Toward an Understanding of Local Search Cost in Job-Shop Scheduling

Jean-Paul Watson[1], J. Christopher Beck[2],
Adele E. Howe[1], and L. Darrell Whitley[1]

[1] Computer Science Department
Colorado State University
Fort Collins, CO 80538 USA
{watsonj,howe,whitley}@cs.colostate.edu
[2] ILOG, S.A.
9, rue de Verdun, B.P. 85
F-94523 Gentilly Cedex France
cbeck@ilog.fr

Keywords: Scheduling, Search Space Analysis, Problem Difficulty*

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 SAT. We show that several factors known to influence the difficulty of local search in 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.

*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. J. Christopher Beck would also like to thank Paul Shaw (ILOG, S.A.) for discussions relating to this work.

1 Introduction

The job-shop scheduling problem (JSP) has a long-standing reputation as one of the most difficult NP-complete problems; even today, the optimal solutions for many small benchmark problems are still unknown [Bea90]. Significant progress was made in the 1990's, when researchers introduced a number of algorithms capable of producing optimal solutions to many notoriously difficult JSP benchmark problems in only minutes of CPU time; for example, the Tabu Search algorithm of Nowicki and Smutnicki (1996) is one of the best-performing algorithms for the JSP [BDP96] [VAL96]. Yet despite these advances, very little research has been devoted to understanding *why* these various algorithms work.

In this paper, we study descriptive cost models of local search for the JSP. Descriptive cost models are statistical models which relate search space features to search cost; better models account for more of the variance in search cost across different problem instances. We specifically examine the cost of local search in the JSP, considering a Tabu Search algorithm introduced by Taillard (1994). Taillard's algorithm is closely related to the Nowicki and Smutnicki's Tabu Search algorithm, but is significantly more amenable to analysis.

In contrast to the JSP, researchers have expended significant effort in recent years to produce relatively accurate descriptive cost models of local search for SAT [SGS00]. Intuitively, we would expect some factors present in the SAT cost model, 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. On the other hand, both the search space and constraint structure of the JSP differ in many important ways from SAT.

Our research investigates whether or not the SAT cost models can be leveraged in an effort to understand local search cost in the JSP. We demonstrate that the factors influencing local search cost in SAT *can* also influence local search cost in the JSP. These factors include the number of solutions [CFG$^+$96], backbone size [Par97], the distance to the nearest solution, and an analog of backbone robustness [SGS00]. Together, these factors provide a relatively accurate model of local search cost in the *general* JSP.

Concurrently, we also ask the question "Are there problem-independent factors influencing local search cost?" The SAT cost models describe local search cost for problems where the constraints (clauses) are randomly generated, with no expected structure. In contrast, real-world problem constraints are often structured. We apply the same analysis to JSPs with workflow, which possess simple, structured constraints. For these problems, we find that while the number of solutions and backbone size continue to influence search cost, the distance to the nearest solution and backbone robustness is very weakly correlated with search cost. We conclude by discussing the implications of our analysis for the JSP, SAT, and local search in general.

2 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 an arbitrary, pre-specified duration. Each machine can process only one job at a time, and once initiated, processing cannot be interrupted. Any job can start at time 0, and the objective is to minimize the *makespan*, or the maximum completion time of all jobs. In the *general* JSP, the machine processing orders for each job are independently sampled from a uniform distribution. In some forms of the JSP, the machine processing orders are more structured. In *workflow* JSPs, machines are typically divided into two equal-sized partitions containing machines 1 through $m/2$ and $m/2+1$ through m, respectively, and every job must be processed on all machines in the first partition before any machine in the second partition. Within each partition, the machine processing orders are sampled from a uniform distribution. The JSP with the makespan minimization objective is known to be NP-complete for $m \geq 2$ [GJS76]. Furthermore, the JSP is one of the most difficult NP-complete problems encountered in practice; the optimal solutions for many small (e.g., 15×15 and 20×20) benchmark problems are still unknown [Bea90], although algorithms for solving the JSP have been intensely studied for over 30 years.

No research has produced descriptive cost models for the JSP. However, some general qualitative observations regarding problem difficulty have emerged. First, independent of local search algorithm, we have the following trends:

1. For both general and workflow JSPs, "square" ($n/m \approx 1$) problem instances are significantly harder than "rectangular" ($n/m \gg 1$) problem instances.

2. Given fixed n and m, workflow JSPs are substantially more difficult than general JSPs.

Second, given either general or workflow JSPs with a fixed n and m, the relative difficulty of problem instances appears to be algorithm-independent. For example, easy/hard 20×10 general JSPs are generally easy/hard for all local search algorithms, from Tabu Search to Simulated Annealing. Clearly, any descriptive cost model for the JSP must be consistent with these observations.

Only one study reports on a quantitative analysis of problem difficulty in the JSP. Mattfeld et al. (1999) identify differences between the search spaces of some well-known 50×10 general and workflow JSPs. For example, they demonstrate that the extension of the search space (as measured by the average distance between random local optima) is larger in JSPs with workflow, suggesting a cause for the large search cost (relative to general JSPs) associated with these problem instances. Two other measures, entropy and correlation length, were also used to establish quantitative differences in the search spaces of these 50×10 general and workflow JSPs.

While the measures studied by Mattfeld et al. do demonstrate differences in the search spaces of general and workflow JSPs, they do not constitute a descriptive cost model. Furthermore, Mattfeld et al. did not investigate whether these same factors were responsible for the relative difficulty of square versus rectangular JSPs, or for the variance in local search cost for different

problem instances of the same size and workflow configuration. In preliminary experiments, we attempted to develop descriptive cost models using these measures, and found that while these factors *did* influence search cost, the influence was much weaker than for the factors we discuss in Section 5.

3 Descriptive Cost Models in other NP-Complete Problems

Many techniques for characterizing and analyzing problem difficulty have found application in a wide variety of NP-complete problems; for example, phase transitions and the associated peak in search cost [HHW96] have been documented for a number of problems. Given such universals, it is therefore important to examine, and if possible leverage, existing descriptive cost models for other NP-complete problems. However, a detailed review of the literature on local search yields descriptive cost models for only two related problems: SAT and the general CSP. Outside of these two example, the dominant methods for quantifying problem difficulty are unable to account for the large cost variance found in different problem instances of a given size. For example, correlation length [RS01] is strictly a function of problem size (e.g., the number of cities in the TSP).

Clark et al. (1996) introduced the first descriptive cost model for both SAT and more general CSPs, hypothesizing that the number of solutions was the primary factor influencing local search cost. Intuitively, a decrease in the number of solutions should yield an increase in local search cost. Clark et al. demonstrated a relatively strong (negative) log-log correlation between the number of solutions and local search cost, with r-values ranging anywhere from -0.77 to -0.91. However, the model failed to account for the large cost variance in problems with small numbers of solutions, where model residuals varied over three or more orders of magnitude.

The descriptive cost model introduced by Singer et al. (2000) largely corrected this deficiency and went further by proposing a causal model for local search cost in SAT. The problem backbone is a key concept underlying the Singer et al. cost model. The *backbone* of a SAT instance, first introduced by Parkes (1997), consists of the subset of literals that have the same truth value in *all* solutions. Singer et al. demonstrated that the backbone size is an important factor in the SAT cost model, showing that when the backbone is small, there is a strong (negative) log-log correlation ($r \approx -0.77$) between the number of solutions and the local search cost. However, this correlation nearly vanishes ($r \approx -0.12$) when the backbone is large [SGS00].

To account for the inability of the number of solutions to predict search cost in problems with large backbones, it is important to note that the number of solutions to a problem instance is not indicative of search space topology, although this topology clearly influences the overall search cost. For example, local search algorithms for SAT quickly locate sub-optimal *quasi-solutions*, which contain relatively few unsatisfied clauses. These quasi-solutions form a sub-space which contains all solutions, and is largely interconnected; once a point in this sub-space is identified, local search algorithms for SAT typically restrict search to this sub-space. This observation led Singer et al. (2000) 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 (the number of differing variable assignments) between the first quasi-solution encountered by a local search algorithm and the nearest solution, 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, and degraded only slightly for problems with larger backbones ($r \approx 0.75$); no explanation for the degradation was provided. Thus, with strong correlations at all backbone sizes, Singer et al. hypothesize that the distance to the nearest solution, and not the number of solutions, is the primary factor influencing local search cost in SAT.

While the distance from a quasi-solution to the nearest solution does estimate the size of the quasi-solution sub-space, it does not provide any intuition about the problem features that *cause* the sub-space size to vary among different 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 just 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" ([SGS00], 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.

To confirm their hypothesis, Singer et al. demonstrated a moderate (≈ -0.5) negative correlation between backbone robustness and the log of local search cost for large-backboned SAT instances. Surprisingly, this correlation degraded as the backbone size was decreased, leading Singer et al. to hypothesize that "finding the backbone is less of an issue and so backbone fragility, which hinders this, has less of an effect" ([SGS00], p. 254), although this conjecture was not explicitly tested.

4 Algorithms, Test Problems, and Methodology

Descriptive cost models are tied to specific algorithms, although they may be generalized to other algorithms, depending on both the factors present in the cost model and the similarity of the algorithms involved. For example, the SAT cost model proposed by Singer et al. is based on a particular tabu-like local search algorithm, Walk-SAT. Yet, it is reasonable to hypothesize that the size of the quasi-solution sub-space should influence the performance of similar tabu-like algorithms for SAT (such as GSAT), although we would expect the parameters of the model to be slightly different. We now introduce the Tabu Search algorithm used to develop our descriptive cost model for the JSP. This algorithm is closely related to the state-of-the-art Tabu Search algorithm of Nowicki and Smutnicki (1996), but is far more amenable to analysis.

4.1 Algorithm Description

The Tabu Search [GL97] algorithm we consider in our analysis was introduced by Taillard (1994). This was one of the first Tabu Search algorithms for the JSP and is the basis for more advanced, state-of-the-art Tabu Search algorithms such as that of Nowicki and Smutnicki (1996). Taillard's algorithm uses the move operator introduced by Van Laarhoven et al. (1992) , which is often denoted by $N1$. The $N1$ neighborhood is generated by swapping all adjacent pairs of jobs on a randomly selected critical path in the current solution. As in most Tabu Search algorithms for the JSP, the tabu list contains recently swapped pairs of jobs, and the list length is dynamically updated to avoid cycling behavior. The only long-term memory mechanism is a simple aspiration criterion, which over-rides the 'tabu' status of any move that results in a solution that is better than any encountered in the current run; as Taillard indicates, frequency-based long-term memory is only necessary "for problems that need a great number of iterations" ([EDT94], p. 110). All runs are initiated from a randomly generated "active" solution [GT60]. In each *iteration* of Taillard's algorithm, all $N1$ neighbors are generated, the best non-tabu move is taken, and the tabu list is updated accordingly.

The cost required to solve a given problem instance using Taillard's algorithm is naturally defined as the number of iterations required to locate a global optima. However, the number of iterations is stochastic (with an approximately exponential distribution [EDT94]), due to both the randomly generated initial solution and random tie-breaking when more than one 'best' move is available. Consequently, we define the search cost for a particular JSP instance as the median number of iterations required to locate an optimal solution over 1000 independent runs; with 1000 samples, the estimate of the distribution median is relatively stable [Hoo98] [SGS00].

For analysis purposes, the most important feature of Taillard's algorithm is the $N1$ move operator. More advanced critical path move operators for the JSP, such as that used in Nowicki and Smutnicki's algorithm, induce search spaces that are *disconnected*, such that it is not always possible to move between a randomly generated solution and a global optimum. Consequently, any algorithm using such a move operator is not Probabilistically Approximately Complete (PAC) [Hoo98]: even with infinite run-time, the algorithm is not guaranteed to locate a global optimum. This severely complicates algorithm analysis, as it is unclear how to define the search cost associated with a given problem instance. However, use of a connected move operator does *not* automatically guarantee that an algorithm is PAC [Hoo98]. While we have no analytic proof that Taillard's algorithm is PAC, the empirical evidence is compelling, having yet to encounter even a single instance for which the algorithm is incapable of locating a global optimum.

4.2 Defining a Backbone for JSP

We also found that backbone size was an important factor in a descriptive cost model for the JSP. The definition of a backbone in the JSP, or any problem, depends on how solutions are represented. In the JSP, the disjunctive graph is a widely used method for encoding solutions

[Bal65]. A solution is encoded using $n(n-1)/2$ 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.

4.3 Test Problems

For a variety of reasons, we are restricted to relatively small problem sizes in our experiments. From a technical standpoint, the factors present in our descriptive cost model, such as backbone size or distance to the nearest optimal solution, are functions of *all* optimal solutions to a problem instance. Generating all optimal solutions to a problem instance is much more expensive than merely proving optimality (the enumeration is explicit, in contrast to the implicit approach characteristic of branch-and-bound algorithms). Further, the number of optimal solutions in even small problem instances is measured in the millions, which can easily exceed available memory in modern workstations. From a pragmatic standpoint, we consider a wide range of problems in our experiments, controlling for backbone size, workflow configuration, and problem size. We study over 4000 problem instances, computing the median search cost over 1000 independent runs for each. Even for the problem sizes we consider, the overall CPU time invested was approximately 8 CPU months on 750 MHz Pentium III workstations.

In our experiments, we examine 6×4 and 6×6 problems, both with and without workflow partitions. Operation durations are sampled uniformly from the interval $[1, 99]$, following [EDT93]. Backbone size is an integral factor in our descriptive cost model. Unfortunately, even for these problem sizes, it is infeasible to control for a *specific* backbone size: computation of the backbone is considerably more expensive in the JSP 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 of problems by $\approx X$. For each problem size, we generated 100 general and workflow JSP instances at each of the following backbone sizes: ≈ 0.1, ≈ 0.3, ≈ 0.5, ≈ 0.7, and ≈ 0.9. Finally, we used a constraint-directed scheduling algorithm to compute the optimal makespan, the backbone size, and to enumerate all optimal solutions. The specific algorithm is documented in [BF00]; features include the min-slack variable and value ordering heuristics [CS97] and constraint propagators based on edge-finding [Nui94].

5 A Descriptive Cost Model for the General JSP

It is unclear, a-priori, whether the factors in the SAT cost model are relevant to local search in the JSP. In SAT, plateaus of quasi-solutions with equal numbers of unsatisfied clauses dominant the search space,and there is little gradient information to guide search. In contrast, local optima with deep attractor basins are the dominant feature in the search space of the JSP, and local search algorithms spend the majority of their time escaping local optima. In this section we

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	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
	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 Optimal 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 number of optimal solutions, local search cost, and log_{10}-log_{10} correlation (r) between the number of optimal solutions and local search cost for general JSPs. $X \pm Y$ denotes a mean of X with a std. dev. of Y.

demonstrate that, despite such differences in search space topologies, the factors present in the descriptive cost model for SAT *do* influence local search cost in the JSP.

5.1 Number of Solutions and Search Cost

Clark et al. first showed that the number of solutions influences the cost of local search in both SAT and more general CSPs. Singer et al. further demonstrated that the strength of this influence depends critically on the backbone size: the influence is strong in problems with small backbones, but weak in problems with large backbones. In Table 1, we show summary statistics for the number of optimal solutions and the local search cost for our general JSPs. 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 the backbone size ≈ 0.9). This observation is consistent with the increase in search space size for the larger problems, but inconsistent with the observation that square problems are generally much more difficult than rectangular problems.

The bottom third of Table 1 shows the log_{10}-log_{10} correlation between the number of optimal solutions and the local search cost. The computed r-values demonstrate that both the number of optimal solutions and the backbone size influence local search cost in the general JSP. Correlation is relatively strong for problems with small backbones, and drops rapidly as backbone size increases. Although more factors are required to fully explain the variance in local search cost for 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.

5.2 Distribution of Backbone Sizes

While rectangular JSPs tend to be much easier than square JSPs, this difference was not observed in the local search costs reported in Table 1. In a straightforward experiment, we generated 100 6×4 and 6×6 general JSPs and computed the local search cost for each group, leaving the backbone size *uncontrolled*. The resulting mean search costs were 32.91 and 498.13

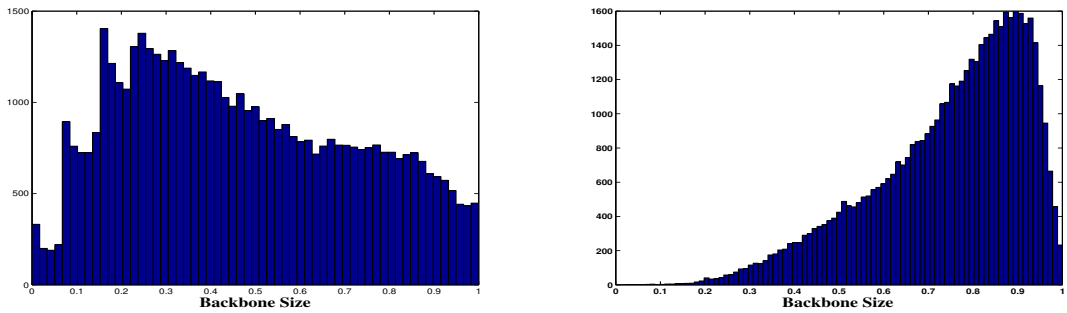

Figure 1: Histogram of backbone sizes for $50,000$ 6×4 (left figure) and 6×6 (right figure) general JSPs.

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 distribution of backbone sizes depends on the ratio of the number of clauses c to the number of variables v [Par97]. 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; the frequency of large-backboned problems increases rapidly in the so-called 'critically constrained' region. In the JSP and many other optimization problems, there is no parameter analogous to c/v which controls the expected degree of constrainedness. Thus, we can only observe the relative frequency of backbone sizes in these problems.

To examine the relative frequency of backbone sizes in the general JSP, we generated $50,000$ problem instances for each problem size and computed the backbone for each instance. Histograms of the resulting backbone sizes 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.

5.3 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. They 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? Tabu Search, like all local search algorithms, is heavily biased toward local optima. Con-

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 optimal solution and 2) backbone robustness with log_{10}(local search cost) for general JSPs.

sequently, we consider local minima in the JSP to be analogous to quasi-solutions in SAT; it is not unreasonable to hypothesize that if random local minima are very distant from a global optimum, search is likely to be quite expensive.

For our general JSP instances, 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 solutions in the JSP is the number of order variables, out of the $mn(n-1)/2$ possible, with different assigned values). 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 solution which improves on the makespan of the current solution; 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 account for much of the cost variance in problems with large backbones. Thus, we also found that the distance to the nearest optimal solution, and not the number of solutions, is the primary factor influencing the cost of local search in the JSP.

5.4 Backbone Robustness and Search Cost

Singer et al. proposed backbone robustness as the primary causal factor determining the size 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 the constraint that makespans must be integral).

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

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	Number of Optimal Solutions				
6×4wf	27369 $\pm$ 71049	81255 $\pm$ 295593	2515.25 $\pm$ 4704	293 $\pm$ 425	18 $\pm$ 16
6×6wf	1147650 $\pm$ 6555440	429102 $\pm$ 1676350	19017 $\pm$ 44556	4553 $\pm$ 6898	80 $\pm$ 94
	Local Search Cost				
6×4wf	119.44 $\pm$ 89.32	122.2 $\pm$ 114.26	333.42 $\pm$ 442.39	920.72 $\pm$ 1515.75	2087.44 $\pm$ 2973.86
6×6wf	318.77 $\pm$ 113.82	513.13 $\pm$ 143.72	1086.33 $\pm$ 1979.39	1730.53 $\pm$ 2846.15	5036.53 $\pm$ 5132.54
	log_{10}-log_{10} Correlation (r) Between the Number of Optimal 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 number of solutions,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 std. dev. of Y.

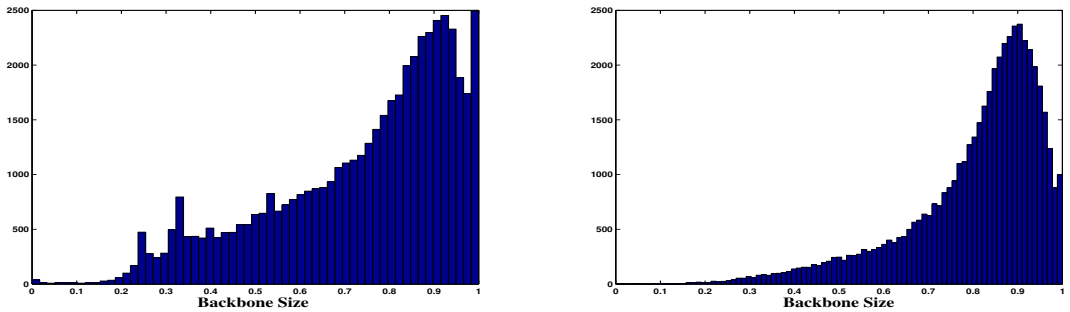

Figure 2: Histogram of backbone sizes for $50,000$ 6×4 (left figure) and 6×6 (right figure) JSPs with workflow.

gradual decay as backbone size is decreased. As with SAT, backbone robustness does appear to influence the size of the sub-space containing local optima in the general JSP. As we noted in Section 3, Singer et al. provide a justification for the smaller influence of backbone robustness for small-backboned instances.

6 Extending the Analysis to JSPs with Workflow

The machine processing orders for each job in the JSP are constraints, as are the individual clauses in SAT. In both the general JSP and Random 3-SAT (the particular SAT instances used

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.

to develop the Singer et al. cost model), these constraints are generated in a uniform, random fashion. An important issue is then generalization: real-world problems have non-random constraints, and it is unclear whether the factors present in the SAT and general JSP cost models are applicable in problems with more structured constraints. To study the effect of non-random constraints, we extend the analysis of Section 5 to JSPs with workflow–which impose a simple, specific structure on the machine processing orders for each job.

First, we consider 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.

Finally, 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.

7 Discussion and Implications

Our results show that the distance to the nearest global optimum is a good predictor of search cost in both the general JSP and SAT, despite strong differences in the underlying search space. In both cases, this distance indirectly measures the size of the search space explored by the respective local search algorithms. Modern local search algorithms for SAT, such as Walk-SAT, perform a random walk over the quasi-solution sub-space; consequently, it is unsurprising that search cost is an exponential function of the sub-space size. However, this inference also applies to Taillard's Tabu Search algorithm for the general JSP: it is effectively performing a random walk in the space of local optima. The ability of distance to the nearest global optimum to pre-

dict search cost also indicates that there is no, or at most a very weak, bias in the search spaces of both problems; if there were, distance alone would fail to accurately predict the search cost.

In the workflow JSP, we found that the distance to the nearest optimal solution was a very poor predictor of search cost. Follow-up experiments indicate that there is a very strong bias toward sub-optimal solutions in many of these problems: there are many more 'paths' in the search space to sub-optimal solutions than to optimal solutions. However, we as yet have no method to quantify this bias. The results for the workflow JSPs are important, because in contrast to Random 3-SAT, these problems have weakly structured constraints. Our experiments demonstrate, at least for the JSP, that the factors influencing search cost in random and structured problems are different. In addition to questioning the utility of descriptive cost models for purely random problems, this has immediate implications for algorithm design. For example, local search algorithms with no long-term memory, which include Genetic Algorithms, Simulated Annealing, and many forms of Tabu Search, are likely to perform poorly on these problems; without such memory, these algorithms mirror any biases in the underlying search space.

We view this research as a first step toward understanding why local search algorithms for the JSP are so effective. We selected Taillard's Tabu Search algorithm precisely because it serves as a baseline for more advanced algorithms, such as that of Nowicki and Smutnicki, which enhance Taillard's algorithm through either more advanced move operators or long-term memory. With descriptive cost models for the basic algorithm, we can begin to *systematically* assess the influence of these improvements on the descriptive cost model.

8 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 the underlying search spaces. These factors include backbone size, number of optimal solutions, distance to the nearest optimal 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 randomly generated 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.

References

[Bal65] Egon Balas. Machine scheduling via disjunctive graphs. *Operations Research*, 13:517–546, 1965.

[BDP96] Jacek Blażewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93:1–33, 1996.

[Bea90] J.E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[BF00] J. Christopher Beck and Mark S. Fox. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. *Artificial Intelligence*, 117(2):31–81, 2000.

[CFG+96] David A. Clark, Jeremy Frank, Ian P. Gent, Ewan MacIntyre, Neven Tomov, and Toby Walsh. Local search and the number of solutions. In *Proceedings of the Second International Conference on Principles and Practices of Constraint Programming (CP-96)*, pages 119–133, 1996.

[CS97] Cheng-Chung Cheng and Stephen F. Smith. Applying constraint satisfaction techniques for job shop scheduling. *Annals of Operations Research*, 70:327–378, 1997.

[EDT93] Éric D. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285, 1993.

[EDT94] Éric D. Taillard. Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, 6(2):108–117, 1994.

[GJS76] Michael R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[GL97] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[GT60] B. Giffler and G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8(4):487–503, 1960.

[HHW96] Tad Hogg, Bernardo A. Huberman, and Colin P. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81(1–2):1–15, 1996.

[Hoo98] Holger H. Hoos. *Stochastic Local Search - Methods, Models, Applications*. PhD thesis, Darmstadt University of Technology, 1998.

[LAL92] 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.

[NS96] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[Nui94] W.P.M. Nuijten. *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.

[Par97] Andrew J. Parkes. Clustering at the phase transition. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 340–345, 1997.

[RS01] Christian M. Riedys and Peter F. Stadler. Combinatorial landscapes. Technical Report 01-03-014, The Santa Fe Institute, 2001.

[SGS00] Josh Singer, Ian P. Gent, and Alan Smaill. Backbone fragility and the local search cost peak. *Journal of Artificial Intelligence Research*, 12:235–270, 2000.

[VAL96] R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra. Job shop scheduling by local search. *INFORMS Journal on Computing*, 8(3):302–317, 1996.