Heuristic Selection for Stochastic Search Optimization: Modeling Solution Quality by Extreme Value Theory

Vincent A. Cicirello[1] and Stephen F. Smith[2]

[1] Department of Computer Science, College of Engineering
Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104
cicirello@cs.drexel.edu
[2] The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
sfs@cs.cmu.edu

Abstract. The success of stochastic algorithms is often due to their ability to effectively amplify the performance of search heuristics. This is certainly the case with stochastic sampling algorithms such as heuristic-biased stochastic sampling (HBSS) and value-biased stochastic sampling (VBSS), wherein a heuristic is used to bias a stochastic policy for choosing among alternative branches in the search tree. One complication in getting the most out of algorithms like HBSS and VBSS in a given problem domain is the need to identify the most effective search heuristic. In many domains, the relative performance of various heuristics tends to vary across different problem instances and no single heuristic dominates. In such cases, the choice of any given heuristic will be limiting and it would be advantageous to gain the collective power of several heuristics. Toward this goal, this paper describes a framework for integrating multiple heuristics within a stochastic sampling search algorithm. In its essence, the framework uses online-generated statistical models of the search performance of different base heuristics to select which to employ on each subsequent iteration of the search. To estimate the solution quality distribution resulting from repeated application of a strong heuristic within a stochastic search, we propose the use of models from extreme value theory (EVT). Our EVT-motivated approach is validated on the NP-Hard problem of resource-constrained project scheduling with time windows (RCPSP/max). Using VBSS as a base stochastic sampling algorithm, the integrated use of a set of project scheduling heuristics is shown to be competitive with the current best known heuristic algorithm for RCPSP/max and in some cases even improves upon best known solutions to difficult benchmark instances.

1 Introduction

The success of stochastic sampling algorithms such as Heuristic-Biased Stochastic Sampling (HBSS) [1] and Value-Biased Stochastic Sampling [2, 3] stems from their ability to amplify the performance of search heuristics. The essential idea underlying these algorithms is to use the heuristic's valuation of various choices at a given search node to bias a stochastic decision, and, in doing so, to randomly perturb the heuristic's prescribed (deterministic) trajectory through the search space. In the case of HBSS, a rank-ordering of possible choices is used as heuristic bias; in VBSS, alternatively, the actual heuristic value attributed to each choice is used. This stochastic choice process enables generation of different solutions on successive iterations (or restarts), and effectively results in a broader search in the "neighborhood" defined by the deterministic heuristic. HBSS has been shown to significantly improve the performance of a heuristic for scheduling telescope observations [1]. VBSS has shown similar ability to improve search performance in weighted-tardiness scheduling [2, 3], resource-constrained project scheduling [3], and in a multi-rover exploration domain [4].

The drawback to stochastic sampling algorithms such as HBSS and VBSS is that they require identification of an appropriate domain heuristic, and search performance is ultimately tied to the power of the heuristic that is selected. Heuristics, however, are not infallible, and in most domains there does not exist a single dominating heuristic. Instead different heuristics tend to perform better or worse on different problem instances. In such cases, the choice of any single heuristic will ultimately be limiting and it would be advantageous to gain the collective power of several heuristics. The idea of exploiting a collection of heuristics to boost overall performance has been explored in other search contexts. Allen and Minton use secondary performance characteristics as indicators for which heuristic algorithm is performing more effectively for a given CSP instance [5]. Others have been applying relatively simple learning algorithms to the problem of selecting from among alternative local search operators [6, 7]. Work on algorithm portfolios [8] and the related A-Teams framework [9] take a more aggressive approach, executing several different heuristic search algorithms in parallel.

In this paper, we consider the problem of integrating multiple search heuristics within a stochastic sampling algorithm. Rather than carefully customizing a variant to use a composite heuristic, the approach taken here is to instead design a search control framework capable of accepting several search heuristics and self-customizing a hybrid algorithm on a per problem instance basis. Generally speaking, our approach views each solution constructed by the stochastic sampling algorithm as a sample of the expected solution quality of the base heuristic. Over multiple restarts on a given problem instance, we construct *solution quality distributions* for each heuristic, and use this information to bias the selection of heuristic on subsequent iterations. Gomes et al.'s analysis and use of runtime distributions has led to much success in constraint satisfaction domains [10]. In a similar way, a hypothesis of this paper is that solution quality distributions can provide an analagous basis for understanding and enhancing the performance of stochastic sampling procedures in solving optimization problems.

As suggested above, the search control framework developed in this paper uses online-generated statistical models of search performance to effectively combine multiple search heuristics. Consider that a stochastic search algorithm samples a solution space, guided by a strong domain heuristic. Our conjecture is that the solutions found by this algorithm on individual iterations are generally "good" (with respect to the overall solution space) and that these "good" solutions are rare events. This leads us to the body of work on extreme value theory (EVT). EVT is the statistical study of rare or uncommon events, rather than the usual (e.g., study of what happens at the extreme of a distribution in the tail). With our conjecture stated and with respect to EVT, we can view the distribution of solution qualities found by a stochastic heuristic search algorithm as a sort of snapshot of the tail of the distribution of solution qualities of the overall search space. We specifically employ a distribution called the Generalized Extreme Value (GEV) distribution to model the solution qualities given by individual iterations of the search algorithm. We implement this EVT-motivated approach in two ways: 1) using a well-known, but numerically intensive computation of a maximum likelihood estimation of the GEV; and 2) using kernel density estimation tuned assuming a GEV.

Using VBSS as a base stochastic sampling procedure, we validate this EVT-motivated heuristic performance modeling and heuristic selection policy on the NP-Hard problem of resource-constrained project scheduling with time windows (RCPSP/max). As a baseline and to validate our EVT assumptions, we compare the performance of the approach with one that makes the naive assumption that the solution qualities are normally distributed. We further benchmark the approach against several well-known heuristic algorithms, finding that our EVT-motivated algorithm is competitive with the current best known heuristic algorithm for RCPSP/max; and in some cases even improves upon current best known solutions to difficult problem instances.

2 Modeling a Solution Quality Distribution

2.1 Extreme Value Theory Motivation

Consider that the solutions to a hard combinatorial optimization problem computed on each iteration of a stochastic sampling algorithm are in fact at the extreme when the overall solution-space is considered. If one were to sample solutions uniformly at random, the probability is very low that any of the solutions generated by a stochastic search that is guided by a strong heuristic would be found. In other words, good solutions to any given problem instance from the class of problems of greatest interest to us are, in a sense, rare phenomena within the space of feasible solutions.

For example, using Bresina's concept of a quality density function (QDF) [11], we examined several problem instances of a weighted tardiness scheduling problem. A QDF is the distribution of solution qualities that one would obtain by sampling uniformly from the space of possible solutions to a problem instance. For easy problem instances from our problem set, we found that the optimal solution was on average over 6.4 standard deviations better than the average feasible solution to the problem; the value of the average solution given by a stochastic sampler guided by a strong domain heuristic was also over 6.4 standard deviations better than the average solution in the problem space [3]. Further, for hard problem instances, we found that the average solution given by a single iteration of the stochastic search was over 9.1 standard deviations better than the average random solution; the best known solution was on average 9.4 standard deviations better than the average solution in the problem space [3].

2.2 Generalized Extreme Value Distribution

With this noted, we turn to the field of extreme value theory, which deals with "techniques and models for describing the unusual rather than the usual" [12]. Consider an extreme value 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* [12]:

Theorem 1. *If there exists sequences of constants* $\{a_n > 0\}$ *and* $\{b_n\}$ *such that* $P((M_n - b_n)/a_n \leq z) \to G(z)$ *as* $n \to \infty$*, 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 known as the extreme value distributions, types I (Gumbel), II (Fréchet), and III (Weibull). The types II and III distributions are heavy-tailed – one bounded on the left and the other on the right. The Gumbel distribution is medium-tailed and unbounded. 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.

2.3 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 stochastic sampling algorithm using a strong heuristic to sample from the solution space behaves the same as (or at least similar to) the distribution of "block maxima" and thus its cumulative distribution function can be modeled by the GEV distribution.

To use the GEV as our model, we must first recognize that we have been assuming throughout that our objective function must be minimized so we need a "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 better solution than the best found so far (B) using heuristic i can be defined as:

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

where the b_i, a_i, and ξ_i are estimated from the negative of the sample values. To compute these parameters, we use Hosking's maximum-likelihood estimator of the GEV parameters [13]. In estimating the GEV parameters, Hosking's algorithm is called multiple times, if necessary. The first call uses initial estimates of the parameters as recommended by Hosking (set assuming a Gumbel distribution). If Hosking's algorithm fails to converge, then a fixed number of additional calls are made with random initial values of the parameters. If convergence still fails, we use the values of the parameters as estimated by assuming a type I extreme value distribution (the Gumbel distribution)[1].

2.4 Modeling Solution Quality Using Kernel Density Estimation

A second possibility for estimating the quality distribution is Kernel Density Estimation (see [14]). A kernel density estimator makes little, if any, assumptions regarding the underlying distribution it models. It provides a non-parametric framework for estimating arbitrary probability densities. The advantage of this approach is that it should be possible to more closely estimate arbitrary solution quality distributions. Kernel density estimation takes local averages to estimate a density function by placing smoothed out quantities of mass at each data point. The kernel density estimator is defined as:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h}). \tag{3}$$

$K(\cdot)$ is a kernel function and h is called the bandwidth (also sometimes called the scale parameter or spreading coefficient). The X_i are the n sample values (objective function value of the solutions generated by the n iterations of the stochastic search algorithm).

The kernel function we have chosen is the Epanechnikov kernel [15]:

$$K(x) = \tfrac{3}{4\sqrt{5}}(1 - \tfrac{x^2}{5}) \text{ for } |x| < \sqrt{5} \text{ and otherwise } 0\,. \tag{4}$$

Epanechnikov showed that this is the risk optimal kernel, but estimates using other smooth kernels are usually numerically indistinguishable. Thus, the form of the kernel can be chosen to best address computational efficiency concerns. In our case, the Epanechnikov kernel is a clear winner computationally for the following reasons:

- We are most interested in ultimately computing the probability of finding a better solution than the best found so far. This kernel function allows us to easily compute the cumulative probability distribution for arbitrary solution quality distributions.
- Due to the condition $|x| < \sqrt{5}$, only a limited number of sample values must be considered, reducing the computational overhead.

Although the choice of kernel function is not critical in terms of numerical results, the choice of bandwidth can be very crucial. Epanechnikov showed that the optimal choice of bandwidth is $h = (\frac{L}{nM})^{1/5}$, where $L = \int_{-\infty}^{\infty} K(x)^2\, dx$ where $M = \int_{-\infty}^{\infty} (f''(x))^2\, dx$, and where n is the number of samples [15]. Unfortunately, this

[1] It should be noted that this fallback condition appears to rarely occur, if ever, in our experiments.

computation depends on knowing the true distribution (M depends on $f(x)$). We assume that the underlying distribution is the Gumbel distribution. The reasoning behind this assumption follows our EVT motivation. Given the Gumbel distribution assumption, $M = \frac{1}{4a^5}$, where a is the scale parameter of the Gumbel. Note that the standard deviation of the Gumbel distribution is $\sigma = \frac{\pi a}{\sqrt{6}}$ [16]. From this, we have $a = \frac{\sigma\sqrt{6}}{\pi}$. We can now write M in terms of the sample standard deviation: $M = \frac{\pi^5}{4\sqrt{6}^5\sigma^5}$. We are using the Epanechnikov kernel so $L = \frac{3}{5\sqrt{5}}$. This results in a value of h computed as: $h = 0.79sn^{-1/5}$ where $s = \min\{\sigma, Q/1.34\}$ and where Q is the interquartile range.

We are interested in the cumulative distribution function for the purpose of computing the probability of finding a better solution than the best found so far. This can be obtained from integrating the kernel density estimator. Thus, we have the probability P_i of finding a solution better than the best found so far, B, given heuristic i[2]:

$$P_i = \int_0^B \frac{1}{n_i h_i} \sum_{j=1}^{n_i} K\left(\frac{x - S_{i,j}}{h_i}\right) dx. \tag{5}$$

Given our choice of the Epanechnikov kernel, this evaluates to:

$$\begin{aligned} P_i = \frac{3}{4n_i h_i\sqrt{5}} & \sum_{j,\left|\frac{B-S_{i,j}}{h_i}\right|<\sqrt{5}} \\ & ((B - \frac{1}{5h_i^2}(\frac{B^3}{3} - B^2 S_{i,j} + BS_{i,j}^2)) - \\ & (S_{i,j} - h_i\sqrt{5} - \frac{1}{5h_i^2}(\frac{(S_{i,j} - h_i\sqrt{5})^3}{3} - \\ & (S_{i,j} - h_i\sqrt{5})^2 S_{i,j} + (S_{i,j} - h_i\sqrt{5})S_{i,j}^2))) \end{aligned} \tag{6}$$

It should be noted, that if we maintain the samples in sorted order, then given that B must be less than or equal to the smallest value in this list[3], we compute this sum until we reach a sample $S_{i,j}$ such that $\left|\frac{B-S_{i,j}}{h_i}\right| \geq \sqrt{5}$. Once a sample for which this condition holds is reached in the list, the summation can end. Actually, rather than in a sorted list, we maintain the samples in a sorted histogram, maintaining counts of the number of samples with given discrete values.

2.5 Selecting a Search Heuristic

A method is needed for balancing the tradeoff between exploiting the current estimates of the solution quality distributions given by the algorithm's choices and the need for exploration to improve these estimates. The k-armed bandit focuses on optimizing the expected total sum of rewards from sampling from a multiarmed slot machine. At any point during an iterative stochastic sampling search, there is a current best found solution. Future iterations have the objective of finding a solution that is better than the

[2] This assumes a minimization problem with lower bound of 0 on the objective function.

[3] Assuming we are minimizing an objective function.

current best. From this perspective, a better analogy than the k-armed bandit would be to consider a multiarmed slot machine in which the objective is to sample the arms to optimize the expected best single sample – what we have termed the "Max k-Armed Bandit Problem" [3]. Elsewhere, we showed that the optimal sampling strategy samples the observed best arm at a rate that increases approximately double exponentially relative to the other arms [3].

Specific to our problem, this means sampling with the observed best heuristic with frequency increasing double exponentially relative to the number of samples given the other heuristics. Consider, as the exploration strategy, Boltzmann exploration as commonly used in reinforcement learning [17] and simulated annealing [18]. With a Boltzmann exploration strategy, we would choose to use heuristic h_i with probability $P(h_i)$:

$$P(h_i) = \frac{\exp((P_i F_i)/T)}{\sum_{j=1}^{H} \exp((P_j F_j)/T)}, \tag{7}$$

where P_i is the probability of finding a solution better than the best found so far, where F_i is the ratio of the number of feasible solutions used in estimating P_i to the total number of samples with i, where there are H heuristics to choose from, and where T is a temperature parameter. To get the double exponential sampling increase, we need to decrease T exponentially. For example, let $T = \exp(-N')$ where N' is the number of samples already taken and sample h_i with probability:

$$P(h_i) = \frac{\exp((P_i F_i)/\exp(-N'))}{\sum_{j=1}^{H} \exp((P_j F_j)/\exp(-N'))}. \tag{8}$$

3 Experimental Design

In this Section, consider the resource constrained project scheduling problem with time windows (RCPSP/max). RCPSP/max is the RCPSP with generalized precedence relations between start times of activities. It is a difficult makespan minimization problem well studied by the Operations Research community. Finding feasible solutions to instances of the RCPSP/max is NP-Hard, making the optimization problem very difficult.

RCPSP/max Formalization. The RCPSP/max problem can be defined formally 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 the 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 then to find a feasible schedule with minimum makespan M where $M(S) = \max\{C_i\}$. That is we wish to find a set of assignments to S such that $S_{\text{sol}} = \arg\min_S M(S)$. The maximum time-lag constraints are what makes this problem especially difficult. Particularly, due to the maximum time-lag constraints, finding feasible solutions alone to this problem is NP-Hard.

Branch-and-Bound Approaches. There are many branch-and-bound approaches for the RCPSP/max problem. Though for many problem instances it is too costly to execute a branch-and-bound long enough to prove optimality, good solutions are often obtained in a reasonable amount of computation time through truncation (i.e., not allowing the search to run to completion). The current (known) best performing branch-and-bound approach is that of Dorndorf et al. [19] (referred to later as B&B$_{DPP98}$).

Priority-Rule Methods. It should be noted that a priority rule method, as referred to here, is not the same as a dispatch policy. It actually refers to a backtracking CSP search that uses one or more priority-rules (dispatch heuristics) to choose an activity to schedule next, fixing its start time variable. The RCPSP/max is both an optimization problem and a CSP. When a start time becomes fixed, constraint propagation then takes place, further constraining the domains of the start time variables. The specific priority-rule method that we consider here is referred to as the "direct method" with "serial schedule generation scheme" [20, 21]. Franck et al. found the direct method with serial generation scheme to perform better in general as compared to other priority-rule methods.

The serial schedule generation scheme requires a priority-rule or activity selection heuristic. There are a wide variety of such heuristics available in the literature. Neumann et al. recommend five in particular. These five heuristics are those that we later randomize and combine within a single stochastic search:

- LST: smallest "latest start time" first: $\text{LST}_i = \frac{1}{1+LS_i}$.
- MST: "minimum slack time" first: $\text{MST}_i = \frac{1}{1+LS_i-ES_i}$.
- MTS: "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.
- LPF: "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}.
- RSM: "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 in these heuristics refers to the latest and earliest start times of the activities. Note that we have rephrased a few of these heuristics from Neumann et al.'s definitions so that for each, the eligible activity with the highest heuristic value is chosen. The eligible set of activities are those that can be time-feasibly scheduled given constraints involving already scheduled activities.

Truncating the search when a threshold number of backtracks has been reached and restarting with a different heuristic each restart has been proposed as an efficient and effective heuristic solution procedure. Later, we refer to the following multiple run truncated priority methods:

- PR$_{FNS5}$: Executing the direct method with serial generation scheme 5 times, once with each of the heuristics described above, and taking the best solution of the 5 as originally suggested by Frank et al. [20, 21]. Results shown later are of our implementation.
- PR$_{FN10}$: Similarly, this is a best of 10 heuristics. The results shown later are as reported by Dorndorf et al. [19] and Cesta et al. [22] of Franck and Neumann's best of 10 heuristics method [23][4].

Iterative Sampling Earliest Solutions. Cesta et al. present an algorithm for RCPSP/max that they call *Iterative Sampling Earliest Solutions (ISES)* [22]. ISES begins by finding a time feasible solution with a maximum horizon (initially very large) on the project's makespan, assuming one exists. The resulting time-feasible solution, for any interesting problem instance, is generally not resource-feasible. ISES proceeds by iteratively "leveling" resource-constraint conflicts. That is, it first detects sets of activities that temporally overlap and whose total resource requirement exceeds the resource capacity. Given the set of resource-constraint conflicts, it chooses one of the conflicts using heuristic-equivalency (i.e., chooses randomly from among all resource-conflicts within an "acceptance band" in heuristic value from the heuristically preferred choice). It then levels the chosen conflict by posting a precedence constraint between two of the activities in the conflicted set. It continues until a time-feasible and resource-feasible solution is found or until some resource-conflict cannot be leveled. This is then iterated some fixed number of times within a stochastic sampling framework. Then, given the best solution found during the the stochastic sampling process, the entire algorithm is repeated iteratively for smaller and smaller horizons. Specifically, the horizon is repeatedly set to the makespan of the best solution found so far until no further improvement is possible. Cesta et al. show ISES to perform better than the previous best heuristic algorithm for the RCPSP/max problem (namely PR$_{FN10}$).

Performance Criteria. The set of benchmark problem instances that we use in the experimental study of this Section is that of Schwindt[5]. There are 1080 problem instances in this problem set. Of these, 1059 have feasible solutions and the other 21 are provably infeasible. Each instance has 100 activities and 5 renewable resources. In the experiments that follow, we use the following performance criteria which have been used by several others to compare the performance of algorithms for the RCPSP/max problem:

- Δ_{LB}: the average relative deviation from the known lower bound, averaged across all problem instances for which a feasible solution was found. Note that this is based on the number of problem instances for which the given algorithm was able to find a feasible solution and thus might be based on a different number of problem instances for each algorithm compared. This criteria, as defined, is exactly as used by all of the other approaches to the problem available in the literature.

[4] Franck and Neumann's technical report describing this best of 10 strategy is no longer available according to both the library at their institution as well as the secretary of their lab. We have been unable to find out what the 10 heuristics are that produce these results.

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

- NO: the number of optimal solutions found. Currently, there are known optimal solutions for 789 of the 1080 problem instances.
- NF: the number of feasible solutions found. Of the 1080 problem instances, 1059 possess at least one feasible solution. The other 21 can be proven infeasible (e.g., by the preprocessing step of the priority-rule method).
- TIME: CPU time in seconds.

For all stochastic algorithms, values shown are averages across 10 runs. Values in parentheses are best of the 10 runs. In the results, as an added comparison point, we list the above criteria for the current best known solutions as BEST. Note that BEST is the best known prior to the algorithms presented in this paper. We further improve upon the best known solutions to some of the problem instances, but this is not considered in BEST.

Value-Biased Stochastic Sampling (VBSS). The first part of our approach uses an algorithm called VBSS [3, 2] to randomize the priority-rule method. Rather than following a priority rule deterministically during the course of the search, we bias a stochastic selection process by a function of the heuristic values. The backtracking priority-rule method is truncated as before when a threshold number of backtracks has been reached; and then restarted some number of times. The best feasible solution found of these restarts is chosen. In the results that follow, we refer to using the stochastic sampling framework VBSS within the priority-rule method for N iterations by: LST[N]; MST[N]; MTS[N]; LPF[N]; and RSM[N]. The bias functions used within VBSS are in each case polynomial: degree 10 for each of LST and MST; degree 2 for MTS; degree 3 for LPF; and degree 4 for RSM. These were chosen during a small number of exploratory solution runs for a small sample of problem instances. NAIVE[N] refers to randomly sampling an equal number of times with each of the five heuristics (N iterations total).

Generating and Using Models of Solution Qualities. Further, using the methods of modeling the distribution of solution qualities presented in this paper, we enhance the performance of the VBSS priority-rule method, effectively combining multiple heuristics within a single multistart stochastic search. We refer to this approach using the above five heuristics for N iterations according to the estimation method as follows: NORM[N] using Normal distribution estimates; KDE[N] using kernel density estimates; and GEV[N] using GEV distribution estimates.

4 Experimental Results

Table 1 shows a summary of the results of using VBSS with the priority-rule method and Table 2 shows a summary of the results of generating and using models of solution quality to enhance the search. We can make a number of observations:

- For any number of iterations of the VBSS enhanced priority-rule method, the best single heuristic to use in terms of finding optimal solutions is the “longest-path following first” (LPF) heuristic. However, we can also observe that the VBSS method using LPF is worst in terms of the number of feasible solutions found. Using LPF and VBSS appears to perform very well on the problem instances for which it can

Table 1. Summary of the results of using VBSS with the priority-rule method.

Algorithm	Δ_{LB}	NO	NF	TIME
LPF[20]	4.4 (4.2)	616 (628)	942 (956)	0.4
LST[20]	6.1 (5.7)	600.7 (612)	1041 (1043)	0.2
MTS[20]	4.6 (4.4)	600 (617)	953.3 (965)	0.4
MST[20]	6.6 (6.1)	598.3 (606)	1038.7 (1041)	0.3
RSM[20]	8.5 (7.7)	447.7 (494)	1027.3 (1031)	0.2
LPF[100]	4.2 (4.0)	632.3 (642)	959.7 (969)	1.1
MTS[100]	4.4 (4.3)	626 (638)	970 (981)	1.1
LST[100]	5.5 (5.2)	617.3 (625)	1044 (1044)	0.6
MST[100]	5.9 (5.6)	609 (614)	1042 (1043)	0.8
RSM[100]	7.4 (6.9)	510.3 (536)	1033.3 (1035)	0.6
LPF[200]	4.1 (4.0)	638.7 (647)	965 (974)	2.1
MTS[200]	4.3 (4.2)	634(648)	979 (986)	2.0
LST[200]	5.3 (5.1)	625.3 (633)	1044.3 (1045)	1.1
MST[200]	5.7 (5.5)	614.3 (623)	1043.3 (1044)	1.5
RSM[200]	7.1 (6.5)	529.7 (555)	1034.7 (1036)	1.0
LPF[400]	4.1 (4.0)	643.3 (650)	972.3 (980)	4.0
MTS[400]	4.3 (4.2)	641.7 (654)	983.7 (989)	3.7
LST[400]	5.2 (4.9)	631 (638)	1044.7 (1045)	1.9
MST[400]	5.5 (5.3)	619 (629)	1043.7 (1045)	2.8
RSM[400]	6.7 (6.3)	544 (564)	1035.7 (1037)	1.7

find feasible solutions, while at the same time having difficulties finding any feasible solution for a large number of other problem instances.

– We observe similar behavior when the second best heuristic in terms of number of optimal solutions found (VBSS using the "most total successors" (MTS)) is considered. However, like VBSS using LPF, VBSS using MTS performs poorly in terms of finding feasible solutions to the problems of the benchmark set.
– Although VBSS using any of the other three heuristics does not perform as well in terms of finding optimal solutions as compared to using LPF or MTS, using these other heuristics allows the search to find feasible solutions for many more of the problem instances as compared to using only LPF or MTS. Thus, we can see that by combining the five heuristics either by the naive strategy or by using quality models, that we can find feasible solutions to nearly all of the 1059 problem instances on average; while at the same time combining the strengths of the individual heuristics in terms of finding optimal, or near-optimal, solutions.
– Comparing the use of quality models to guide the choice of search heuristic to the naive strategy of giving an equal number of iterations to each of the heuristics, we see that the naive strategy is always the worst in terms of finding optimal solutions. Somewhat more interestingly, it is also always worst in terms of CPU time. Despite the overhead required for estimating the solution quality models, the naive strategy appears to be generally slower – as much as 2.5 seconds slower in the 2000 iteration case. The reason for this is that although there is extra computational overhead in the modeling, using the models gives less iterations to heuristics that appear less likely to find feasible solutions. The naive strategy results in more iterations that do

Table 2. Summary of the results of using VBSS and models of solution quality to enhance the priority-rule method.

Algorithm	Δ_{LB}	NO	NF	TIME
GEV[100]	5.3 (4.9)	650.7 (667)	1050.7 (1053)	0.8
KDE[100]	5.3 (4.9)	649.7 (662)	1050.7 (1053)	0.8
NORM[100]	5.3 (4.9)	648.7 (661)	1050.7 (1053)	0.8
NAIVE[100]	5.3 (5.0)	646.3 (650)	1050 (1052)	0.9
KDE[500]	4.8 (4.6)	665.7 (680)	1053 (1055)	3.1
NORM[500]	4.9 (4.6)	662.3 (673)	1053 (1055)	3.0
GEV[500]	4.9 (4.6)	660 (677)	1053 (1055)	3.2
NAIVE[500]	4.8 (4.6)	658.3 (666)	1052.7 (1055)	3.7
KDE[1000]	4.7 (4.5)	670.3 (683)	1054.7 (1057)	5.8
GEV[1000]	4.8 (4.5)	667(682)	1054.7 (1057)	6.5
NORM[1000]	4.8 (4.5)	666.7 (678)	1054.7 (1057)	5.8
NAIVE[1000]	4.7 (4.5)	664.7 (673)	1054.7 (1057)	7.0
KDE[2000]	4.6 (4.4)	675.7 (689)	1057 (1059)	11.2
NORM[2000]	4.7 (4.4)	672.3 (685)	1057 (1059)	11.0
GEV[2000]	4.7 (4.4)	672.3 (685)	1057 (1059)	13.0
NAIVE[2000]	4.6 (4.4)	669.7 (678)	1057 (1059)	13.5

not find a feasible solution, thus performing the maximum number of backtracking steps allowed by the serial generation scheme for such infeasible iterations.

- Of the three methods for estimation, kernel density estimation performs best for the RCPSP/max problem. Except for $N = 100$, KDE[N] finds more optimal solutions than the other considered methods. Furthermore, KDE[N] requires significantly less CPU time than does GEV[N] (at least for the particular estimation procedure of the GEV distribution employed here). Also, the additional overhead of KDE[N] compared to NORM[N] appears to be negligible given the CPU timing results.

Table 3 lists the results of a comparison of the enhanced priority-rule method and other algorithms, including branch-and-bound approaches and stochastic search algorithms. We can make the following observations:

- The best performing heuristic method is clearly KDE[N]. In approximately 1/6 of the CPU time used by the previous best performing heuristic method – ISES – KDE[1000] finds as many optimal solutions with a significantly lower average deviation from the known lower bounds. In less than 1/3 of the CPU time required by ISES, KDE[2000] consistently finds as many feasible solutions as ISES; KDE[2000] consistently finds more optimal solutions than ISES; and KDE[2000] on average finds solutions that deviate significantly less from the known lower bounds as compared to ISES.
- In approximately 1/6 of the CPU time, KDE[2000] on average performs as well as the current best branch-and-bound algorithm – B&B$_{DPP98}$ – in terms of deviation from lower bounds (and better than B&B$_{DPP98}$ for the best run of KDE[2000]). However, B&B$_{DPP98}$ finds more optimal solutions than KDE[2000]. KDE[2000] is a competitive alternative to truncated branch-and-bound if one requires good solutions but not necessarily optimal solutions in a highly limited amount of time.

Table 3. Comparison of the enhanced priority-rule method with other algorithms for the RCPSP/max problem.

Algorithm	Δ_{LB}	NO	NF	TIME
BEST	3.3	789	1059	–
B&B$_{DPP98}$	4.6	774	1059	66.7[a]
PR$_{FNS5}$	6.5	603	991	0.2
PR$_{FN10}$	7.7	601	1053	n/a[c]
ISES	8.0 (7.3)	669.8 (683)	1057 (1059)	35.7[b]
KDE[1000]	4.7 (4.5)	670.3 (683)	1054.7 (1057)	5.8
KDE[2000]	4.6 (4.4)	675.7 (689)	1057 (1059)	11.2

[a] Adjusted from original publication by a factor of $\frac{200}{300}$. The branch-and-bound algorithm was implemented on a 200 Mhz Pentium, while we used for our algorithms a Sun Ultra 10 / 300MHz.

[b] Adjusted from original publication by a factor of $\frac{266}{300}$. ISES was originally implemented on a Sun UltraSparc 30 / 266 MHz, while we used for our algorithms a Sun Ultra 10 / 300MHz.

[c] Timing results were not available in some cases. This is indicated by "n/a".

Table 4. New best known solutions found by the algorithms 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	Algorithm(s)
C364	341	372	**365**	MTS[100]
D65	440	539	**521**	KDE[2000], GEV[2000]
D96	434	450	**445**	LPF[20]
D127	428	445	**434**	LPF[200]
D277	558	575	**569**	KDE[2000], GEV[2000]

Table 4 lists the problem instances for which we were able to improve upon the current best known solutions. VBSS using the LPF heuristic is able to improve upon the best known solutions to a couple of problem instances. The same is true of VBSS using MTS. KDE[2000] and GEV[2000] also improve upon a couple additional best known solutions.

5 Summary and Conclusions

In this paper, we introduced a general framework for combining multiple search heuristics within a single stochastic search. The stochastic search algorithm that the study focused on was that of VBSS which is a non-systematic tree-based iterative search that uses randomization to expand the search around a heuristic's prescribed search-space region. 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 is obviously better than others.

In developing the approach to combining multiple search heuristics, we have conjectured that the distribution of the quality of solutions produced by a stochastic search algorithm that is guided by a strong domain heuristic 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 framework. Two methods of implementing the GEV have been considered: 1) maximum likelihood estimates computed by potentially costly numerical methods; and 2) kernel density estimation using a bandwidth parameter tuned under the assumption of a GEV distribution.

The effectiveness of this approach was validated using the NP-Hard constrained optimization problem known as RCPSP/max. On standard benchmark RCPSP/max problems, our EVT-motivated approach was shown to be competitive with the current best known heuristic algorithms for the problem. The best available truncated branch-and-bound approach is capable of finding a greater number of optimal solutions, but at a much greater computational cost. Our EVT-motivated approach is, however, able to find more optimal solutions than ISES (the previous best known heuristic algorithm for RCPSP/max) and with less deviation than ISES from the known lower bounds on solution quality. The approach we have taken in this paper has also improved upon current best known solutions to difficult benchmark instances.

One potentially interesting future direction to explore is whether or not there is any connection between the heavy-tailed nature of runtime distributions of CSP search algorithms noted by Gomes et al. [10] 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 at all correlated, and if so can this be used to enhance search? This is a direction that will be worth exploring.

References

1. Bresina, J.L.: 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, AAAI Press (1996) 271–278
2. Cicirello, V.A., Smith, S.F.: Amplification of search performance through randomization of heuristics. In Van Hentenryck, P., ed.: Principles and Practice of Constraint Programming – CP 2002: 8th International Conference, Proceedings. Volume LNCS 2470 of Lecture Notes in Computer Science., Springer-Verlag (2002) 124–138 Ithaca, NY.
3. Cicirello, V.A.: Boosting Stochastic Problem Solvers Through Online Self-Analysis of Performance. PhD thesis, The Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (2003) Also available as technical report CMU-RI-TR-03-27.
4. Goldberg, D., Cicirello, V., Dias, M.B., Simmons, R., Smith, S., Stentz, A.: Market-based multi-robot planning in a distributed layered architecture. In: Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings of the 2003 International Workshop on Multi-Robot Systems. Volume 2., Kluwer Academic Publishers (2003) 27–38 Washington, DC.
5. Allen, J.A., Minton, S.: Selecting the right heuristic algorithm: Runtime performance predictors. In: Proceedings of the Canadian AI Conference. (1996)
6. Cowling, P., Kendall, G., Soubeiga, E.: Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R., eds.: Applications of Evolutionary Computing: EvoWorkshops 2002 Proceedings. Number LNCS 2279 in Lecture Notes in Computer Science, Springer-Verlag (2002) 1–10

7. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In Resende, M.G.C., de Sousa, J.P., eds.: Metaheuristics: Computer Decision Making. Kluwer Academic Publishers (2003)
8. Gomes, C.P., Selman, B.: Algorithm portfolios. Artificial Intelligence **126** (2001) 43–62
9. Talukdar, S., Baerentzen, L., Gove, A., de Souza, P.: Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics **4** (1998) 295–321
10. Gomes, C.P., Selman, B., Crato, N.: Heavy-tailed distributions in combinatorial search. In: Principles and Practices of Constraint Programming (CP-97). Lecture Notes in Computer Science, Springer-Verlag (1997) 121–135
11. Bresina, J., Drummond, M., Swanson, K.: Expected solution quality. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1995) 1583–1590
12. Coles, S.: An Introduction to Statistical Modeling of Extreme Values. Springer-Verlag (2001)
13. Hosking, J.R.M.: Algorithm AS 215: Maximum-likelihood estimation of the paramaters of the generalized extreme-value distribution. Applied Statistics **34** (1985) 301–310
14. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Monographs on Statistics and Applied Probability. Chapman and Hall (1986)
15. Epanechnikov, V.A.: Non-parametric estimation of a multivariate probability density. Theory of Probability and Its Applications **14** (1969) 153–158
16. NIST/SEMATECH: e-Handbook of Statistical Methods. NIST/SEMATECH (2003) http://www.itl.nist.gov/div898/handbook/.
17. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285
18. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220** (1983) 671–680
19. Dorndorf, U., Pesch, E., Phan-Huy, T.: A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. Management Science **46** (2000) 1365–1384
20. Franck, B., Neumann, K., Schwindt, C.: Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. OR Spektrum **23** (2001) 297–324
21. Neumann, K., Schwindt, C., Zimmermann, J.: Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag (2002)
22. Cesta, A., Oddi, A., Smith, S.F.: A constraint-based method for project scheduling with time windows. Journal of Heuristics **8** (2002) 109–136
23. Franck, B., Neumann, K.: Resource-constrained project scheduling with time windows: Structural questions and priority-rule methods. Technical Report WIOR-492, Universität Karlsruhe, Karlsruhe, Germany (1998)