm should grow double exponentially in $n^* = m^*(k-1)$.

4.3 Exploration Strategy

Recall that our goal is to find a good exploration strategy for allocating trials to different heuristics. To design an exploration policy that follows the double exponential sampling theorems, consider Boltzmann exploration [26, 39]. Let the temperature parameter T decay exponentially, choosing heuristic h_i with probability:

$$P(h_i) = \frac{\exp((R_i)/T)}{\sum_{j=1}^{H} \exp((R_j)/T)}. \tag{36}$$

The R_i is some indicator/estimator of the expected max of a series of trials of heuristic h_i. For example, R_i can be an estimator for the expected max for some fixed length series of trials given some distribution assumption. Specifically, we consider $R_i = P_i$, where P_i is defined by Equation 2 of Section 3 (i.e., the estimates of the probability of finding a solution of quality better than some threshold). To derive the double exponentially increasing allocation of trials to the observed best arm, the temperature parameter must follow an exponentially decreasing cooling schedule (e.g., $T_j = \exp(-j)$ where j is the trial number). That is, on iteration j choose heuristic h_i with probability:

$$P(h_i|j) = \frac{\exp((R_i)/\exp(-j))}{\sum_{k=1}^{H} \exp((R_k)/\exp(-j))}. \tag{37}$$

We adopt this as the recommended exploration policy (line 7 of Algorithm 2).

5 Experimental Results

We now present results of an empirical investigation of our framework for multi-heuristic stochastic sampling. The experiments presented here use VBSS as the underlying stochastic sampling search algorithm. Problem solving feedback is used to maintain GEV estimates (as described in Section 3) of the AQDFs for each of the possible search heuristics, and the exploration policy of Section 4 is employed to balance

the allocation of trials to the set of heuristics. In the next two subsections we discuss results for two NP-Hard scheduling problems: weighted tardiness scheduling (Section 5.1) and resource-constrained project scheduling (Section 5.2).

5.1 Weighted Tardiness Scheduling

The Weighted Tardiness Scheduling Problem is a sequencing problem. A set of jobs $J = \{j_1, \ldots, j_N\}$ must be sequenced on a single machine. Each of the N jobs j has a weight w_j, duedate d_j, and process time p_j. Preempting a job during processing is not permitted. Only one job at a time can be processed. The objective is to sequence the set of jobs J on the machine to minimize the total weighted tardiness: $T = \sum_{j \in J} w_j T_j = \sum_{j \in J} w_j \max(c_j - d_j, 0)$, where T_j is the tardiness of job j; and c_j, d_j is the completion time and duedate of job j. The completion time of job j is equal to the sum over the process times of all jobs that come before it in the sequence plus that of the job j itself. Specifically, let $\pi(j)$ be the position in the sequence of job j. We can now define c_j as: $c_j = \sum_{i \in J, \pi(i) <= \pi(j)} p_i$.

We use VBSS [15] here to generate biased initial configurations for a local search procedure for the weighted tardiness problem known as Multistart Dynasearch [17]—i.e., dynasearch is used as the OPTIONALPOSTPROCESSINGSTEP() of line 20 of Algorithm 2. Multistart Dynasearch is a state-of-the-art solver for this scheduling problem. The original Multistart Dynasearch used unbiased random initial solutions. The claim of Congram et al was that heuristically biasing the initial configurations would do no better than beginning at random solutions, although they did not present any experimental evidence to support this claim [17]. As we will see in our experimental results, QD-BEACON can be used to more effectively seed the starting point for a run of dynasearch.

For the set of search heuristics, we turn to the literature on dispatch scheduling policies [30]. A few of the best known dispatch policies for this scheduling problem are used here as candidate search heuristics:

- weighted shortest process time, $\text{WSPT}_i = \frac{w_i}{p_i}$;
- earliest duedate, $\text{EDD}_i = \frac{1}{d_i}$;
- $\text{COVERT}_i(t) = \frac{w_i}{p_i}(1 - \frac{\max(0, d_i - p_i - t)}{k p_i})$, with current time t and parameter k; and
- $\text{R\&M}_i(t) = \frac{w_i}{p_i} \exp(-1 * \frac{\max(0, d_i - p_i - t)}{k \bar{p}})$, with average process time $\bar{p}$.

5.1.1 Experimental Setup

In this experiment, these heuristics are combined across multiple runs of the dynasearch algorithm. On any given run, VBSS is used, along with one of these heuristics to construct an initial solution, which is then locally optimized using dynasearch. At the start of each run, a search heuristic is selected according to one of the following exploration policies: double exponentially increasing rate of allocations to the observed best heuristic (D-EXP); faster than double exponentially increasing allocation rate (FASTER); and an exponentially increasing allocation rate (EXP). We also compare to multistart dynasearch (M-DYNA) as originally specified by Congram et al, as well as a round-robin selection scheme (RR) where heuristics are selected in round robin order to seed successive runs of dynasearch. Additionally, we compare to an approach that models heuristic performance under a normality assumption (NORM). D-EXP, FASTER, and EXP all use the GEV model.

Table 2: Weighted tardiness: For each number of runs $[N]$, bold indicates the most best known solutions found. ARPD (and MRPD) are the average (and maximum) relative percentage deviation from the best known solutions.

Algorithm	NB	ARPD	MRPD
D-EXP[400]	94.3	0.12	10.07
NORM[400]	85.4	0.18	11.35
EXP[400]	85.0	0.14	11.28
FASTER[400]	78.7	0.19	13.51
M-DYNA[400]	62.0	1.66	76.18
RR[400]	59.8	2.47	92.59
D-EXP[800]	100.7	0.12	10.07
NORM[800]	91.9	0.13	10.67
EXP[800]	89.3	0.14	11.28
FASTER[800]	83.6	0.17	13.51
M-DYNA[800]	68.3	1.29	74.28
RR[800]	65.4	2.08	82.26
D-EXP[1600]	107.3	0.11	8.47
NORM[1600]	97.1	0.12	9.13
EXP[1600]	95.0	0.12	9.75
FASTER[1600]	87.5	0.16	11.28
M-DYNA[1600]	73.3	1.16	71.06
RR[1600]	70.9	1.45	77.62

5.1.2 Results

The results presented here are for the 100 job instances from the benchmark problem set in the OR-Library [5]. The set contains 125 instances. Results are shown in Table 2. NB is the number of best known solutions found (no further improvement is made). ARPD (and MRPD) are the average (and maximum) relative percentage deviation from the best known solutions. The results shown are averages of 10 executions for each of the 125 problem instances (average of 1250 total executions).

The trend for any number of iterations considered is that the double exponentially increasing rate of allocations finds the most best known solutions, with smallest percentage deviation from the best knowns. The next best in terms of these criteria is when a normal assumption is made, followed closely by when the observed best heuristic is given an exponentially increasing allocation of trials, followed by the variation with a faster than double exponentially increasing rate of allocations. All variations of QD-BEACON outperform the round robin approach to multi-heuristic search, supporting our earlier hypothesized shortcoming of a round robin approach. M-DYNA is the next to the worst of the six variations considered. There is clearly benefit to biasing the initial configurations of the M-DYNA local search, contrary to the untested hypothesis of Congram et al [17]. What is peculiar is that M-DYNA, using unbiased random initial configurations, outperforms the round-robin approach. This implies that biasing the initial configurations alone is not sufficient to improve the performance of dynasearch. The initial configurations also need to be biased by the "right" heuristic. One possible explanation for this unexpected result is that the runs that RR wastes on weaker heuristics are especially counterproductive, leading the search (on those runs) so far astray that it would be better to use no heuristic.

5.2 Resource Constrained Project Scheduling with Time Windows (RCPSP/max)

The RCPSP/max problem is defined as follows. Define $P =< A, \Delta, R >$ as an instance of RCPSP/max. Let A be the set of activities $A = \{a_0, a_1, a_2, \ldots, a_n, a_{n+1}\}$. Activity a_0 is a dummy activity representing the start of the project and a_{n+1} is similarly the project end. Each activity a_j has a fixed duration p_j, a start-time S_j, and a completion-time C_j which satisfy the constraint $S_j + p_j = C_j$. Let Δ be a set of temporal constraints between activity pairs $< a_i, a_j >$ of the form $S_j - S_i \in [T_{i,j}^{\min}, T_{i,j}^{\max}]$. The Δ are generalized precedence relations between activities. The $T_{i,j}^{\min}$ and $T_{i,j}^{\max}$ are minimum and maximum time-lags between the start times of pairs of activities. Let R be the set of renewable resources $R = \{r_1, r_2, \ldots r_m\}$. Each resource r_k has an integer capacity $c_k \geq 1$. Execution of an activity a_j requires one or more resources. For each resource r_k, the activity a_j requires an integer capacity $rc_{j,k}$ for the duration of its execution. An assignment of start-times to activities in A is time-feasible if all temporal constraints are satisfied and is resource-feasible if all resource constraints are satisfied. A schedule is feasible if both sets of constraints are satisfied. The problem is to find a feasible schedule with minimum makespan M where $M(S) = \max\{C_i\}$. That is, find a set of assignments to S such that $S_{\text{sol}} = \arg\min_S M(S)$. The maximum time-lag constraints are the source of difficulty—i.e., finding feasible solutions alone under these constraints is NP-Hard.

5.2.1 Experimental Setup

We begin by modifying Algorithm 2 to be relevant for a problem that is not only one of optimization but also one of constraint satisfaction. Pseudocode of the modified algorithm can be seen in Algorithm 5. The primary modifications can be seen in the additional lines 21–26. Specifically, there is now a constraint propagation step in line 21 and a backtracking step in line 26. Additionally, a rapid randomized restart [21] approach is taken (lines 23–25) to cut off the search after a pre-specified number of backtracks in favor of restarting along a randomized search trajectory. We specifically employ the constraint propagation and backtracking steps of Franck et al [18].[6] We use VBSS to bias the choice made by the heuristic at each decision point.

Five priority rules for the RCPSP/max problem are used as candidate search heuristics:

- smallest "latest start time" first, $\text{LST}_i = \frac{1}{1+LS_i}$;
- "minimum slack time" first, $\text{MST}_i = \frac{1}{1+LS_i-ES_i}$;
- "most total successors" first, $\text{MTS}_i = |\text{Successors}_i|$, where Successors_i is the set of not necessarily immediate successors of a_i in the project network;
- "longest path following" first, $\text{LPF}_i = \text{lpath}(i, n+1)$, where $\text{lpath}(i, n+1)$ is the length of the longest path from a_i to a_{n+1}; and
- "resource scheduling method", $\text{RSM}_i = \frac{1}{1+\max(0, \max_{g \in \text{eligible set}, g \neq i}(ES_i + p_i - LS_g))}$.

LS_i and ES_i are the latest and earliest start times. A few of these heuristics have been redefined from Neumann et al.'s definitions so that the eligible activity with the highest heuristic value is chosen. Eligible activities are those that can be time-feasibly scheduled given constraints involving already scheduled activities.

[6] More details of the constraint propagation and backtracking steps are also available Cicirello's dissertation [11].

Algorithm 5: QD-BEACON Multi-Heuristic VBSS in a Backtracking CSP Search

Input: Number of iterations I; a set of heuristics H; an "objective" function; the threshold parameter τ; a limit L on the number of backtracks before restarting; and a search-tree T.

Output: A solution S.

QD-BEACON/VBSS/BACKTRACKING(I, H, objective, τ, L, T)

(1) bestsofar $\leftarrow$ solution S obtained if heuristic $h_1 \in H$ followed from T
(2) **foreach** heuristic $h \in H - h_1$
(3) $S \leftarrow$ solution obtained if heuristic h is followed from T
(4) **if** objective(S) is superior to objective(bestsofar)
(5) bestsofar $\leftarrow S$
(6) **repeat** I times
(7) heuristic $\leftarrow$ SELECTHEURISTIC()
(8) $S \leftarrow$ root search-node of T
(9) count $\leftarrow 0$
(10) **while** S is a decision node of T
(11) **foreach** choice C from S
(12) score[C] $\leftarrow$ heuristic(C, S)
(13) totalweight $\leftarrow 0$
(14) **foreach** choice C from S
(15) weight[C] $\leftarrow$ bias(score[C])
(16) totalweight $\leftarrow$ totalweight + weight[C]
(17) **foreach** choice C from S
(18) prob[C] $\leftarrow$ weight[C] / totalweight
(19) **select** randomly the choice C biased according to prob[C]
(20) $S \leftarrow$ Successor(S, C)
(21) $S \leftarrow$ PROPAGATECONSTRAINTS(S)
(22) **if** INFEASIBILITYDETECTED(S)
(23) count $\leftarrow$ count + 1
(24) **if** count $\geq L$
(25) give up run and goto next iteration at line 6
(26) $S \leftarrow$ EXECUTEBACKTRACKINGSTEP(S)
(27) $S \leftarrow$ OPTIONALPOSTPROCESSINGSTEP(S)
(28) UPDATEAQDF(heuristic, objective(S), τ)
(29) **if** objective(S) is superior to objective(bestsofar)
(30) bestsofar $\leftarrow S$
(31) **return** bestsofar

Table 3: Summary of the RCPSP/max results.

Algorithm	NO	NF	Δ_{LB}
D-EXP[100]	649.7	1050.7	5.3
EXP[100]	646.3	1050.0	5.3
FASTER[100]	617.5	1044.1	5.5
NORM[100]	601.1	1012.6	5.8
RR[100]	572.2	1000.4	6.0
D-EXP[500]	665.7	1053.2	4.8
EXP[500]	658.2	1052.6	4.8
FASTER[500]	631.0	1045.0	5.2
NORM[500]	619.4	1043.1	5.3
RR[500]	598.9	1009.8	5.5
D-EXP[2000]	675.7	1057.0	4.6
EXP[2000]	669.9	1057.0	4.6
FASTER[2000]	654.0	1051.1	4.7
NORM[2000]	648.1	1050.6	4.9
RR[2000]	615.8	1016.2	5.2
[36]	679	1059	6.8
[10]	670	1057	8.0

Table 4: New best known solutions found by the algorithm of this paper. LB is the lower bound for the makespan. Old is the previous best known. New is the new best known.

Instance	LB	Old	New
C364	341	372	**365**
D65	440	539	**521**
D96	434	450	**445**
D127	428	445	**434**
D277	558	575	**569**

As before, we consider the following exploration policies: double exponentially increasing rate of allocations to the observed best heuristic (D-EXP); faster than double exponentially increasing rate (FASTER); and exponentially increasing rate (EXP); as well as a round-robin selection scheme (RR) where heuristics are selected in round robin order to seed successive runs of the search; and an approach that models heuristic performance under a normality assumption (NORM).

5.2.2 Results

Table 3 shows the results. We use a well-known benchmark problem set [35] consisting of 1059 feasible instances.[7] We execute each algorithm 10 times on each problem instance and report average results. Δ_{LB} is the average (of 10590 runs) relative deviation from the known lower bounds. NO and NF are the average number of optimal solutions and feasible solutions found (out of 1059). The results are comparable to the first problem domain. The worst performers were the round-robin approach to integrating multiple heuristics

[7] http://www.wior.uni-karlsruhe.de/LS_Neumann/Forschung/ProGenMax/rcpspmax.html

and the approach based on a normality assumption. The exponential rate of allocations to the observed best heuristic leads to over-exploration and the faster than double exponential rate leads to over-exploitation—both outperformed by the policy that allocates a double exponentially increasing number of trials to the observed best heuristic. These results are competitive with the current best known heuristic approaches (also shown in Table 3) for this NP-Hard problem (e.g., [10, 36]). Additionally, we have been able to find new best known solutions for some of the problem instances of this difficult benchmark set. Table 4 summarizes the new best known solutions for these problem instances, providing the makespan of the new best known solutions, along with the previous best known solutions and the known lower bounds for those problem instances.

6 Summary and Conclusions

In this paper, we presented two key foundational elements for integrating multiple search heuristics within a heuristically-biased stochastic sampling algorithm. We formalized the problem of selecting among multiple heuristics as the Max K-Armed Bandit problem, and analyzed the basic exploration-exploitation tradeoff under extreme value theory assumptions. These foundations are then used to design the framework that we call QD-BEACON.

The first of the foundations is based on our conjecture that the distribution of the quality of solutions produced by a heuristically-biased stochastic search procedure can best be modelled by a family of distributions motivated by extreme value theory. This leads to the use of the GEV distribution within our QD-BEACON framework. Our analysis indicates that the GEV provides a distribution family that fits solution quality data well.

The second foundational element is a key to the tradeoff between exploring to improve the performance models and exploiting the current models. In order to balance this tradeoff of exploration to improve our estimates of the GEV parameters for our solution quality distributions versus exploitation of the current model estimates, we have presented an analysis of the Max K-Armed Bandit that indicates we must allocate a double exponentially increasing rate of trials of the stochastic sampling search to the observed best search heuristic.

These foundations for multi-heuristic search are at the heart of our QD-BEACON framework for combining multiple search heuristics within a single stochastic search. The stochastic sampling search algorithm that the study employed during empirical validation was that of VBSS which is a non-systematic tree-based iterative search that uses randomization to expand the search around a given heuristic's prescribed trajectory through the search-space. The approach recommended by this paper, however, can be applied to other search algorithms that rely on the advice of a search heuristic and for any problem for which there is no one heuristic that dominates. In fact, our experiments with the RCPSP/max problem actually integrated multiple search heuristics within a hybrid of VBSS and a CSP backtracking search.

The experimental results validate the multi-heuristic theory presented in this paper. Specifically, heuristic selection based on learned models motivated by extreme value theory outperforms a simple round robin approach, and that the doubly exponential sampling rate consistently outperforms lesser and greater rates of allocation. The effectiveness of this approach was validated using two NP-Hard scheduling problems. Most notable are the results on the constrained optimization problem known as RCPSP/max. On standard benchmark RCPSP/max problems, our extreme value theory motivated approach was shown to be competitive with the current best known heuristic algorithms for the problem. Our approach is able to find closer to optimal results on average compared to SWO [36], although SWO optimally solves more instances on average. The approach we have taken in this paper has also improved upon current best known solutions to

difficult benchmark instances. Additionally, for the weighted tardiness scheduling problem, our approach is able to boost the performance of one of the state-of-the-art algorithms for the problem.

One obvious direction for future research is further theoretical analysis of the Max K-Armed Bandit problem. Our analysis has proceeded under the specific assumption that a Gumbel distribution is appropriate for modeling solution quality distributions. However, alternative distributional assumptions are possible, and can lead to different solution strategies. Some initial work in this direction is taken by Streeter and Smith [38, 37].

A second research direction of interest is the exploration of possible connections between the heavy-tailed nature of runtime distributions of CSP search algorithms noted by Gomes et al. [20] and the heavy-tailed nature of the solution quality distributions observed in our own work—the extreme value distributions type II & III are both heavy-tailed. Are the runtime and solution quality distributions in constrained optimization domains such as RCPSP/max at all correlated, and if so can this be used to enhance search? This is another interesting research question.

References

[1] R. Agrawal. The continuum-armed bandit problem. *SIAM Journal on Control and Optimization*, 33 (6):1926–1951, 1995.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. In *Proceedings of the 15th International Conference on Machine Learning*, pages 100–108. Morgan Kaufmann, 1998.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[5] J. E. Beasley. Weighted tardiness. In *OR-Library*. Imperial College Management School, 1998. Available at, http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html.

[6] D. A. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall, London, UK, 1985.

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

[8] J. L. Bresina. 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, 1996.

[9] A. Cesta, A. Oddi, and S. F. Smith. 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, 1999.

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

[11] V. A. Cicirello. *Boosting Stochastic Problem Solvers Through Online Self-Analysis of Performance*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[12] V. A. Cicirello and S. F. Smith. Amplification of search performance through randomization of heuristics. In *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, September 2002. doi:10.1007/3-540-46135-3_9.

[13] V. A. Cicirello and S. F. Smith. Heuristic selection for stochastic search optimization: Modeling solution quality by extreme value theory. In *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, September/October 2004. doi:10.1007/978-3-540-30201-8_17.

[14] V. A. Cicirello and S. F. Smith. The max k-armed bandit: A new model of exploration applied to search heuristic selection. In *The Proceedings of the Twentieth National Conference on Artificial Intelligence*, volume 3, pages 1355–1361. AAAI Press, July 2005.

[15] V. A. Cicirello and S. F. Smith. Enhancing stochastic search performance by value-biased randomization of heuristics. *Journal of Heuristics*, 11(1):5–34, January 2005. doi:10.1007/s10732-005-6997-8.

[16] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag, 2001.

[17] R. K. Congram, C. N. Potts, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, Winter 2002.

[18] B. Franck, K. Neumann, and C. Schwindt. Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spektrum*, 23:297–324, 2001.

[19] C. Gomes, B. Selman, and H. Kautz. 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, 1998.

[20] C. P. Gomes, B. Selman, and N. Crato. 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, 1997.

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

[22] J. H. Holland. Genetic algorithms and the optimal allocations of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[23] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975. Second Edition: MIT Press, 1992.

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

[25] B. A. Huberman, R. M. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, 3 January 1997.

[26] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

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

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

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

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

[31] NIST/SEMATECH. *e-Handbook of Statistical Methods*. NIST/SEMATECH, 2003. http://www.itl.nist.gov/div898/handbook/.

[32] A. Oddi and S. F. Smith. 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, 1997.

[33] W. Ruml. Incomplete tree search using adaptive probing. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 235–241, 4-10 August 2001.

[34] N. M. Sadeh, Y. Nakakuki, and S. R. Thangiah. 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, 1997.

[35] C. Schwindt. Project generator progen/max and psp/max-library, 2003. http://www.wior.uni-karlsruhe.de/ LS_Neumann/Forschung/ProGenMax/rcpspmax.html.

[36] T. B. Smith and J. M. Pyle. An effective algorithm for project scheduling with arbitrary temporal constraints. In *AAAI'04*, pages 544–549, 2004.

[37] M. J. Streeter and S. F. Smith. An asymptotically optimal algorithm for the max k-armed bandit problem. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, 2006.

[38] M. J. Streeter and S. F. Smith. A simple distribution-free approach to the max k-armed bandit problem. In *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming (CP 2006)*, 2006.

[39] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[40] J. P. Watson, L. Barbulescu, A. E. Howe, and L. D. Whitley. 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, 1999.