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] Colorado State University, Fort Collins, CO 80523-1873 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

Abstract. Local search algorithms are among the most effective approaches for solving the JSP, yet we have little understanding of which problem features influence search cost in these algorithms. We study a descriptive cost model of local search in the job-shop scheduling problem (JSP), borrowing from the MAX-SAT cost models. We show that several factors known to influence the difficulty of local search in MAX-SAT directly carry over to the general JSP, including the number of optimal solutions, backbone size, the distance between initial solutions and the nearest optimal solution, and an analog of backbone robustness. However, these same factors only weakly influence local search cost in JSPs with workflow, which possess structured constraints. While the factors for the MAX-SAT cost models provide an accurate description of local search cost in the general JSP, our results for workflow JSPs raise concerns regarding the applicability of cost models derived using random problems to those exhibiting specific structure.

1 Introduction

Local search algorithms, particularly those based on tabu search, are among the most effective approaches for solving the JSP [BDP96]. Yet, we have little understanding as to *why* these algorithms work so well, and under what conditions. In this paper, we study descriptive cost models of local search in the JSP. Descriptive cost models relate search space features to search cost; better models account for more of the variance in search cost across different problem instances. We examine the cost of tabu search in the JSP by considering an algorithm introduced by Taillard (1994), which is closely related to many state-of-the-art algorithms for the JSP (e.g., [NS96]), and is significantly more amenable to analysis.

Although no descriptive cost models for the JSP exist, researchers have expended significant effort in recent years to produce relatively accurate descriptive cost models of local search for MAX-SAT [SGS00]. Intuitively, we would expect some factors present in these models, such as the number of optimal solutions, to influence the difficulty of local search in other problems such as the JSP. At the same time, both the search space and constraint structure of the JSP differ in many important ways from MAX-SAT, making the a-priori applicability of these models unclear.

We investigate whether or not the descriptive cost models for MAX-SAT can be leveraged in an effort to understand local search cost in the JSP. We demonstrate that the factors present in the MAX-SAT cost models also influence local search cost in the JSP, including the number of solutions [CFG+96], backbone size [Par97], the distance between initial solutions and the nearest optimal solution [SGS00], and an analog of backbone robustness [SGS00]. Together, these factors form the basis of a relatively accurate descriptive model of local search cost in the *general* JSP.

The constraints in both MAX-SAT (clauses) and the general JSP (machine processing orders) are randomly generated, and in expectation are unstructured. In contrast, the constraints in real-world problems are often structured. We apply the same analysis to JSPs with workflow, a restricted form of the JSP with simple, structured constraints. We find that the factors present in the MAX-SAT and general JSP descriptive cost models only weakly influence search cost in workflow JSPs. We conclude by discussing the implications of our analysis for the JSP, MAX-SAT, and local search in general.

2 The JSP and Problem Difficulty

We consider the well-known $n \times m$ static 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 any job. In the *general* JSP, the machine processing orders are independently sampled from a uniform distribution. In the *workflow* JSP, 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.

While no descriptive cost models for the JSP exist, 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: e.g., a problem instance that is difficult for tabu search is likely to be difficult for simulated annealing. Clearly, any descriptive cost model for the JSP must be consistent with each of these observations.

Mattfeld et al. (1999) perform a quantitative analysis of problem difficulty in the JSP, identifying significant differences in the search spaces of some well-known 50×10 general and workflow JSPs. Specifically, they show that the extension of the search space (as measured by the average distance between random local optima) is larger in workflow JSPs, suggesting a cause for the generally larger search cost associated with these problem instances. Two other measures, entropy and correlation length, also demonstrated quantitative differences in the search spaces of these same problems.

While Mattfeld et al. do identify differences in the search spaces of general and workflow JSPs, it is unclear whether these differences account for the variance in local search cost for different problem instances of the same size and workflow configuration. In preliminary experiments, we found that while the factors introduced by Mattfeld et al. *did* influence search cost, the influence was much weaker than for the factors we discuss in Section 5. Furthermore, Mattfeld et al. did not investigate whether these same factors were responsible for the relative difficulty of square versus rectangular JSPs.

3 The MAX-SAT Descriptive Cost Model

The MAX-SAT descriptive cost model is the basis for our study. Many methods for characterizing problem difficulty have found application in a wide variety of problems, for example phase transitions and the associated peak in search cost. Given such universals, it is important to examine, and if possible leverage, any existing analysis on problem difficulty. However, the literature on local search yields descriptive cost models for only two, related problems: MAX-SAT and MAX-CSP. Outside of these two examples, 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).

Intuitively, a decrease in the number of optimal solutions should yield an increase in local search cost. This observation formed the basis of the first descriptive cost model for MAX-SAT, in which Clark et al. (1996) 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 very small numbers of optimal solutions, where model residuals varied over three or more orders of magnitude.

Singer et al. (2000) subsequently introduced a descriptive cost model that largely corrected the deficiency present in the Clark model, and went further by proposing a causal model for local search cost in MAX-SAT. The *backbone* of a problem instance is a key concept in Singer's descriptive cost model. The backbone of a MAX-SAT instance consists of the subset of literals that have the same truth value in *all* optimal solutions[Par97]. Singer demonstrated that the backbone size does influence search cost in MAX-SAT, showing that when the backbone is small, there is a strong (negative) log-log correlation ($r \approx -0.77$) between the number of optimal solutions and the local search cost. However, this correlation nearly vanishes ($r \approx -0.12$) when the backbone is large [SGS00].

Local search algorithms for MAX-SAT quickly locate sub-optimal *quasi-solutions*, that contain relatively few unsatisfied clauses. These quasi-solutions form a sub-space that contains all optimal solutions, and is largely interconnected; once a point in this sub-space is identified, local search algorithms for MAX-SAT typically restrict search to this sub-space. This observation led Singer to hypothesize that the size of this sub-space dictates the overall search cost, which could overcome the inability of the number of optimal solutions to predict local search cost in problems with large backbones.

To test this hypothesis, Singer measured the mean Hamming distance (the number of differing variable assignments) between the first quasi-solution encountered during

local search and the nearest optimal solution, which we denote $d_{init-opt}$, and computed the correlation between $d_{init-opt}$ 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 ($r \approx 0.75$). Consequently, $d_{init-opt}$, and not the number of optimal solutions, is the primary factor influencing local search cost in MAX-SAT.

Singer also posited a causal explanation for the variance in $d_{init-opt}$ across different MAX-SAT instances, which is based on the notion of *backbone robustness*. A MAX-SAT instance is said to have a *robust* backbone if a substantial number of clauses can be deleted before the backbone size is reduced by half. Conversely, an instance is said to have a *fragile* backbone if the deletion of just a few clauses reduces the backbone size by half. Singer argues that "backbone fragility approximately corresponds to how extensive the quasi-solution area is" ([SGS00], p. 251), by noting that a fragile backbone allows for large $d_{init-opt}$ because of the sudden drop in backbone size, while $d_{init-opt}$ is necessarily small in problem instances with robust backbones.

To confirm this hypothesis, Singer measured a moderate (≈ -0.5) negative correlation between backbone robustness and the log of local search cost for large-backboned MAX-SAT instances. Surprisingly, this correlation degraded as the backbone size was decreased, leading Singer 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

We now briefly describe Taillard's tabu search algorithm for the JSP, and introduce the test problems and methodology we use to investigate descriptive cost models for the JSP.

4.1 Algorithm Description

The tabu search algorithm we consider in our analysis was introduced by Taillard (1994). This was the first tabu search algorithm for the JSP and is the basis for more advanced, state-of-the-art JSP algorithms such as that of Nowicki and Smutnicki (1996). Taillard's algorithm uses the Van Laarhoven move operator [LAL92], which is often denoted by *N1*. The *N1* neighborhood is generated by swapping all adjacent pairs of jobs on any critical path in the current solution. As in most tabu search algorithms for the JSP, recently swapped pairs of jobs are prevented from being re-established for a particular duration, called the tabu tenure; the tabu tenure is dynamically updated to avoid cycling behavior. All runs are initiated from randomly generated "active" solutions [GT60]. In each *iteration* of Taillard's algorithm, all *N1* neighbors are generated, and the best non-tabu move is taken. 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 ([EDT94], p. 110), frequency-based long-term memory is only necessary for problems that require a very large (> 1M) number of iterations, which is not the case for the test problems introduced later in this section.

The cost required to solve a given problem instance using Taillard's algorithm is naturally defined as the number of iterations required to locate an optimal solution. 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 local search cost for a problem 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 somewhat 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, can induce search spaces that are *disconnected*, such that it is not always possible to move between a randomly generated solution and an optimal solution. 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 an optimal solution. This severely complicates algorithm analysis, as it is unclear how to define the search cost associated with a 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: in producing the results discussed in Sections 5 and 6, Taillard's algorithm never failed to locate an optimal solution.

4.2 Defining a Backbone for JSP

The definition of a backbone in any problem depends on how the solutions are represented. Taillard's algorithm encodes solutions using a *disjunctive graph*, which contains $n(n-1)/2$ Boolean "order" variables for each of the m machines, each of which represents a precedence relation between a distinct pair 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 optimal solutions. 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 (e.g., backbone size and the distance between initial and optimal solutions) 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 $[1, 99]$. 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 MAX-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 Beck and Fox (2000), which uses min-slack variable and value ordering heuristics and edge-finding constraint propagators.

5 A Descriptive Cost Model for the General JSP

A-priori, it is unclear whether the factors present in the MAX-SAT cost models are relevant to local search in the JSP. In MAX-SAT, the search space is dominated by plateaus of equally-fit quasi-solutions, and the main challenge for local search is to either find an exit from a plateau to an improving quasi-solution, or to escape the plateau by accepting a short sequence of dis-improving moves [FCS97]. In contrast, the JSP search space is dominated by local optima with variable-sized and variable-depth attractor basins. Consequently, local search algorithms for the JSP spend much of their time either escaping or avoiding local optima. In this section, we examine the MAX-SAT cost models in the context of Taillard's algorithm for the JSP, and demonstrate that despite qualitative differences in search space topologies, the MAX-SAT cost model factors *do* form the basis of an accurate descriptive model of local search in the JSP.

5.1 Number of Solutions and Search Cost

In MAX-SAT, the number of optimal solutions *can* influence local search cost, although the strength of this influence depends critically on backbone size: it is strong in problems with small backbones, and very weak in problems with large backbones. In Table 1, we report 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 optimal solutions and a gradual increase in local search cost as the backbone size is increased. Further, at a fixed backbone size the difference in local search cost between the 6×6 and 6×4 problems is minimal, and can be attributed to the larger size of the search space in the 6×6 problem instances.

In the bottom third of Table 1, we report the log_{10}-log_{10} correlation between the number of optimal solutions and local search cost. The r-values indicate that both the number of optimal solutions and the backbone size influence local search cost in the general JSP. As in MAX-SAT, the correlation is relatively strong for small-backboned problems, and drops rapidly with increases in backbone size. Although additional factors are required to fully account for the variance in local search cost for large-backboned general JSPs, these results demonstrate that the interaction effect between backbone size and the number of solutions is not unique to MAX-SAT.

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 # 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.

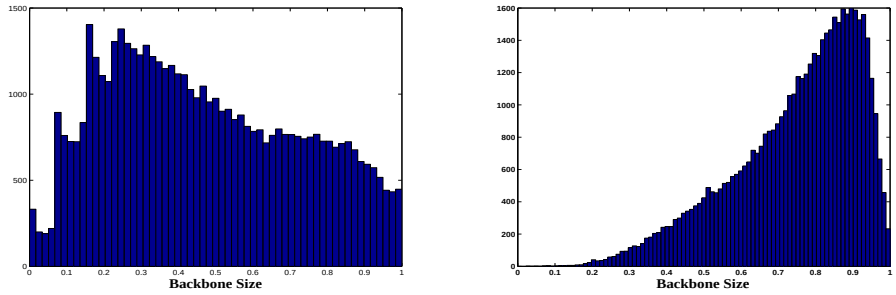

Fig. 1. Histogram of backbone sizes for 50 000 6×4 (left figure) and 6×6 (right figure) general JSPs.

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 problem set, leaving the backbone size *uncontrolled*. The mean local search costs were 32.91 and 498.13 for the 6×4 and 6×6 problem sets, respectively, suggesting a strong bias in the distribution of backbone sizes for the two problem types.

In MAX-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 problem (with small values of c/v) tend to have small backbones, while over-constrained problems (with large values of c/v) tend to have large backbones; the relative 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 known parameter analogous to c/v by which we can control for the expected degree of constrainedness. Consequently, 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 6×4 and 6×6 problems, and computed the backbone size for each instance. In Figure 1, we provide histograms illustrating the relative frequency of the backbone sizes. The most common backbone sizes for the square 6×6 instances are roughly 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

Problem	Backbone Size				
Size	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
$d_{init-opt}$-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) $d_{init-opt}$ and 2) backbone robustness with log_{10}(local search cost) in general JSPs.

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 optimal 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

In MAX-SAT, the mean distance between the initial quasi-solutions encountered by local search and the nearest optimal solution ($d_{init-opt}$) is strongly correlated with local search cost, across all backbone sizes. Intuitively, we would also expect the distance between the first local optima encountered by local search and the nearest optimal solution to influence local search cost in the JSP; the question is then "How strong is this influence?".

For each of our general JSPs, we generated 1000 local optima, computed the Hamming distance to the nearest optimal solution for each of the resulting optima, 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); as with MAX-SAT, we denote this measure by $d_{init-opt}$. We generated the local optima by applying a next-descent algorithm from random "active" solutions [GT60]. Our next-descent algorithm evaluates the neighbors of the current solution under the *N1* move operator in a random order, selecting the first solution that improves on the makespan of the current solution; the algorithm terminates when no such improvements are possible.

In Table 2, we report the correlations between $d_{init-opt}$ and log_{10}(local search cost). For backbone sizes of ≈ 0.1 through ≈ 0.5, the correlation is extremely high, and only moderately degrades for the two larger backbone sizes. The r-values are uniformly and significantly better than those achieved using the number of solutions, and account for a significant proportion of the variance in local search cost for large-backboned problems. Thus, we also conclude that the distance between initial and optimal solutions, and not the number of optimal solutions, is the primary factor influencing the cost of local search in the general JSP, independent of backbone size.

5.4 Backbone Robustness and Search Cost

Singer et al. propose backbone robustness a causal factor that largely determines the size of the quasi-solution sub-space in MAX-SAT. Abstractly, backbone robustness is a

measure of 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 MAX-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 those reported by Singer et al. for MAX-SAT; a moderate negative correlation for large-backboned instances, and a gradual decay as backbone size is decreased. Analogous to MAX-SAT, backbone robustness does appear to partially dictate the size of the subspace containing local optima in the general JSP. As we noted in Section 3, Singer et al. provide a justification for the lower correlations for small-backboned instances.

6 Extending the Analysis to Workflow JSPs

The primary problem constraints in the JSP are the machine processing orders for each job, while in MAX-SAT they are the individual clauses. In both cases, researchers typically generate problem instances such that these constraints are uniformly random. An important issue is then generalization: real-world problems have non-random constraints, and it is unclear whether the descriptive cost models for MAX-SAT and general JSP are applicable to problem instances with more structured constraints. To study the effect of non-random constraints on the accuracy of the descriptive cost model, 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 optimal solutions on local search cost in workflow JSPs, reported in Table 3. As with general JSPs, we see both a dramatic drop in the number of optimal solutions and a gradual increase in local search cost as the backbone size is increased. Workflow JSPs have significantly fewer optimal solutions than general JSPs, and the local search cost is generally an order of magnitude higher. However, the log_{10}-log_{10} correlation between the number of optimal 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 computed the relative frequency of backbone sizes for both the 6×4 and 6×6 workflow JSPs; the resulting histograms are shown in Figure 2. Relative to general JSPs (Figure 1), it is clear that the presence of workflow partitions dramatically increases the frequency of large-backboned problem instances. For the rectangular 6×4 problems, workflow changes a bias toward small backbones in the general JSP into a relatively large bias toward large backbones. For the 6×6 problems, workflow further magnifies the already large bias toward large backbones found in the general JSP. 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 $d_{init-opt}$ and log_{10}(local search cost); the results are reported in the upper portion of Table 4. Here, we see a dramatic difference between general JSPs and workflow JSPs: while the influence of $d_{init-opt}$ at

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	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
	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 # 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.

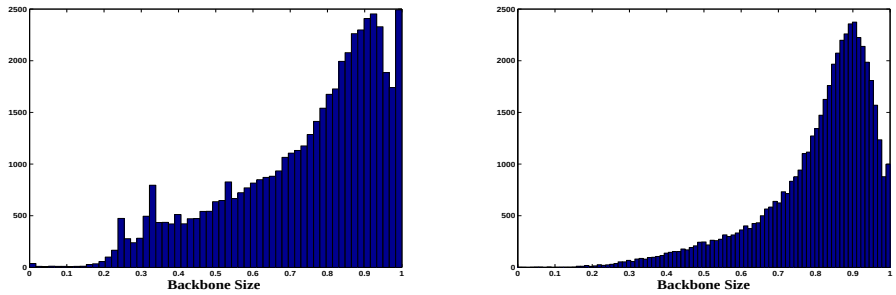

Fig. 2. Histogram of backbone sizes for 50 000 6 × 4 (left figure) and 6 × 6 (right figure) JSPs with workflow.

small backbones is relatively large, it drops very rapidly, ultimately vanishing at ≈ 0.9. Additionally, because of the lesser influence of $d_{init-opt}$ on local search cost, we see a corresponding drop in the influence of backbone robustness, as shown in the bottom half of Table 4. Given the strong bias toward large backbones in workflow JSPs, we conclude by noting that the factors present in the MAX-SAT and general JSP cost models are unable to account for *any* significant proportion of the variance in local search cost in these problems.

7 Discussion and Implications

Our results demonstrate that $d_{init-opt}$ is a good predictor of local search cost in both the general JSP and MAX-SAT, despite qualitative differences in the underlying search spaces. In both cases, $d_{init-opt}$ indirectly measures the size of the search space explored by the respective local search algorithms. Modern local search algorithms for MAX-SAT (e.g., Walk-SAT [SGS00]) basically 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 [Hoo98]. 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 $d_{init-opt}$ to predict local 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 local search cost.

Problem	Backbone Size				
Size	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
$d_{init-opt}$-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) $d_{init-opt}$ and 2) backbone robustness with log_{10}(local search cost) in workflow JSPs.

In contrast, we found that $d_{init-opt}$ was a very poor predictor of local search cost in workflow JSPs. Follow-up experiments indicate that there is a very strong bias toward particular sub-optimal solutions in some of these problems: there are many more 'paths' in the search space to sub-optimal solutions than to optimal solutions. In other problems, we have observed very distant clusters of optimal solutions, suggesting that a more complicated definition of $d_{init-opt}$ may be required. Our results also raise issues regarding the descriptive cost models for MAX-SAT, as we have shown that the factors influencing local search cost in random and structured problems *may* in fact be quite different.

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 Nowicki and Smutnicki's tabu search algorithm, 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. Finally, we note that our analysis is only directly applicable to tabu-like search algorithms for the JSP. Because descriptive cost models are tied to specific algorithms, it seems likely that other factors are responsible for local search cost in algorithms such as iterated local search or genetic algorithms, which are based on principles quite different from tabu search.

8 Conclusions

Our results clearly demonstrate that the factors influencing local search cost in MAX-SAT also influence local search cost in the general JSP, despite qualitative differences in the underlying search spaces. Consequently, we have a relatively clear picture of local search cost in the general JSP, although our model fails to account for a moderate amount of the variance in local search cost of large-backboned problem instances. Our results also suggest the possibility that these same factors may be applicable in a much wider 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 frequency of backbone sizes in the two problems: large backbones are very common in square problems, while we see a bias toward smaller backbones in rectangular problems.

Finally, we also demonstrate that the factors influencing search cost in the general JSP do not necessarily transfer to JSPs with workflow, suggesting that the descriptive cost models for random and structured problems may in fact be quite different.

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

References

[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.

[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.

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

[FCS97] Jeremy Frank, Peter Cheeseman, and John Stutz. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.

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

[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.

[MBK99] Dirk C. Mattfeld, Christian Bierwirth, and Herbert Kopfer. A search space analysis of the job shop scheduling problem. *Annals of Operations Research*, 86:441–453, 1999.

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

[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.