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 why these algorithm work so well, and under what conditions. We develop descriptive cost models of local search for the job-shop scheduling problem (JSP), borrowing from the models developed for MAX-SAT. 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 present in 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, in particular those based on tabu search, are among the most effective approaches for solving the JSP [1]. Yet, we have little understanding as to *why* these algorithms work so well, and under what conditions. In this paper, we examine 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 observed for different problem instances. We develop our models using the tabu search algorithm introduced in [2], which is closely related to many state-of-the-art algorithms for the JSP (e.g., [3]), but is significantly more amenable to analysis.

Researchers have expended significant effort in recent years to produce relatively accurate descriptive cost models of local search in MAX-SAT [4]. Intuitively, we would expect some of the 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 structures of the JSP differ in many important ways from MAX-SAT, making the a-priori applicability of these models unclear.

We investigate whether 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 optimal solutions [5], backbone size [6], the distance between initial solutions and the nearest optimal solution [4], and an analog of backbone robustness [4]. Together, these factors form the basis of a relatively accurate descriptive model of local search cost in the general JSP.

In contrast to real-world problems, the constraints in both MAX-SAT and the general JSP (clauses and machine processing orders, respectively) are randomly generated. We also consider descriptive cost models for JSPs with workflow, in which the machine processing orders are structured. Here, our experiments indicate that the factors present in the cost models for MAX-SAT and the general JSP only weakly influence search cost in workflow JSPs.

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, two general qualitative observations regarding problem difficulty have emerged:

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.

Clearly, any descriptive cost model must be consistent with these observations.

[7] identify 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 larger search cost associated with these problem instances. Similar differences were observed for two other quantitative search space measures (entropy and correlation length).

While there are significant differences in the search spaces of general and workflow JSPs, it is unclear whether these differences account for the often large variance in local search cost observed for different problem instances of the same size and workflow configuration. In preliminary experiments, we found that while the factors introduced in [7] *did* influence search cost, the influence was much weaker than for the factors we discuss in Section 5. Furthermore, [7] did not account for the relative difficulty of square versus rectangular JSPs.

3 The MAX-SAT Descriptive Cost Model

Many difficulty models have found application in a wide variety of problems, for example phase transitions and the associated peak in search cost. Given such apparent universals, it is important to examine, and if possible leverage, any existing work. To date, researchers have only developed descriptive cost models of local search 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 [8] 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 [5] 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 optimal solutions, where model residuals varied over three orders of magnitude.

A subsequent model was introduced in [4] that largely corrected for this deficiency, and went further by proposing a causal model of local search cost in MAX-SAT. The *backbone* of a problem instance is a key concept in this model. The backbone of a MAX-SAT instance consists of the subset of literals that have the same truth value in *all* optimal solutions[6]. [4] 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 local search cost. However, this correlation nearly vanishes ($r \approx -0.12$) when the backbone is large.

Local search algorithms for MAX-SAT quickly locate sub-optimal *quasi-solutions*, in which relatively few clauses are unsatisfied. 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 [4] to hypothesize that the size of this sub-space dictates the overall search cost.

To test this hypothesis, [4] 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 $d_{init\text{-}opt}$, and computed the correlation between $d_{init\text{-}opt}$ 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, $d_{init\text{-}opt}$, and not the number of optimal solutions, is the primary factor influencing local search cost in MAX-SAT.

[4] also posited a causal explanation for the variance in $d_{init\text{-}opt}$ 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 at least 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 or more. [4] argued that "backbone fragility approximately corresponds to how extensive the quasi-solution area is" ([4], p. 251), by noting that a fragile backbone allows for large $d_{init\text{-}opt}$ because of the sudden drop in backbone size, while $d_{init\text{-}opt}$ is necessarily small in problem instances with robust backbones.

As evidence of this hypothesis, [4] 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 to the hypothesis that "finding the backbone is less of an issue and so backbone fragility, which hinders this, has less of an effect" ([4], p. 254), although this conjecture was not explicitly tested.

4 Algorithms, Test Problems, and Methodology

We now introduce the tabu search algorithm, test problems, and methodology that we use to investigate descriptive cost models for the JSP.

4.1 Algorithm Description

Our analysis is based on the tabu search algorithm introduced in [2], which we denote $TS_{taillard}$. $TS_{taillard}$ 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 [3]. $TS_{taillard}$ uses the move operator introduced in [9], 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 "semi-active" local optima [2]. In each *iteration* of $TS_{taillard}$, all *N1* neighbors are generated, and the best non-tabu move is taken; ties are broken randomly. 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 during the current run. As indicated in [2](p. 110), frequency-based long-term memory is only necessary for problems that require a very large (> 1 million) 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 $TS_{taillard}$ 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 [2]), 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 5000 independent runs, which we denote $cost_{med}$. With 5000 samples, the estimate of the median is somewhat stable [10] [4].

For analysis purposes, the most important feature of $TS_{taillard}$ is the *N1* move operator. More advanced critical path move operators for the JSP, such as that used in [3], can induce search spaces that are *disconnected*, such that it is not always possible to move between random solutions and an optimal solution. Consequently, any algorithm using such a move operator is not Probabilistically Approximately Complete (PAC) [10]: 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. In contrast, *N1* induces a connected search space, enabling $TS_{taillard}$ (at least empirically) to be PAC: in producing the results discussed in Sections 5 and 6, no trial of $TS_{taillard}$ failed to locate an optimal solution.

4.2 Defining a Backbone for JSP

The definition of a backbone depends on how the solutions are represented. $TS_{taillard}$ 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 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 are functions of *all* optimal solutions to a problem instance, which can number in the millions for small problem instances. From a pragmatic standpoint, our experimental design necessitates examination of over 4000 problem instances, and includes computing the median search cost over 5000 independent runs for each instance; even for the small problems we consider, the overall time invested was approximately 6 CPU months on 1.5 GHz Pentium 4 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 [11].

5 A Descriptive Cost Model for the General JSP

A-priori, it is unclear whether the factors present in the MAX-SAT cost models affect problem difficulty in the JSP. The MAX-SAT 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 [12]. In contrast, the JSP search space is thought to be dominated by local optima with variable-sized and variable-depth attractor basins, and the focus is on escaping or avoiding the attractor basins of local optima. In this section, we 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 cost in the JSP.

5.1 The Number of Optimal Solutions and Search Cost

The first factor we consider is the number of optimal solutions to a problem instance, which we denote $|opt|$. In Table 1, we report summary statistics for both $|opt|$ and $cost_{med}$ for our general JSPs. We see both a dramatic drop in $|opt|$ and a gradual increase in $cost_{med}$ as the backbone size is increased. Further, at a fixed backbone size the difference in $cost_{med}$ between the 6×6 and 6×4 problems is minimal, and can be attributed to differences in the size of the search spaces.

In the bottom third of Table 1, we report the log_{10}– log_{10} correlation between $|opt|$ and $cost_{med}$. The r-values indicate that both $|opt|$ and backbone size influence local search cost in the general JSP; the unexpectedly weak r-value for the $6 \times 4 \approx 0.1$ problems is due to the large number of 0-cost samples. 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.

5.2 The 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

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. $X(Y)$ denotes a mean of X with a standard deviation of Y. The superscript notation $*(X)$ indicates that X instances (out of 100) had $cost_{med} = 0$, and were not used in the computation of r.

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	$\lvert opt \rvert$				
6×4	$481837(1 \times 10^6)$	30007(38072)	3221(3742)	642(1374)	22(22)
6×6	$1.1 \times 10^7(2 \times 10^7)$	$1.4 \times 10^6(3 \times 10^6)$	85292(157617)	9037(36713)	85(106)
	$cost_{med}$				
6×4	1.05(3.19)	14.95(21.28)	41.9(56.96)	93.28(151.43)	323.25(388.33)
6×6	13.8(29.50)	20.69(29.78)	40.48(57.75)	75.7(120.93)	447.12(1550.56)
	Correlation (r) between $log_{10}(\lvert opt \rvert)$ and $log_{10}(cost_{med})$				
6×4	$-0.0837^{*(82)}$	$-0.4332^{*(14)}$	-0.4852	-0.5080	-0.2846
6×6	$-0.7641^{*(38)}$	$-0.4491^{*(9)}$	-0.2651	-0.2398	-0.2298

$cost_{med}$ for each problem instance, leaving the backbone size uncontrolled. The mean $cost_{med}$ were 47.73 and 322.41 for the 6×4 and 6×6 problem sets, respectively, suggesting strong differences in the backbone size distributions.

In MAX-SAT, the distribution of backbone sizes depends on the ratio of the number of clauses c to the number of variables v [6]. Under-constrained problems (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.

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 frequencies of the backbone sizes. The most common backbone size for the square 6×6 instances is roughly 0.9, and backbones below 0.3 are very rare. In contrast, the distribution for the rectangular 6×4 instances is more uniform, 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 $|opt|$ and $cost_{med}$ 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 The Distance to Global Optima and Search Cost

For each of our general JSPs, we generated 5000 local optima, computed the Hamming distance to the nearest optimal solution for each of the resulting optima, and recorded the mean of the 5000 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

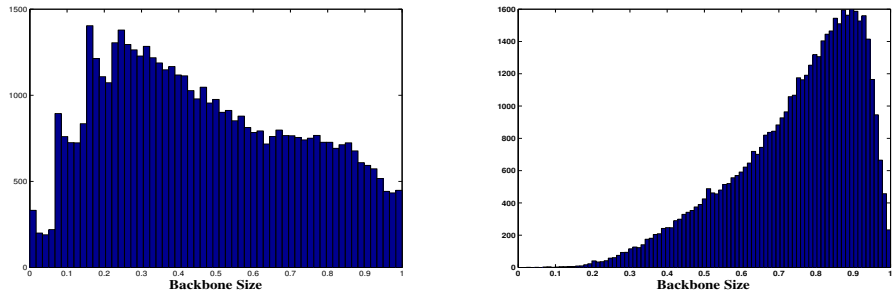

Fig. 1. Backbone size histograms for 6×4 (left) and 6×6 (right) general JSPs.

denote this measure by $d_{init\text{-}opt}$. We generated the local optima by applying a steepest-descent algorithm from random "semi-active" solutions[2]; when multiple equally good neighbors are available, ties are broken randomly.

In Table 2, we report the correlations between $d_{init\text{-}opt}$ and $log_{10}(cost_{med})$. 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 optimal solutions, and account for a significant proportion of the variance in local search cost for large-backboned problems; as with $|opt|$, the unexpectedly weak r-value for the $6 \times 4 \approx 0.1$ problems is due to the large number of 0-cost samples. To conclude, we also find that the distance between random local optima and global optima, and not the number of global optima, 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

In [4], backbone robustness is proposed as a causal factor in determining 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 $log_{10}(cost_{med})$ for our general JSPs. The results are very similar to those reported in [4] 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 sub-space containing local optima in the general JSP. As we

Table 2. Correlation of $d_{init\text{-}opt}$ and backbone robustness with local search cost in general JSPs. The superscript notation $*(X)$ indicates that X instances (out of 100) had $cost_{med} = 0$, and were not used in the computation of r.

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
Correlation (r) between $d_{init\text{-}opt}$ and $log_{10}(cost_{med})$					
6×4	$0.5295^{*(82)}$	$0.9391^{*(14)}$	0.8957	0.8486	0.6855
6×6	$0.9608^{*(38)}$	$0.9294^{*(9)}$	0.8924	0.8815	0.7423
Correlation (r) between backbone robustness and $log_{10}(cost_{med})$					
6×4	$-0.0357^{*(82)}$	$-0.2727^{*(14)}$	-0.3705	-0.4827	-0.5349
6×6	$-0.1829^{*(38)}$	$-0.2194^{*(9)}$	-0.4082	-0.5195	-0.5723

noted in Section 3, a justification for the lower correlations for small-backboned instances in provided in [4].

6 Extending the Analysis to Workflow JSPs

In the JSP and MAX-SAT, the primary problem constraints are the machine processing orders and the set of clauses, respectively. 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 the general JSP are applicable to more structured problem instances. To study the effect of non-random constraints on the efficacy of our descriptive cost models, we extend the analysis of Section 5 to JSPs with workflow–which impose a simple structure on the machine processing orders.

First, we consider the influence of $|opt|$ on local search cost in workflow JSPs, which we report in Table 3. As with general JSPs, we see both a dramatic drop in $|opt|$ and a gradual increase in $cost_{med}$ as the backbone size is increased. Workflow JSPs have significantly fewer optimal solutions than general JSPs, and $cost_{med}$ is generally an order of magnitude higher. However, the log_{10}-log_{10} correlations between $|opt|$ and $cost_{med}$ are qualitatively similar to the results for general JSPs: correlation is strong for small-backboned problems, but decays as backbone size is increased.

The backbone size histograms for both 6×4 and 6×6 workflow JSPs 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 strong bias toward large backbones found in the general JSP. We note that the rarity of small-backboned workflow JSPs further diminishes the utility of $|opt|$ as a predictor of $cost_{med}$ for these instances.

Finally, we measured the correlation between $d_{init\text{-}opt}$ and $log_{10}(cost_{med})$; 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 $d_{init\text{-}opt}$

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. $X(Y)$ denotes a mean of X with a standard deviation Y.

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
	$\lvert opt \rvert$				
6×4wf	26971(69299)	81255(295593)	2515(4704)	294(425)	18(16)
6×6wf	$1 \times 10^6(7 \times 10^6)$	$429102(2 \times 10^6)$	19017(44556)	4554(6898)	80(94)
	$cost_{med}$				
6×4wf	107.47(81.67)	104.63(107.71)	297.3(213.36)	741.19(600.62)	1829.65(1458.91)
6×6wf	286.26(359.26)	511.51(1020.56)	845.28(695.91)	1178.85(1109.19)	4454.84(5012.81)
	Correlation (r) between $log_{10}(\lvert opt \rvert)$ and $log_{10}(cost_{med})$				
6×4wf	-0.7591	-0.7935	-0.4527	-0.3701	-0.2341
6×6wf	-0.7718	-0.7591	-0.5346	-0.1997	-0.3089

at small backbones is somewhat strong, it drops very rapidly, ultimately vanishing at ≈ 0.9. Additionally, because of the lesser influence of $d_{init\text{-}opt}$ 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 $d_{init\text{-}opt}$ 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, $d_{init\text{-}opt}$ 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 [4]) 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 [10]. However, this inference also applies to $TS_{taillard}$: it is effectively performing a random walk in the space of local optima. The ability of $d_{init\text{-}opt}$ 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.

In contrast, we found that $d_{init\text{-}opt}$ 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 $d_{init\text{-}opt}$ 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 $TS_{taillard}$ precisely because

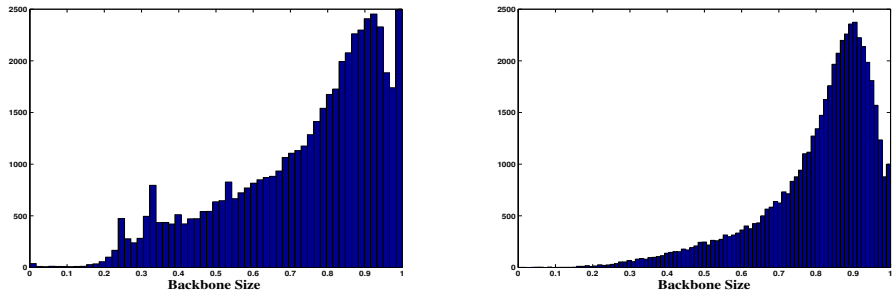

Fig. 2. Backbone size histograms for 6×4 (left) and 6×6 (right) workflow JSPs.

Table 4. Correlation of $d_{init\text{-}opt}$ and backbone robustness with local search cost in workflow JSPs.

Problem Size	Backbone Size				
	≈ 0.1	≈ 0.3	≈ 0.5	≈ 0.7	≈ 0.9
Correlation (r) between $d_{init\text{-}opt}$ and $log_{10}(cost_{med})$					
6×4wf	0.6892	0.5283	0.4106	0.3023	0.1752
6×6wf	0.5714	0.4901	0.5835	0.2114	0.2559
Correlation (r) between backbone robustness and $log_{10}(cost_{med})$					
6×4wf	-0.1298	-0.0752	-0.1301	-0.1296	-0.1930
6×6wf	-0.0224	-0.0650	-0.1834	-0.1921	-0.2147

it serves as a baseline for more advanced algorithms, such as the tabu search algorithm of [3], which enhance $TS_{taillard}$ 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

1. Blażewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research **93** (1996) 1–33
2. Éric D. Taillard: Parallel taboo search techniques for the job shop scheduling problem. ORSA Journal on Computing **6** (1994) 108–117
3. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Management Science **42(6)** (1996) 797–813
4. Singer, J., Gent, I.P., Smaill, A.: Backbone fragility and the local search cost peak. Journal of Artificial Intelligence Research **12** (2000) 235–270
5. Clark, D.A., Frank, J., Gent, I.P., MacIntyre, E., Tomov, N., Walsh, T.: Local search and the number of solutions. In: Proceedings of the Second International Conference on Principles and Practices of Constraint Programming (CP-96). (1996) 119–133
6. Parkes, A.J.: Clustering at the phase transition. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97). (1997) 340–345
7. Mattfeld, D.C., Bierwirth, C., Kopfer, H.: A search space analysis of the job shop scheduling problem. Annals of Operations Research **86** (1999) 441–453
8. Riedys, C.M., Stadler, P.F.: Combinatorial landscapes. Technical Report 01-03-014, The Santa Fe Institute (2001)
9. van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K.: Job shop scheduling by simulated annealing. Operations Research **40** (1992) 113–125
10. Hoos, H.H.: Stochastic Local Search - Methods, Models, Applications. PhD thesis, Darmstadt University of Technology (1998)
11. Beck, J.C., Fox, M.S.: Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. Artificial Intelligence **117** (2000) 31–81
12. Frank, J., Cheeseman, P., Stutz, J.: When gravity fails: Local search topology. Journal of Artificial Intelligence Research **7** (1997) 249–281