Toward a Descriptive Model Of Local Search Cost in Job-Shop Scheduling

Jean-Paul Watson
Computer Science Department
Colorado State University
Fort Collins, CO 80538 USA
watsonj@cs.colostate.edu

J. Christopher Beck
ILOG, S.A.
9, rue de Verdun, B.P. 85
F-94523 Gentilly Cedex France
cbeck@ilog.fr

Adele E. Howe and **Darrell Whitley**
Computer Science Department
Colorado State University
Fort Collins, CO 80538 USA
{howe,whitley}@cs.colostate.edu

Abstract

Many search techniques are available for solving scheduling problems. Yet, little is known about what problem features influence the cost of search. We study a descriptive cost model of local search in the job-shop scheduling problem (JSP), borrowing from the local search cost models available for random 3-SAT. We show that several factors known to influence the difficulty of local search in random 3-SAT directly carry over to the general JSP, including the number of solutions, backbone size, the distance to the nearest solution, and an analog of backbone robustness. However, in applying the same analysis to JSPs with structured constraints, we find that many of these factors only weakly influence local search cost. While the factors for the SAT cost models provide an accurate description of local search cost in the general JSP, our results for JSPs with structured constraints raise concerns regarding the applicability of cost models derived using random problems to those exhibiting specific structure.

1 Introduction

We study descriptive cost models of local search for the well-known job-shop scheduling problem (JSP). Although many state-of-the-art JSP algorithms are based on local search techniques (e.g., [Nowicki and Smutnicki, 1996]), very little is known about which search space features influence the search cost. In contrast to the JSP, researchers have expended significant effort in recent years to produce relatively accurate descriptive models of local search cost for random 3-SAT [Singer *et al.*, 2000]. Intuitively, we would expect some factors present in the SAT cost models, including the number of optimal solutions and the distance between sub-optimal and optimal solutions, to influence the difficulty of local search in other optimization problems such as the JSP. Yet, scheduling problems differ in many important aspects from SAT, making the applicability of these factors unclear a-priori.

Our research investigates whether or not the SAT models can be leveraged in an effort to understand local search cost in the JSP. Concurrently, we are also asking the question “Are there problem-independent factors influencing local search cost?”. We demonstrate that the factors influencing local search cost in SAT also influence local search cost in the JSP. These factors include the number of solutions [Clark *et al.*, 1996], backbone size [Parkes, 1997], the distance to the nearest solution, and an analog of backbone robustness [Singer *et al.*, 2000]. Together, these factors provide a relatively accurate model of local search cost in the JSP.

In contrast to random 3-SAT, constraints in real-world scheduling problems are often structured. We apply the same analysis to JSPs with workflow, which possess structured constraints. Here, we find that while the number of solutions and backbone size continue to influence search cost, the distance to the nearest solution has a surprisingly weak influence on local search cost. We conclude by discussing the implication of these contrasting results.

2 Descriptive Cost Models for Random 3-SAT

High-performance local search algorithms for SAT were introduced in the early 1990’s and continue to represent the state-of-the-art. Subsequently, significant effort has been devoted to understanding the factors influencing search cost for these algorithms.

The problem backbone is a key concept underlying SAT cost models. The *backbone* of a SAT instance consists of the subset of literals that have the same truth value in *all* solutions. [Parkes, 1997] showed that the proportion of literals appearing in the backbone has a strong influence on local search cost. When the backbone is small, there is a strong (negative) log-log correlation between the number of solutions and the local search cost. However, this correlation nearly vanishes when the backbone is large [Singer *et al.*, 2000].

The number of solutions to a problem instance says little about the topology of sub-optimal solutions, although this topology clearly influences local search cost. In SAT, local search algorithms quickly locate sub-optimal *quasi-solutions*, with relatively few unsatisfied clauses. These quasi-solutions form a largely inter-connected region of the search space (containing all global optima); once a point in this sub-space is identified, local search algorithms for SAT typically restrict search to the sub-space. This observation led Singer et al. to hypothesize that the size of this sub-space dictates the overall search cost.

To test this hypothesis, Singer et al. measured the mean Hamming distance between the first quasi-solution encountered by a local search algorithm and the nearest global optimum, and computed the correlation between this measure and the logarithm of local search cost. The resulting correlations were extremely high ($r \approx 0.95$) for problems with small backbones, degrading slightly for problems with larger backbones ($r \approx 0.75$); no explanation for the degradation was provided.

While the distance from a quasi-solution to the nearest global optimum does estimate the size of the quasi-solution sub-space, it does not provide any intuition into the problem features that *cause* the sub-space size to vary among SAT instances. Singer et al. posited a causal explanation based on the notion of *backbone robustness*. A SAT instance is said to have a *robust* backbone if a substantial number of clauses must be deleted before the backbone size is reduced by half. Conversely, an instance is said to be backbone *fragile* if the deletion of a few clauses reduces the backbone size by half.

Singer et al. argue that "backbone fragility approximately corresponds to how extensive the quasi-solution area is" ([Singer *et al.*, 2000], p. 251). Intuitively, a fragile backbone allows for large Hamming distances between quasi-solutions and solutions because of the sudden drop in backbone size. Due to the small backbone, local search algorithms can easily find elements in the quasi-solution subspace. But because these elements are distant from actual solutions, the search cost is relatively high. In contrast, a robust backbone implies a gradual fall in backbone size, and therefore no sudden jumps in Hamming distance.

For large-backboned SAT instances, Singer et al. demonstrated a moderate (≈ 0.5) negative correlation between backbone robustness and the log of local search cost. Surprisingly, this correlation degraded as the backbone size was decreased. Here, Singer et al. hypothesize that "finding the backbone is less of an issue and so backbone fragility, which hinders this, has less of an effect" (p. 254), although this conjecture was not explicitly tested.

3 Problem Difficulty and the JSP

We consider the well-known $n \times m$ JSP, in which n jobs must be processed exactly once on each of m machines for a pre-specified duration. Each machine can only process one job at a time, and pre-emption is disallowed. All jobs can start at time 0 and the standard optimization problem is to minimize the *makespan*, the maximum end time of all jobs.

In the *general* JSP, the machine processing orders for each job are generated at random; a more constrained form of the processing orders are found in JSPs with *workflow* partitions. In a JSP with workflow, the machines are typically partitioned into two equally-sized subsets consisting of machines 1 through $m/2$ and $m/2 + 1$ through m, respectively. Every job must then be processed on all machines in the first partition before proceeding to any machine in the second.

Later in this paper, we make a distinction between random and structured JSPs. Specifically, we refer to regularities in the machine processing orders, which are completely random in general JSPs, and have a specific form of structure in JSPs with workflow. This is slightly different from the notion of problem structure introduced in [Watson *et al.*, 1999], where structure refers to regularities in the operation durations.

Many high-quality search algorithms have been developed for the JSP [Jain and Meeran, 1999], largely through competitive testing on a widely used benchmark suite [Taillard, 1993]. Independent of algorithm, two observations regarding relative problem difficulty have emerged:

1. "square" problem instances ($n/m \approx 1$) are significantly harder than "rectangular" instances ($n/m > 1$)
2. given fixed n and m, workflow JSPs are substantially more difficult than general JSPs.

Yet, almost no research has been devoted to explaining the differences in terms of the underlying search space. The sole exception is the study of [Mattfeld *et al.*, 1999], which identified differences in the search spaces of some well-known 50×10 general and workflow JSPs. However, no causal link between these factors and local search cost was established. A goal of our research is to account for the relative difficulty of square versus rectangular JSPs, general versus workflow JSPs, and for variance in local search cost for different problem instances of the same size and workflow configuration.

4 Adapting SAT Analysis to the General JSP

In this section, we adopt the experimental methodology that Singer et al. used to develop their SAT local search cost model and apply it to the general JSP.

4.1 Defining a Backbone for JSP

The disjunctive graph representation is a commonly used encoding for the JSP [Balas, 1965]. There are $n(n-1)/2$ boolean "order" variables for each of the m machines, representing the precedence relations between all distinct pairs of jobs on a machine. We define the *backbone* of a JSP, therefore, as the set of order variables that have the same truth value in all solutions at the optimal makespan. We define the *backbone size* as the fraction of the possible $mn(n-1)/2$ order variables that are fixed to the same value in all optimal solutions.

4.2 Test Problems and Algorithms

Given the requirement to enumerate all optimal solutions to a problem, we are restricted to relatively small problems in our experiments. We consider two problem sizes: 6×4 and 6×6. Operation durations are sampled uniformly from the interval $[1, 99]$ and machine processing orders are randomly generated, following [Taillard, 1993]. While small, the problem sizes we consider are representative of some simpler JSP benchmark suites [Beasley, 1990], and many of the instances we generated were significantly harder than the benchmark problems of equal size (as measured by the mean CPU time required for Nowicki and Smutnicki's (1996) tabu search algorithm to locate an optimal solution).

Pilot experiments suggested a strong influence of backbone size on local search cost in the JSP. Unfortunately, even for the problems we consider, it is computationally infeasible to control for a specific backbone size: solution evaluation and construction in the JSP is considerably more expensive than in SAT. Instead, we filter for problems within $\pm 5\%$ of a target backbone size X, $0.0 \leq X \leq 1.0$. We denote the backbone size of the resulting set by $\approx X$. For each problem size, we generated 100 instances at each of the following backbone

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	Mean Number of Optimal Solutions				
6×4	481837 ± 1158660	30007 ± 38072	3221 ± 3742	642 ± 1374	21 ± 22
6×6	6233821 ± 8070114	1405290 ± 3221150	85292 ± 157617	9037 ± 9037	85 ± 106
	Mean Local Search Cost				
6×4	6.69 ± 3.34	23.05 ± 20.93	52.64 ± 61.22	94.76 ± 136.56	312.27 ± 332.27
6×6	8.34 ± 4.13	32.94 ± 32.58	53.43 ± 58.52	83.98 ± 91.80	514.70 ± 1853.08
	log_{10}-log_{10} correlation (r) between the number of solutions and local search cost				
6×4	-0.7508	-0.5100	-0.4905	-0.4131	-0.2683
6×6	-0.7328	-0.4865	-0.3807	-0.3227	-0.2010

Table 1: The mean number of solutions, mean local search cost, and log_{10}-log_{10} correlation (r) between the number of solutions and local search cost for general JSPs. $X \pm Y$ denotes a mean of X with a standard deviation of Y.

sizes: ≈ 0.1, ≈ 0.3, ≈ 0.5, ≈ 0.7, and ≈ 0.9. We refer to the 100 instances at a given problem and backbone size as a *problem class*.

Good local search algorithms for the JSP vary along two primary dimensions: the meta-heuristic and the move operator. Many state-of-the-art local search algorithms use tabu search as the meta-heuristic [Jain and Meeran, 1999]. Among these, the choice of move operator varies considerably. Unfortunately, most advanced move operators induce search spaces that are *disconnected*, where it is not always possible to move between randomly generated initial points and a global optimum. And algorithms using such move operators are not Probabilistically Approximately Complete (PAC) [Hoos, 1998], which severely complicates algorithm analysis.

For our local search algorithm, we use tabu search in conjunction with the move operator introduced by [Laarhoven *et al.*, 1992] (often denoted $N1$), which induces a provably connected search space. The $N1$ neighborhood is generated by swapping all adjacent pairs of jobs on a single randomly selected critical path (and belonging to the same critical block). Our tabu search meta-heuristic is based on the algorithm of [Nowicki and Smutnicki, 1996], which prevents recently swapped pairs of adjacent jobs from being re-established. No elite solution recovery (the diversification mechanism used by [Nowicki and Smutnicki, 1996]) is performed. The algorithm initially proceeds from a randomly generated active solution [Giffler and Thompson, 1960]. A single iteration of our algorithm consists of the evaluation of all $N1$ moves, and the selection of the best non-tabu move available.

[Nowicki and Smutnicki, 1996] and other JSP researchers generally use fixed-size tabu lists. However, in our pilot experiments, we often found JSP instances where the optimal solution was never located (even after several million iterations), and special-purpose code failed to detect, and therefore escape, cycles in the search. Additional experiments indicated that by using slightly different tabu list sizes, the instances were easily solved. Therefore, we use a dynamic tabu list length, where the length is sampled uniformly from the interval $[8, 12]$ (representing small deviations from our fixed-length tabu list size of 10) after every 25 iterations of the algorithm.

Following Singer et al. we take the local search cost of a JSP instance as the median number of tabu search iterations required to find an optimal solution, over 1000 independent runs. We note that the medians are not completely stable at this sampling rate. Complete stabilization requires roughly $10,000$ runs, but even the computational cost of $1,000$ runs for each of our problem instances is nearly prohibitive (requiring 6 CPU hours on 600 MHz Pentium III's for the more difficult ≈ 0.9 backbone 6×6 instances).

4.3 Number of Solutions and Search Cost

Clark et al. first showed that the number of solutions influences the cost of local search in SAT. Singer et al. further demonstrated that the strength of this influence depends critically on the backbone size: the influence is strong for small backbones, but weak for large backbones. To study the influence of these two factors on local search cost in the general JSP, we enumerated the solutions at the optimal makespan for all 100 instances in each of our problem classes. Both the mean number of solutions and the mean local search costs are reported in Table 1. We see both a dramatic drop in the number of solutions and a gradual increase in local search cost as the backbone size is increased. Further, the local search cost is only slightly larger for the square 6×6 problems (although the exponential growth in search cost is becoming apparent at large backbones). This observation is consistent with the increase in search space size, but inconsistent with the observation that square problems are generally more difficult than rectangular problems.

The bottom third of Table 1 shows the log_{10}-log_{10} correlation between the number of solutions and the local search cost. The computed r-values demonstrate that both the number of solutions and the backbone size influence local search cost in the general JSP. Correlation is strong for problems with small backbones, and drops rapidly as backbone size increases. Although more factors are required to explain local search cost variance in large-backboned general JSPs, the results in Table 1 demonstrate that the interaction effect between backbone size and the number of solutions is not unique to SAT.

The results in Table 1 are produced by controlling for backbone size, instead of controlling for the number of solutions, raising the question "Does the number of solutions predict the search cost, independently of the backbone size?". A linear regression of log_{10}(search cost) on the number of solutions for *all* instances of a given problem size yields extremely high

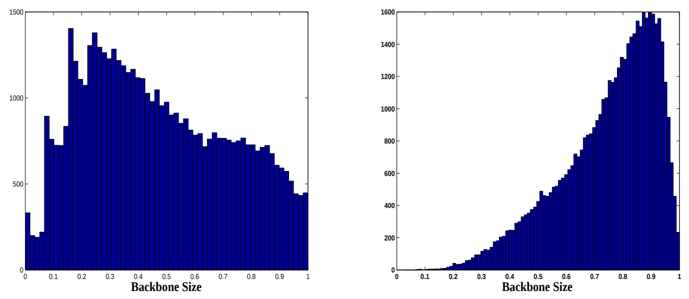

Figure 1: Histogram of backbone sizes for $50,000$ 6×4 (left figure) and 6×6 (right figure) general JSPs.

r-values, ranging from 0.82 to 0.91. While these values are better than *any* r-values observed while controlling for backbone size, the utility of the model is negligible as the search cost varies over *three* orders of magnitude for instances with backbone size ≈ 0.9 and a similar number of solutions. Similar observations led Singer et al. to control for backbone size, correcting the methodological deficiency of Clark et al.

4.4 Distribution of Backbone Sizes

While rectangular JSPs are believed to be significantly easier than square JSPs, this difference was not observed in the mean local search costs reported in Table 1. In a straightforward experiment, we generated 100 6×4 and 6×6 general JSPs, leaving the backbone size uncontrolled, and computed the mean local search cost for each group. The resulting mean costs were 32.91 and 498.13 for the 6×4 and 6×6 problems, respectively. This suggests a strong bias in the distribution of backbone sizes for the two types of problem.

In SAT, the relative distribution of backbone sizes depends on the ratio of the number of clauses c to the number of variables v [Parkes, 1997]. Under-constrained SAT instances (with small values of c/v) tend to have small backbones, while over-constrained SAT instances (with large values of c/v) tend to have large backbones. In the JSP, and many other optimization problems, there is no control parameter analogous to c/v by which we can control the expected degree of constrainedness. Instead, we can only control for problem size, asking the question "How common are various backbone sizes for JSPs of varying size?".

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
Distance-log_{10}(local search cost) correlation					
6 × 4	0.9890	0.9526	0.9070	0.8296	0.5303
6 × 6	0.9912	0.9327	0.8911	0.8371	0.6484
Backbone robustness-log_{10}(local search cost) correlation					
6 × 4	-0.2193	-0.3993	-0.4412	-0.5277	-0.5606
6 × 6	-0.1621	-0.3629	-0.4507	-0.4712	-0.5134

Table 2: Correlation (r) of 1) the mean distance to the nearest solution and 2) backbone robustness with log_{10}(local search cost) for general JSPs.

To answer this question, we generated $50,000$ problem instances for each problem size, and computed the backbone size for each instance. Histograms of the resulting backbone sizes for the general JSP instances are shown in Figure 1. The most common backbone sizes for the square 6×6 instances center near 0.9, and are exceedingly rare below 0.3. In contrast, the backbone sizes for the rectangular 6×4 instances are more uniformly distributed, with a slight bias toward smaller backbone sizes. We have also generated similar histograms for other small problem sizes; for ratios of $n/m > 1.5$, the bias toward small backbones becomes more pronounced, while for ratios < 1, the bias toward larger backbones is further magnified. Finally, we note that the utility of the correlation between number of solutions and local search cost depends heavily on problem size; the influence is negligible for nearly all 6×6 JSPs (which generally have large backbones), and for many 6×4 JSPs.

Finally, we note a discrepancy in the distribution of backbone sizes between the general JSP and the Euclidean Traveling Salesman Problem (TSP). In many TSP benchmark problems, there exists a single global optimum, causing the normalized backbone size to be 1. Data presented in [Walsh and Slaney, 2001] (Figures 3 and 4) further support the claim that backbones in the TSP tend to be maximal. However, as Figure 1 shows, maximal backbones are relatively rare in the general JSP. We believe this discrepancy is due to a fundamental difference in the objective functions of the TSP and JSP. In the TSP, each solution attribute (edge) contributes to solution fitness (tour length). However, in the JSP, only some solution attributes (operations) directly influence solution fitness (makespan): namely, the operations on the critical path. Depending on the constrainedness of the problem, there can be flexibility in the ordering of non-critical operations in the JSP, leading to smaller backbone sizes. In the TSP, there is no such flexibility, and multiple global optima can only be produced if alternative edges with *identical* inter-city distances are available; a relatively rare event in both real-world and randomly generated Euclidean TSPs.

4.5 Distance to Global Optima and Search Cost

Singer et al. hypothesized that the size of the quasi-solution sub-space largely dictates the cost of local search in SAT, and supported this conjecture by demonstrating strong correlations between local search cost and the mean Hamming distance between the first quasi-solutions encountered by a local search algorithm and the nearest optimal solution.

In applying this methodology to the general JSP, we encounter an immediate problem: what is a 'quasi-solution' in the JSP? Local search in the JSP is qualitatively different from that of SAT, instead progressing via alternating series of descent into and escape from deep local minima. Thus, local minima in the JSP are naturally analogous to quasi-solutions in SAT. Further, we hypothesize that the extent of the local minima influences local search cost; if the first local minimum encountered by a local search algorithm is distant from a global optimum, search will be expensive.

For each problem instance in each of our problem classes, we generated 1000 local optima, computed the Hamming distance to the *nearest* global optimum, and recorded the mean of the 1000 distances. The Hamming distance between two solu-

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	Mean Number of Optimal Solutions				
6×4wf	27369 ± 71049	81255 ± 295593	2515.25 ± 4704	293 ± 425	18 ± 16
6×6wf	1147650 ± 6555440	429102 ± 1676350	19017 ± 44556	4553 ± 6898	80 ± 94
	Mean Local Search Cost				
6×4wf	119.44 ± 89.32	122.2 ± 114.26	333.42 ± 442.39	920.72 ± 1515.75	2087.44 ± 2973.86
6×6wf	318.77 ± 113.82	513.13 ± 143.72	1086.33 ± 1979.39	1730.53 ± 2846.15	5036.53 ± 5132.54
	log_{10}-log_{10} correlation (r) between the number of solutions and local search cost				
6×4wf	-0.7650	-0.6663	-0.3484	-0.2613	-0.2208
6×6wf	-0.7345	-0.6877	-0.4316	-0.2700	-0.2561

Table 3: The mean number of solutions, mean local search cost, and log_{10}-log_{10} correlation (r) between the number of solutions and local search cost for JSPs with workflow. $X \pm Y$ denotes a mean of X with a standard deviation of Y.

tions in the JSP is taken as the number of order variables, out of the $mn(n-1)/2$ possible, whose truth values are different. Our local optima were generated by applying a next-descent algorithm from random active solutions. Our next-descent algorithm evaluates $N1$ neighbors in a random order, selecting the first improvement in makespan; the algorithm terminates when no such improvements are possible.

In Table 2, we report the correlation between the mean Hamming distance to the nearest global optimum and log_{10}(local search cost). For backbone sizes of ≈ 0.1 through ≈ 0.5, the correlation is extremely high, and only moderately degrades at the larger backbone sizes. The r-values are uniformly and significantly better than those achieved using the number of solutions, and further account for much of the variance in local search cost for large-backboned JSPs.

4.6 Backbone Robustness and Search Cost

Singer et al. propose backbone robustness as the primary factor determining the extent of the quasi-solution sub-space. Abstractly, backbone robustness represents the number of problem constraints that must be relaxed to produce a problem with a significantly smaller backbone. While in the JSP there is no analog to relaxing individual constraints (as is possible in SAT), there is a parameter controlling the global constrainedness: deviation from the optimal makespan. Thus, we define backbone robustness for the JSP as the minimum percentage above the optimal makespan at which the backbone size is reduced by at least half (subject to integral makespan constraints).

In the lower half of Table 2 we report the correlation between the backbone robustness and log_{10}(local search cost) for our general JSPs. The results are very similar to that reported by Singer et al. for SAT; a moderate negative correlation for large-backboned instances, and a gradual decay as backbone size is decreased. And as with SAT, backbone robustness does influence the extent of the quasi-solution subspace in the general JSP. Because our research deals with the extension of existing SAT cost models to the JSP, we did not establish the cause of the lesser influence of backbone robustness at smaller backbone sizes.

5 Extending the Analysis to Workflow JSPs

In both SAT and the general JSP, constraints are generated in a uniform, random fashion. Yet, many real-world scheduling problems have non-random constraints, and it is unclear to what degree the factors present in the SAT cost models are affected by such constraints. To study the effect of non-random constraints, we performed the same series of experiments detailed in the previous section on JSPs with workflow–which impose a specific structure on the machine processing orders for each job.

First, we considered the influence of the number of solutions on the local search cost of JSPs with workflow, as shown in Table 3. As with general JSPs, we see both a dramatic drop in the number of solutions and a gradual increase in the search cost as backbone size is increased. Workflow JSPs have significantly fewer solutions than general JSPs, and the search cost is generally an order of magnitude higher. However, the log_{10}-log_{10} correlation between the number of solutions and the local search cost is nearly identical with the results for general JSPs: correlation is strong for small-backboned problems, but decays as backbone size is increased.

Next, we generated histograms of the backbone size distributions for both the 6×4 and 6×6 workflow JSPs; the results are shown in Figure 2. In comparison with the distributions of general JSPs (Figure 1), it is clear that the presence of workflow dramatically shifts the distribution mass to larger backbones. For the rectangular 6×4 problems, workflow changes a bias toward small backbones into a relatively large bias toward large backbones, and for the 6×6 problems, workflow magnifies the already large bias toward large backbones. We note that the rarity of small-backboned workflow JSPs further diminishes the utility of the number of solutions as a predictor of local search cost for these instances.

Following the methodology of the previous section, we measured the correlation between the mean Hamming distance to the nearest global optimum and log_{10}(local search cost); the results are shown in the upper portion of Table 4. Here, we see a dramatic difference between general JSPs and workflow JSPs: while the influence of the Hamming distance to the nearest solution (which indirectly measures the size of the quasi-solution sub-space) at small backbones is relatively large, it drops very rapidly, ultimately vanishing at ≈ 0.9. Additionally, because of the lesser influence of Hamming distance to the nearest solution on local search cost, we see a corresponding drop in the influence of backbone robustness, as shown in the bottom half of Table 4. Regardless of backbone size, the moderate correlation with search cost observed for general JSPs does not exist for JSPs with workflow.

6 Discussion

Realistic scheduling problems contain non-random constraints. Yet, it is unclear whether the same factors that influence local search cost in random problems extend to those with structured constraints. When applying the analysis we performed on general JSPs to JSPs with workflow constraints, we observed that the presence of structured constraints strongly biased the problem space toward large-backboned instances. While the number of solutions remained a strong predictor for the relatively rare small-backboned instances, the correlation between the Hamming distance to the nearest solution and local search cost was negligible for the common large-backboned instances. Thus, the source of the majority of variance in local search cost for JSPs with workflow remains elusive.

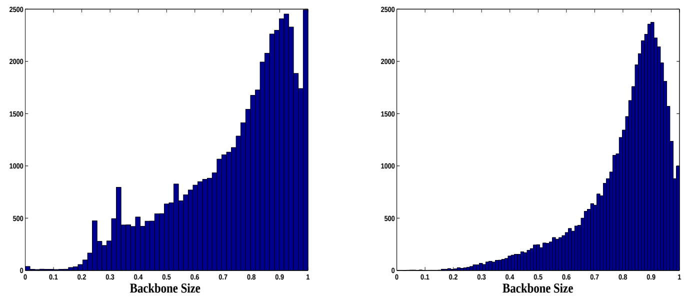

Figure 2: Histogram of backbone sizes for $50,000$ 6×4 (left figure) and 6×6 (right figure) JSPs with workflow.

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
Distance-log_{10}(local search cost) correlation					
6×4wf	0.8727	0.7122	0.5109	0.1811	0.0862
6×6wf	0.8231	0.6781	0.5264	0.1367	0.0711
Backbone robustness-log_{10}(local search cost) correlation					
6×4wf	-0.0029	-0.0217	-0.0372	-0.0752	-0.1423
6×6wf	-0.0165	-0.0348	-0.0513	-0.0941	-0.1239

Table 4: Correlation (r) of 1) the mean distance to the nearest global optimum and 2) backbone robustness with log_{10}(local search cost) for JSPs with workflow.

Two, perhaps obvious, conjectures arise from the fact that the factors influencing local search cost in general JSPs fail to carry over to JSPs with workflow. The first is that there are problem-independent factors influencing local search cost that have not yet been identified. These factors may play a much stronger role in structured problems than in more general, random problems. For both the general JSP and SAT, the correlation between Hamming distance to the nearest solution and local search cost was not overwhelmingly high, leaving room for other factors.

The second conjecture is that structured constraints introduce factors that do not influence search cost in random problems. If this is the case, work on descriptive models of search cost in random problems will not be relevant to non-random subclasses. This is especially critical in areas such as scheduling, where problems found in real-world applications often have a significant non-random component.

7 Conclusions

Our results clearly demonstrate that the factors influencing local search cost in SAT also influence local search cost in the general JSP, despite significant differences in search space topology and local search algorithms between the two problems. These factors include backbone size, number of solutions, Hamming distance to the nearest solution, and an analog of backbone robustness. Together, these factors provide a reasonably accurate local search cost model for the general JSP, although there is still non-negligible unexplained search cost variance for relatively dominant large-backboned instances. Our results also suggest these cost factors may be applicable to a wide range of optimization problems.

We also shed more light on the observation that rectangular JSPs are significantly easier than square JSPs. If we control for backbone size, rectangular JSPs are *not* significantly easier than square JSPs. Instead, the observed difference in difficulty stems primarily from the relative distributions of backbone sizes in the two problems: large backbones are dominant in square problems, while we see a bias toward smaller backbones in rectangular problems.

Finally, we also demonstrated that the same factors influencing local search cost in the general JSP do not necessarily transfer to JSPs with workflow, suggesting that the factors influencing local search cost in structured and random problems may be different.

8 Acknowledgments

The authors from Colorado State University were sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-00-1-0144. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[Balas, 1965] E. Balas. Machine scheduling via disjunctive graphs. *Operations Research*, 13:517–546, 1965.

[Beasley, 1990] J.E. Beasley. OR-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[Clark *et al.*, 1996] D.A. Clark, J. Frank, I.P. Gent, E. MacIntyre, N. Tomov, and T. Walsh. Local search and the number of solutions. In *Proceedings of CP-96*, 1996.

[Giffler and Thompson, 1960] B. Giffler and G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8(4):487–503, 1960.

[Hoos, 1998] H. H. Hoos. *Stochastic Local Search = Methods, Models, Applications*. PhD thesis, Darmstadt University of Technology, 1998.

[Jain and Meeran, 1999] A. S. Jain and S. Meeran. Deterministic job-shop scheduling: Past, present, and future. *European Journal of Operational Research*, 113(2), 1999.

[Laarhoven *et al.*, 1992] P.J.M Van Laarhoven, E.H.L. Aarts, and J.K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40:113–125, 1992.

[Mattfeld *et al.*, 1999] D. C. Mattfeld, C. Bierwirth, and H. Kopfer. A search space analysis of the job shop scheduling problem. *Annals of Operations Research*, 86:441–453, 1999.

[Nowicki and Smutnicki, 1996] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[Parkes, 1997] A. Parkes. Clustering at the phase transition. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

[Singer *et al.*, 2000] J. Singer, I.P. Gent, and A. Smaill. Backbone fragility and the local search cost peak. *Journal of Artificial Intelligence Research*, 12:235–270, 2000.

[Taillard, 1993] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operations Research*, 64:278–285, 1993.

[Walsh and Slaney, 2001] Toby Walsh and John Slaney. Backbones in optimization and approximation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[Watson *et al.*, 1999] J.P. Watson, L. Barbulescu, A.E. Howe, and L.D. Whitley. Algorithm performance and problem structure for flow-shop scheduling. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.