Journal of Artificial Intelligence Research 24 (2005) 221–261 Submitted 09/04; published 08/05

Linking Search Space Structure, Run-Time Dynamics, and Problem Difficulty: A Step Toward Demystifying Tabu Search

Jean-Paul Watson JWATSON@SANDIA.GOV
Sandia National Laboratories
P.O. Box 5800, MS 1110
Albuquerque, NM 87185-1110 USA

L. Darrell Whitley WHITLEY@CS.COLOSTATE.EDU
Adele E. Howe HOWE@CS.COLOSTATE.EDU
Computer Science Department
Colorado State University
Fort Collins, CO 80523 USA

Abstract

Tabu search is one of the most effective heuristics for locating high-quality solutions to a diverse array of NP-hard combinatorial optimization problems. Despite the widespread success of tabu search, researchers have a poor understanding of many key theoretical aspects of this algorithm, including models of the high-level run-time dynamics and identification of those search space features that influence problem difficulty. We consider these questions in the context of the job-shop scheduling problem (JSP), a domain where tabu search algorithms have been shown to be remarkably effective. Previously, we demonstrated that the mean distance between random local optima and the nearest optimal solution is highly correlated with problem difficulty for a well-known tabu search algorithm for the JSP introduced by Taillard. In this paper, we discuss various shortcomings of this measure and develop a new model of problem difficulty that corrects these deficiencies. We show that Taillard's algorithm can be modeled with high fidelity as a simple variant of a straightforward random walk. The random walk model accounts for nearly all of the variability in the cost required to locate both optimal and sub-optimal solutions to random JSPs, and provides an explanation for differences in the difficulty of random versus structured JSPs. Finally, we discuss and empirically substantiate two novel predictions regarding tabu search algorithm behavior. First, the method for constructing the initial solution is highly unlikely to impact the performance of tabu search. Second, tabu tenure should be selected to be as small as possible while simultaneously avoiding search stagnation; values larger than necessary lead to significant degradations in performance.

1. Introduction

Models of problem difficulty have excited considerable recent attention (Cheeseman, Kanefsky, & Taylor, 1991; Clark, Frank, Gent, MacIntyre, Tomov, & Walsh, 1996; Singer, Gent, & Smaill, 2000). These models[1] are designed to account for the variability in search cost observed for one or more algorithms on a wide range of problem instances and have yielded

1. We refer to models of problem difficulty as *cost models* throughout this paper.

significant insight into the relationship between search space structure, problem difficulty, and algorithm behavior.

In this paper, we investigate cost models of tabu search for the job-shop scheduling problem (JSP). The JSP is an NP-hard combinatorial optimization problem and has become one of the standard and most widely studied problems in scheduling research. Tabu search algorithms are regarded as among the most effective approaches for generating high-quality solutions to the JSP (Jain & Meeran, 1999) and currently represent the state-of-the-art by a comfortable margin over the closest competition (Nowicki & Smutnicki, 2005; Błażewicz, Domschke, & Pesch, 1996). While researchers have achieved considerable advances in performance since Taillard first demonstrated the effectiveness of tabu search algorithms for the JSP in 1989, comparatively little progress has been made toward developing an understanding of how these algorithms work, i.e., characterizing the underlying high-level search dynamics, understanding why these dynamics are so effective on the JSP, and deducing how these dynamics might be modified to yield further improvements in performance.

It is well-known that the structure of the search space influences problem difficulty for tabu search and other local search algorithms (Reeves, 1998). Consequently, a dominant approach to developing cost models is to identify those features of the search space that are highly correlated with search cost. We have performed extensive analyses of the relationship between various search space features and problem difficulty for Taillard's tabu search algorithm for the JSP (Watson, Beck, Howe, & Whitley, 2001, 2003). Our findings were largely negative: many features that are widely believed to influence problem difficulty for local search are, in fact, only weakly correlated with problem difficulty. Features such as the number of optimal solutions (Clark et al., 1996), the backbone size (Slaney & Walsh, 2001), and the mean distance between random local optima (Mattfeld, Bierwirth, & Kopfer, 1999) account for less than a third of the total variability in search cost for Taillard's algorithm. In contrast, drawing from research on problem difficulty for local search and MAX-SAT (Singer et al., 2000), we found that the mean distance between random local optima and the nearest optimal solution, which we denote $\overline{d}_{lopt\text{-}opt}$, is highly correlated with problem difficulty, accounting for at least 2/3 of the total variability in search cost (Watson et al., 2003). We further demonstrated that $\overline{d}_{lopt\text{-}opt}$ accounts for much of the variability in the cost of locating sub-optimal solutions to the JSP, and for differences in the relative difficulty of "square" versus "rectangular" JSPs.

Nevertheless, the $\overline{d}_{lopt\text{-}opt}$ cost model has several shortcomings. First, the expense of computing $\overline{d}_{lopt\text{-}opt}$ limited our analyses to relatively small problem instances, raising concerns regarding scalability to more realistically sized problem instances. Second, residuals under the $\overline{d}_{lopt\text{-}opt}$ model are large for a number of problem instances, and the model is least accurate for the most difficult problem instances. Third, because the $\overline{d}_{lopt\text{-}opt}$ model provides no direct insight into the run-time behavior of Taillard's algorithm, we currently do not understand *why* $\overline{d}_{lopt\text{-}opt}$ is so highly correlated with search cost.

We introduce a novel cost model that corrects for the aforementioned deficiencies of the $\overline{d}_{lopt\text{-}opt}$ cost model. This model, which based on a detailed analysis of the run-time behavior of Taillard's algorithm, is remarkably accurate, accounting for over 95% of the variability in the cost of locating both optimal and sub-optimal solutions to a wide range of problem instances. More specifically, we establish the following results:

1. Search in Taillard's algorithm appears to be effectively restricted to a sub-space S_{lopt+} of solutions that contains both local optima and solutions that are very close to (specifically, 1-2 moves away from) local optima.

2. Taillard's algorithm can be modeled with remarkable fidelity as a variant of a simple one-dimensional random walk over the S_{lopt+} sub-space. The walk exhibits two notable forms of bias in the transition probabilities. First, the probability of search moving closer to (farther from) the nearest optimal solution is proportional (inversely proportional) to the current distance from the nearest optimal solution. Second, search exhibits momentum: it is more likely to move closer to (farther from) the nearest optimal solution if search previously moved closer to (farther from) the nearest optimal solution. Moreover, the random walk model accounts for at least 96% of the variability in the mean search cost across a range of test instances.

3. The random walk model is equally accurate for random, workflow, and flowshop JSPs. However, major differences exist in the number of states in the walk, i.e., the maximal possible distance between a solution and the nearest optimal solution. Differences in these maximal distances fully account for well-known differences in the difficulty of problem instances drawn from these various sub-classes.

4. The accuracy of the random walk model transfers to a tabu search algorithm based on the powerful *N5* move operator, which is more closely related to state-of-the-art tabu search algorithms for the JSP than Taillard's algorithm.

5. The random walk model correctly predicts that initiating Taillard's algorithm from high-quality starting solutions will only improve performance if those solutions are very close to the nearest optimal solution.

6. The random walk model correctly predicts that any tabu tenure larger than the minimum required to avoid search stagnation is likely to increase the fraction of the search space explored by Taillard's algorithm, and as a consequence yield a net *increase* in problem difficulty. Informally, the detrimental nature of large tabu tenures is often explained simply by observing that large tenures impact search through the loss of flexibility and the resulting inability to carefully explore the space of neighboring solutions. Our results provide a more detailed and concrete account of this phenomenon in the context of the JSP.

The remainder of this paper is organized as follows. We begin in Sections 2 and 3 with a description of the JSP, Taillard's algorithm, and the problem instances used in our analysis. The hypothesis underlying our analysis is detailed in Section 4. We summarize and critique prior research on problem difficulty for tabu search algorithms for the JSP in Section 5. Sections 6 through 9 form the core of the paper, in which we develop and validate our random walk model of Taillard's algorithm. In Section 10 we explore the applicability of the random walk model to more structured problem instances. Section 11 explores the applicability of the random walk model to a tabu search algorithm that is more representative of state-of-the-art algorithms for the JSP than Taillard's algorithm. Section 12 details two uses of the random walk model in a predictive capacity. We conclude by discussing the implications of our results and directions for future research in Section 13.

2. Problem and Test Instances

We consider the well-known $n \times m$ static, deterministic JSP in which n jobs must be processed exactly once on each of m machines (Błażewicz et al., 1996). Each job i ($1 \leq i \leq n$) is routed through each of the m machines in a pre-defined order π_i, where $\pi_i(j)$ denotes the jth machine ($1 \leq j \leq m$) in the routing order of job i. The processing of job i on machine $\pi_i(j)$ is denoted o_{ij} and is called an operation. An operation o_{ij} must be processed on machine $\pi_i(j)$ for an integral duration $\tau_{ij} \geq 0$. Once initiated, processing cannot be pre-empted and concurrency on individual machines is not allowed, i.e., the machines are unit-capacity resources. For $2 \leq j \leq m$, o_{ij} cannot begin processing until $o_{i(j-1)}$ has completed processing. The scheduling objective is to minimize the makespan C_{max}, i.e., the completion time of the last operation of any job. Makespan-minimization for the JSP is NP-hard for $m \geq 2$ and $n \geq 3$ (Garey, Johnson, & Sethi, 1976).

An instance of the $n \times m$ JSP is uniquely defined by the set of nm operation durations τ_{ij} and n job routing orders π_i. We define a *random* JSP as an instance generated by (1) sampling the τ_{ij} independently and uniformly from an interval $[LB, UB]$ and (2) constructing the π_i from random permutations of the integer sequence $\zeta = 1, \ldots, m$. Most often $LB = 1$ and $UB = 99$ (Taillard, 1993; Demirkol, Mehta, & Uzsoy, 1998). The majority of JSP benchmark instances, including most found in the OR Library[2], are random JSPs.

Non-random JSPs can be constructed by imposing structure on either the τ_{ij}, the π_i, or both. To date, researchers have only considered instances with structured π_i, although it is straightforward to adapt existing methods for generating non-random τ_{ij} for flow shop scheduling problems (Watson, Barbulescu, Whitley, & Howe, 2002) to the JSP. One approach to generating structured π_i involves partitioning the set of m machines into wf contiguous, equally-sized subsets called *workflow partitions*. For example, when $wf = 2$, the set of m machines is partitioned into two subsets containing the machines 1 through $m/2$ and $m/2 + 1$ through m, respectively. In such a two-partition scheme, every job must be processed on all machines in the first partition before proceeding to any machine in the second partition. No constraints are placed on the job routing orders within each partition. We refer to JSPs with $wf = 2$ simply as *workflow* JSPs. Less common are *flowshop* JSPs, where $wf = m$, i.e., all of the jobs visit the machines in the same pre-determined order.

While the presence of structure often makes scheduling problems easier to solve (Watson et al., 2002), this is not the case for JSPs with structured π_i. Given fixed n and m, the average difficulty of problem instances – as measured by the cost required to either locate an optimal solution or to prove the optimality of a solution – is empirically proportional to wf. In other words, random JSPs are generally the easiest instances, while flowshop JSPs are the most difficult. Evidence for this observation stems from a wide variety of sources. For example, Storer et al. (1992) introduced sets of 50×10 random and workflow JSPs in 1992; the random JSPs were quickly solved to optimality, while the optimal makespans of all but one of the workflow JSPs are currently unknown. Similarly, the most difficult 10×10 benchmark problems, Fisher and Thompson's infamous 10×10 instance and Applegate and Cook's (1991) `orb` instances, are all "nearly" workflow or flowshop JSPs, in that the requirement that a job be processed on all machines in one workflow partition before proceeding to any machine in the next workflow partition is slightly relaxed.

2. http://www.brunel.ac.uk/depts/ma/research/jeb/orlib/jobshopinfo.html

Accurate cost models of local search algorithms are generally functions of the set of globally optimal solutions to a problem instance (Watson et al., 2003; Singer et al., 2000), and the cost models we develop here further emphasize this dependency. Due in part to the computational cost of enumeration, our analysis is largely restricted to sets of 6×4 and 6×6 random, workflow, and flowshop JSPs. Each set contains 1,000 instances apiece, with the τ_{ij} sampled from the interval $[1, 99]$. To assess the scalability of our cost models, we also use a set of 100 10×10 random JSPs, where the τ_{ij} are uniformly sampled from the interval $[1, 99]$. In Section 10, we consider sets of 10×10 workflow and flowshop JSPs generated in an analogous manner. For comparative purposes with the literature, we additionally report results for various 10×10 instances found in the OR Library. Although the 10×10 OR Library instances are no longer considered particularly challenging, they have received significant historical attention and serve to validate results obtained using our own 10×10 problem set. For each instance in each of the aforementioned problem sets, a variant of Beck and Fox's (2000) constraint-directed scheduling algorithm was used to compute both the optimal makespan and the set of optimal solutions.

3. The Algorithm: Tabu Search and the JSP

Numerous tabu search algorithms have been developed for the JSP (Jain & Meeran, 1999). For our analysis, we select an algorithm introduced by Taillard in 1994. We implemented a variant of Taillard's algorithm, which we denote TS_{N1}[3], and easily reproduced results consistent with those reported by Taillard. TS_{N1} is *not* a state-of-the-art tabu search algorithm for the JSP; the algorithms of Nowicki and Smutnicki (2005), Pezzella and Merelli (2000), and Barnes and Chambers (1995) yield stronger overall performance. All of these algorithms possess a core tabu search mechanism that is very similar to that found in TS_{N1}, but differ in the choice of move operator, the method used to generate initial solutions, and the use of long-term memory mechanisms such as reintensification.

Our choice of TS_{N1} is pragmatic. Before tackling more complex, state-of-the-art algorithms, we first develop cost models of a relatively simple but representative version of tabu search and then systematically assess the influence of more complex algorithmic features on cost models of the basic algorithm. Consequently, our implementation of TS_{N1} deviates from Taillard's original algorithm in three respects. First, we compute solution makespans exactly instead of using a computationally efficient estimation scheme. Second, we do not use frequency-based memory; Taillard (1994, p. 100) indicates that the benefit of such memory is largely restricted to instances requiring a very large number (i.e., > 1 million) of iterations. Third, we initiate trials of TS_{N1} from random local optima (using a scheme described below) instead of those resulting from Taillard's deterministic construction method. As discussed in Section 12.1, there is strong evidence that the type of the initial solution has a negligible impact on the speed with which TS_{N1} locates optimal solutions, which we take as the primary objective in our analysis.

3. TS_{N1} is identical to the algorithm denoted $TS_{Taillard}$ in our earlier paper (Watson et al., 2003); the new notation was chosen to better convey the fact that the algorithm deviates from Taillard's original algorithm in several respects and to emphasize the relative importance of the move operator.

TS_{N1} uses van Laarhoven et al.'s (1992) well-known *N1* move operator, which swaps adjacent operations on critical blocks in the schedule.[4] In our implementation, and in contrast to many local search algorithms for the JSP (Nowicki & Smutnicki, 1996), *all* pairs of adjacent critical operations are considered and not just those on a single critical path. At each iteration of TS_{N1}, the makespan of each neighbor of the current solution s is computed and the non-tabu neighbor $s' \in N1(s)$ with the smallest makespan is selected for the next iteration; ties are broken randomly. Let (o_{ij}, o_{kl}) denote the pair of adjacent critical operations that are swapped in s to generate s', such that o_{ij} appears before o_{kl} in the processing order of machine $\pi_i(j)$. In the subsequent L iterations, TS_{N1} prevents or labels as "tabu" any move that inverts the operation pair (o_{kl}, o_{ij}). The idea, a variant of frequency-based memory, is to prevent recently swapped pairs of critical operations from being re-established. The scalar L is known as the tabu tenure and is uniformly sampled every $1.2L_{max}$ iterations from a fixed-width interval $[L_{min}, L_{max}]$; such dynamic tabu tenures can avoid well-known cyclic search behaviors associated with fixed tabu tenures (Glover & Laguna, 1997). Let s_{best} denote the best solution located during any iteration of the current run or trial of TS_{N1}. When $C_{max}(s') < C_{max}(s_{best})$, the tabu status of s' is negated; in other words, TS_{N1} employs a simple aspiration level criterion. In rare cases, the minimal neighboring makespan may be achieved by both non-tabu and tabu-but-aspired moves, in which case a non-tabu move is always accepted. If all moves in $N1(s)$ are tabu, no move is accepted for the current iteration. We observe that in the absence of tabu moves preventing improvement of the current solution's makespan, TS_{N1} acts as a simple greedy descent procedure. More specifically, it is clear that the core search bias exhibited by TS_{N1} is steepest-descent local search, such that there is significant pressure toward local optima.

Tabu tenure can have a major impact on performance. Based on empirical tests, Taillard defines $L_{min} = 0.8X$ and $L_{max} = 1.2X$, where $X = (n+m/2)\cdot e^{-n/5m} + N/2\cdot e^{-5m/n}$ (Taillard, 1994); n and m are respectively the number of jobs and machines in the problem instance and $N = nm$. In preliminary experimentation, we observed that the resulting tenure values for our 6×4 and 6×6 problem sets (respectively $[3, 5]$ and $[4, 6]$) failed to prevent cycling or stagnation behavior. Instead, we set $[L_{min}, L_{max}]$ equal to $[6, 14]$ for all trials involving these instances and re-sample the tabu tenure every 15 iterations. For 10×10 instances we set $[L_{min}, L_{max}]$ equal to $[8, 14]$ for all trials and again re-sample the tabu tenure every 15 iterations; the specific values are taken from Taillard's research, which also ignored the aforementioned rule for trials involving 10×10 instances (Taillard, 1994). Taillard's rules are used unmodified in all trials involving larger problem instances, e.g., those analyzed below in Section 4.

3.1 Cost and Distance Metrics

Unlike more effective JSP move operators such as *N5* (Nowicki & Smutnicki, 1996), the *N1* operator induces search spaces that are *connected*, in that it is always possible to move from an arbitrary solution to a global optimum. Consequently, it is possible to construct a local search algorithm based on *N1* that is probabilistically approximately complete (PAC) (Hoos, 1998), such that an optimal solution will eventually be located given sufficiently large run-times. Our experimental results suggest that TS_{N1} is PAC, subject to reasonable settings for

4. Our notation for move operators is taken from Błażewicz (1996).

the tabu tenure; given our rules for selecting L_{min} and L_{max}, no trial of TS_{N1} failed to locate an optimal solution to any of the problem instances described in Section 2. In particular, the tabu tenure must be large enough for TS_{N1} to escape local optima; using short tabu tenures, it is straightforward to construct examples where TS_{N1} will become permanently trapped in the attractor basin of a single local optimum. To be provably PAC under general parameter settings, the TS_{N1} algorithm would likely require modifications enabling it to accept an arbitrary move at any given iteration, allowing search to always progress toward a global optimum; Hoos (1998) discusses similar requirements in the context of PAC local search algorithms for MAX-SAT. We have not pursued such modifications because they ignore practical efficiency issues associated with poor parameter value selection, and because it is unclear how the induced randomness would impact the core tabu search dynamics.

The empirical PAC property enables us to naturally define the cost required to solve a given problem instance for a *single* trial of TS_{N1} as the number of iterations required to locate a globally optimal solution. In general, search cost under TS_{N1} is a random variable with an approximately exponential distribution, as we discuss in Section 9. Consequently, we define the search cost for a given problem instance as either the median or mean number of iterations required to locate an optimal solution; the respective quantities are denoted by c_{Q2} and $\overline{c}$. We estimate both c_{Q2} and $\overline{c}$ using 1,000 independent trials. Due to the exponential nature of the underlying distribution, a large number of samples is required to achieve reasonably accurate estimates of both statistics.

Our analysis relies on the notion of the distance $D(s_1, s_2)$ between two solutions s_1 and s_2, which we take as the well-known disjunctive graph distance (Mattfeld et al., 1999). Let $\gamma(i, j, k, s)$ denote a predicate that determines whether job j appears before job k in the processing order of machine i of solution s. The disjunctive graph distance $D(s_1, s_2)$ between s_1 and s_2 is then defined as

$$D(s_1, s_2) = \sum_{i=1}^{m} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \gamma(i, j, k, s_1) \oplus \gamma(i, j, k, s_2)$$

where the symbol $\oplus$ denotes the Boolean XOR operator. Informally, the disjunctive graph distance simply captures the degree of heterogeneity observed in the machine processing sequences of two solutions. A notable property of the disjunctive graph distance is that it serves as a lower bound, which is empirically tight, on the number of *N1* moves required to transform s_1 into s_2. This is key in our analysis, as computation of the exact number of *N1* moves required to transform s_1 into s_2 is NP-hard (Vaessens, 1995). In the remainder of this paper, we use the terms “distance” and “disjunctive graph distance” synonymously.

Finally, we define a “random local optimum” as a solution resulting from the application of steepest-descent local search under the *N1* operator to a random *semi-active* solution. A semi-active solution is defined as a feasible solution (i.e., lacking cyclic ordering dependencies) in which all operations are processed at their earliest possible starting time. To construct random semi-active solutions, we use a procedure introduced by Mattfeld (1996, Section 2.2). The steepest-descent procedure employs random tie-breaking in the presence of multiple equally good alternatives and terminates once a solution s is located such that $\forall s' \in N1(s)$, $C_{max}(s) \leq C_{max}(s')$.

4. The Run-Time Behavior of TS_{N1}: Motivating Observations

For a given problem instance, consider the space of feasible solutions S and the sub-space $S_{lopt} \subseteq S$ containing all local optima. Due to the strong bias toward local optima induced by the core steepest-descent strategy, we expect TS_{N1} to frequently sample solutions in S_{lopt} during search. However, the degree to which solutions in S_{lopt} are actually *representative* of solutions visited by TS_{N1} is a function of both the strength of local optima attractor basins in the JSP and the specifics of the short-term memory mechanism. In particular, strong attractor basins would require TS_{N1} to move far away from local optima in order to avoid search stagnation. Recently, Watson (2003) showed that the attractor basins of local optima in the JSP are surprisingly weak in general and can be escaped with high probability simply by (1) accepting a short random sequence (i.e., of length 1 or 2 elements) of monotonically worsening moves and (2) re-initiating greedy descent. In other words, relatively small perturbations are sufficient to move local search out of the attractor basin of a given local optimum in the JSP. We observe that the aforementioned procedure provides an operational definition of attractor basin strength, e.g., a specific escape probability given a "worsening" random sequence of length k, in contrast to more informal notions such as narrowness, width, or diameter.

Based on these observations, we hypothesize that search in TS_{N1} is effectively restricted to the sub-space $S_{lopt+} \supset S_{lopt}$ containing both local optima and solutions that are very close to local optima in terms of disjunctive graph distance or, equivalently, the number of *N1* moves. To test this hypothesis, we monitor the *descent distance* of solutions visited by TS_{N1} during search on a range of random JSPs taken from the OR Library. We define the descent distance of a candidate solution s as the disjunctive graph distance $D(s, s')$ between s and a local optimum s' generated by applying steepest-descent under the *N1* operator to s. In reality, descent distance is stochastic due to the use of random tie-breaking during steepest-descent. We avoid exact characterization of descent distance for any particular s and instead compute descent distance statistics over a wide range of s. For each problem instance, we execute TS_{N1} for one million iterations, computing the descent distance of the current solution at each iteration and recording the resulting time-series; trials of TS_{N1} are terminated once an optimal solution is encountered and re-started from a random local optimum. Several researchers have introduced measures that are conceptually related to descent distance, but are in contrast based on differences in solution fitness. For example, search *depth* (Hajek, 1988) is defined as the minimal increase in fitness (assuming minimization) that must be accepted in order to escape a local optimum attractor basin. Similarly, Schuurmans and Southey (2001) define search depth as the difference between the fitness of a solution s and that of a global optimum s^*.

Summary statistics for the resulting descent distances are reported in Table 1; for comparative purposes, we additionally report the mean descent distance observed for one million random semi-active solutions. The mean and median descent distance statistics indicate that TS_{N1} consistently remains only 1–2 moves away from local optima, *independent* of problem size. Although search is occasionally driven very far from local optima, such events are rare - as corroborated by the low standard deviations. Empirical evidence suggests that large-distance events are not due to the existence of local optima with very deep attractor basins, but are rather an artifact of TS_{N1}'s short-term memory mechanism. These results