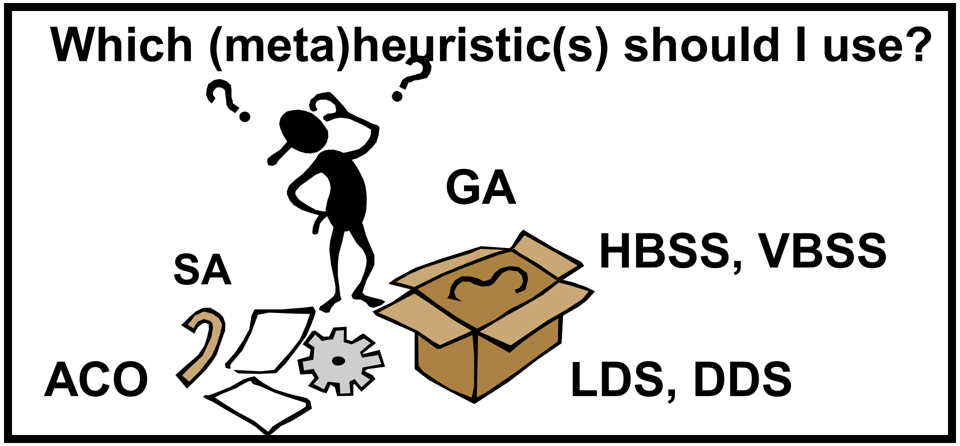
Statistically Modeling the Performance of a Multistart Randomized Heuristic Algorithm

Vincent A. Cicirello
Computer Science & Information Systems
Richard Stockton College

Exploiting Multiple Problem Solvers in Hard Combinatorial Domains

- Large-scale combinatorial optimization problems pose difficult challenges
- Algorithms guaranteed to find optimal solutions often too costly
- Can turn to heuristic and metaheuristic algorithms for quality / cost tradeoff
- But which metaheuristic to use?



- Can we exploit the collective power of several?

Properties of Problems We Have Explored So Far

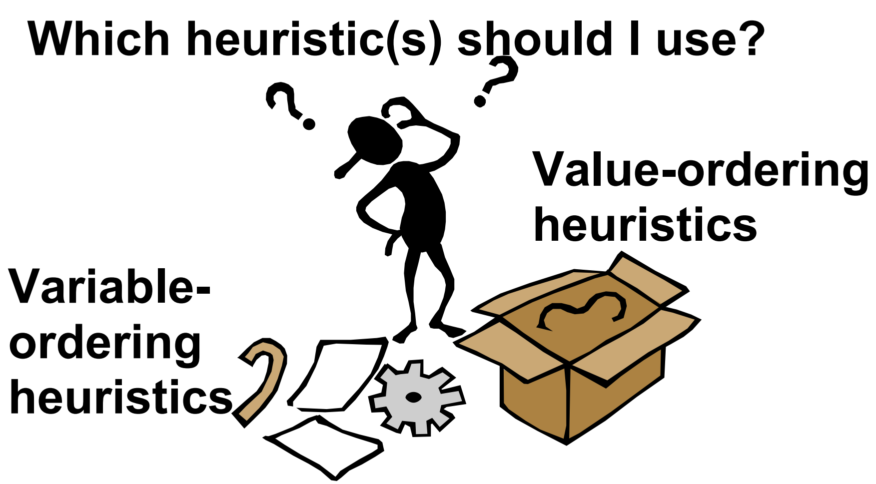
- Combinatorial Optimization problems, such that:
  - Generating optimal or near-optimal solutions is hard
  - Generating **some** solution is easy
    - For example, for the TSP, it is easy to generate **some** tour of the cities.
- An Example: Weighted Tardiness Sequencing
  - Any permutation of the jobs on the machine is a legal solution
  - Minimizing the weighted tardiness objective is NP-Hard in the strong sense
  - Also have explored the sequence-dependent setup version of problem

Heuristically Biased Stochastic Sampling Search

- **Heuristically biased stochastic sampling search algorithms**
  - Construct a solution with randomized process
    - e.g., randomly select among remaining jobs which to add to the end of the sequence
  - Heuristic used to bias the random choice process
    - **HBSS** [Bresina 96] - choices are biased by *rank ordering*
    - **VBSS** [Cicirello & Smith 02] - choices are biased by *heuristic value*
  - Retain the best solution after multiple restarts
    - i.e., retain the solution with the lowest value of objective (if a minimization problem)

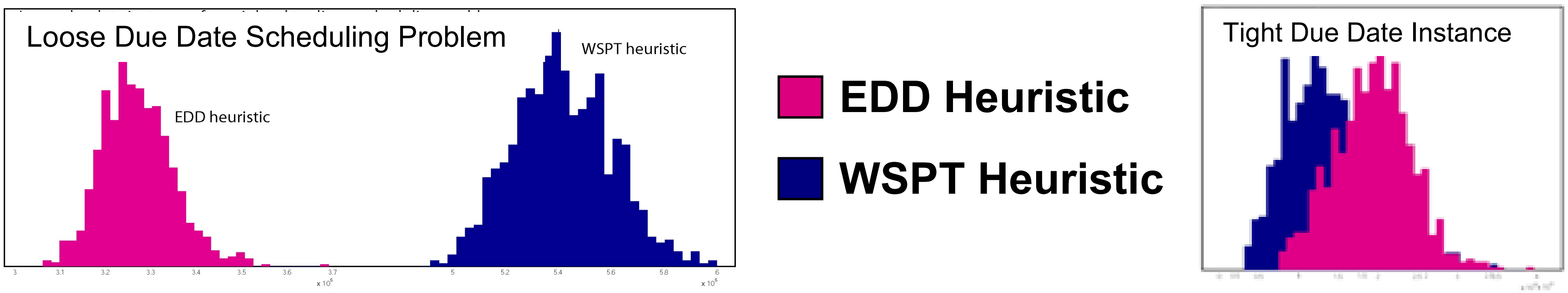
Search Heuristics in Combinatorial Domains

- Search heuristics provide a basis for efficiently reaching high-valued solutions in many combinatorial domains
- Heuristics are not infallible
- In many domains, no single heuristic dominates
  - **How can we exploit the collective power of several heuristics?**



Multi-Heuristic Stochastic Search

In many domains no single heuristic dominates



Approach:

- View each generated solution as a sample of the distribution of solution qualities of the heuristic
- Estimate solution quality distributions of various heuristics over multiple restarts
- Use these performance models to select base heuristic on subsequent iterations

Key Question 1

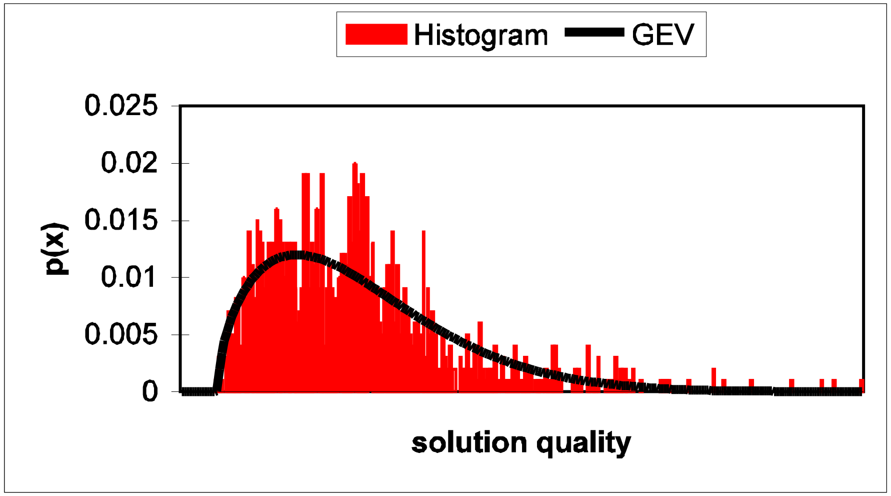
- How should we model the distribution of solution qualities?

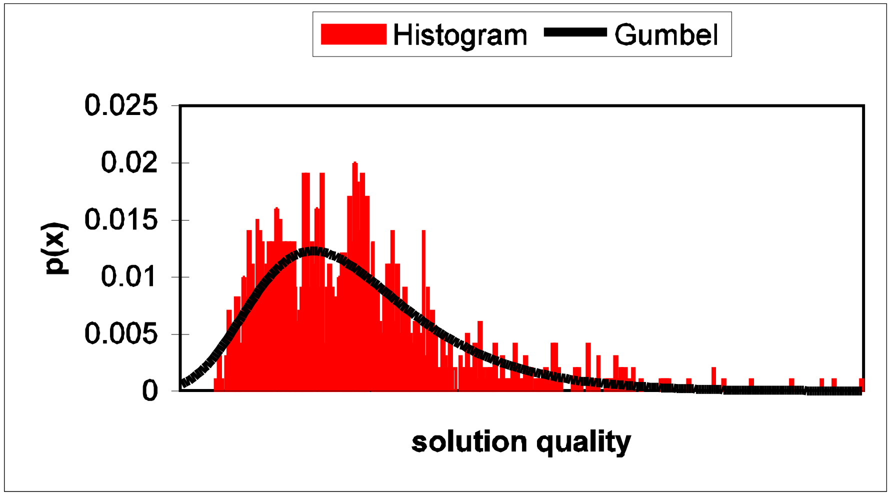
Modeling Solution Quality Using Extreme Value Theory

- ***Assumption 1:*** *In the space of solutions to an instance of an NP-Hard optimization problem, "good" solutions are rare phenomena.*
- ***Assumption 2:*** *A strong domain heuristic generally produces "good" solutions.*
- Extreme value theory model of quality of solutions produced by heuristic search algorithm
- **Generalized extreme value (GEV) distribution:**
  - Gumbel (type I), Frechet (type II), Weibull (type III)

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

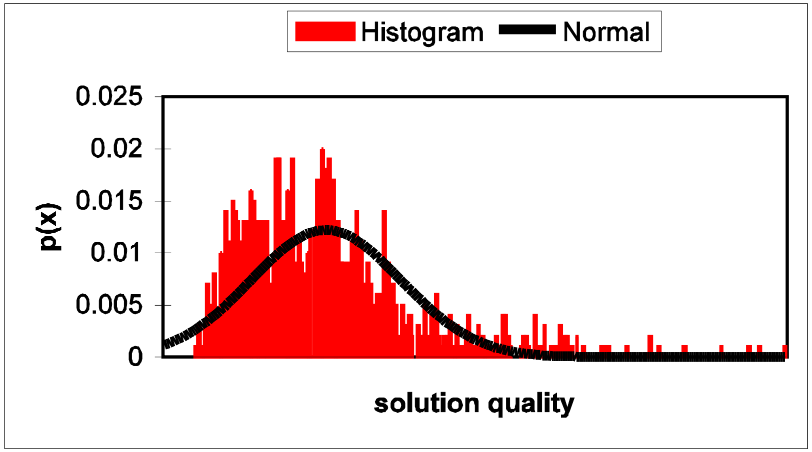
Validating The GEV Model of Heuristic Performance





Significance level: α = 0.001

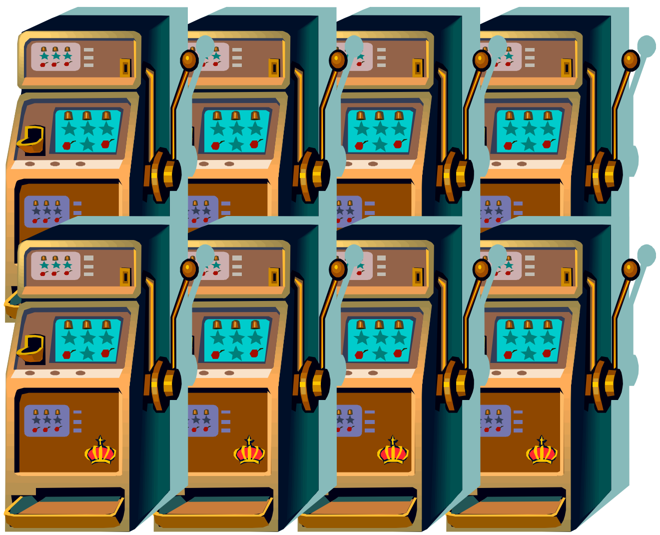
Distribution	X^2	d.o.f.	Upper critical
GEV	17.32	18	42.31
Gumbel	45.82	19	43.82
Normal	84.26	19	43.82
Lognormal	84.26	19	43.82
Exponential	90.36	19	43.82



Key Question 2

- How to allocate trials to various heuristics to balance
  - Exploration to improve models of heuristic performance
  - Exploitation of those models for efficient search

The K-Armed Bandit Problem



- Given…
  - slot machine with k arms
  - different payoff distribution for each
- **What's the optimal sampling policy to maximize expected sum of rewards?**

Allocate trials to the observed best arm at a rate increasing exponentially [Holland, '75]

But…

- Our interest though is in best single trial…
  - After *M* restarts of our search algorithm…
    - We have the best solution of the *M*…
    - We don’t care about the other M-1 solutions…
  - Goal of restarts *M+1* through N…
    - Find an even better solution

Max K-Armed Bandit

[Cicirello & Smith, AAAI '05]

The Max k-Armed Bandit:

- Slot machine with k arms, where…
- We only get to keep the best single sample reward after playing for some length of time
- To determine optimal policy must make assumptions about the payoff distribution of different arms
  - Assumption tied to application to multi-heuristic search (i.e., assume a GEV distribution)

Theorem: *Optimal strategy samples observed best heuristic at a doubly exponential rate relative to other heuristics*

Experimental Design:

Search Problem: Weighted Tardiness Sequencing
Objective: Minimize weighted tardiness

- *Min* $\sum w_i \, Max(0, c_i - d_i)$

Heuristics:

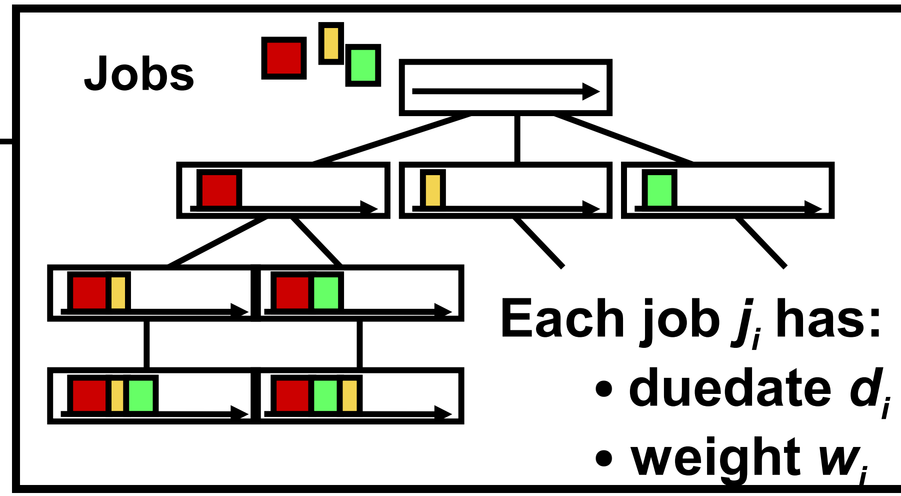
- COVERT, R&M, EDD, WSPT

Benchmark Set:

- 125 instances, 100 jobs

Search Algorithm:

- VBSS + local search
- Extreme value theory to model heuristic performance



Experimental Results:

Algorithm	# best known	ARPD	Time (sec)
VBSS + iterated dynasearch	**123**	**0.002**	**5.61**
VBSS + dynasearch	107	0.11	17.40
Dynasearch	73	1.16	18.21
Iterated Dynasearch	117	0.005	5.56
Genetic Algorithm	77	0.03	11.94
Simulated Annealing	59	0.12	12.96
Tabu Search	103	0.04	12.03
Hill Climbing	86	0.06	12.54

Other Problems Investigated

- Weighted Tardiness Scheduling with Sequence-Dependent Setups
- Resource Constrained Project Scheduling with Arbitrary Precedence Constraints
  - Added challenge: finding **any** feasible solution is NP-complete
  - Approach: integrate our multi-heuristic approach with constraint satisfaction search
- Starting to investigate graph theory problems:
  - E.g., largest common subgraph problem

Future and Ongoing Work

- Coordinating multiple crossover operators within a genetic algorithm
  - First steps taken: benchmarking several permutation crossover operators for a sequencing problem (Cicirello, GECCO'06)
- Extending the approach to coordinating multiple problem solving algorithms
  - E.g., allocating computational resources among multiple problem solving algorithms (e.g., a GA, SA, VBSS, etc)

Questions?

EXTRA SLIDES

Max K-Armed Bandit Sampling Policy

[Cicirello & Smith, AAAI ‘05]

Theorem: *Optimal strategy samples observed best heuristic at a doubly exponential rate relative to other heuristics*

Optimal Exploration Policy:

- Boltzmann exploration with exponential cooling schedule
- Choose heuristic h_i on iteration j with probability:

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

- R_i is an estimate of the probability that heuristic h_i produces a solution of quality better than some threshold quality τ.

Experimental Results:

400 restarts

Policy	# best known	ARPD
EXP	85.0	0.14
D-EXP	**94.3**	**0.12**
>D-EXP	78.7	0.19

1600 restarts

Policy	# best known	ARPD
EXP	95.0	0.12
D-EXP	**107.3**	**0.11**
>D-EXP	87.5	0.16

ARPD: Average Relative Percentage Deviation
from best known solutions

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

Experimental Results:

Algorithm	# best known	ARPD	Time (sec)
D-EXP (VBSS + dyna)	107	0.11	17.40
D-EXP (VBSS + i-dyna)	**123**	**0.002**	**5.61**
Dynasearch	73	1.16	18.21
Iterated Dynasearch	117	0.005	5.56
Genetic Algorithm	77	0.03	11.94
Simulated Annealing	59	0.12	12.96
Tabu Search	103	0.04	12.03
Hill Climbing	86	0.06	12.54