An Analysis of Iterated Local Search for Job-Shop Scheduling

Jean-Paul Watson* Adele E. Howe* L. Darrell Whitley*

*Department of Computer Science, Colorado State University
Fort Collins, Colorado 80523, USA
{watsonj,howe,whitley}@cs.colostate.edu

1 Introduction and Motivation

Iterated local search, or ILS, is among the most straightforward meta-heuristics for local search. ILS employs both *small-step* and *large-step* move operators. Search proceeds via iterative modifications to a single solution, in distinct alternating phases. In the first phase, local neighborhood search (typically greedy descent) is used in conjunction with the small-step operator to transform solutions into local optima. In the second phase, the large-step operator is applied to generate perturbations to the local optima obtained in the first phase. Ideally, when local neighborhood search is applied to the resulting solution, search will terminate at a different local optimum, i.e., the large-step perturbations should be sufficiently large to enable escape from the attractor basins of local optima.

ILS has proven capable of delivering excellent performance on numerous NP-hard optimization problems [LMS03]. However, despite its simplicity, very little is known about why ILS can be so effective, and under what conditions. The goal of this paper is to advance the state-of-the-art in the *analysis* of meta-heuristics, by providing answers to this research question. We focus on characterizing both the relationship between the structure of the underlying search space and ILS performance, and the dynamic behavior of ILS. Our analysis proceeds in the context of the job-shop scheduling problem (JSP) [Tai94]. We begin by demonstrating that the attractor basins of local optima in the JSP are surprisingly weak, and can be escaped with high probability by accepting a short random sequence of less-fit neighbors. This result is used to develop a new ILS algorithms for the JSP, *I-JAR*, whose performance is competitive with tabu search on difficult benchmark instances. We conclude by developing a very accurate behavioral model of *I-JAR*, which yields significant insights into the dynamics of search.

Our analysis is based on a set of 100 random 10×10 problem instances, in addition to some widely used benchmark instances. Both I-JAR and the tabu search algorithm we consider are based on the *N1* move operator introduced by van Laarhoven et al. [vLAL92]. The *N1* operator induces a connected search space, such that it is always possible to move from an arbitrary solution to an optimal solution; this property is integral to the development of a behavioral model of *I-JAR*. However, much of our analysis generalizes to other move operators, including that of Nowicki and Smutnicki [NS96]. Finally, our models are based on the distance between two solutions, which we take as the well-known disjunctive graph distance [MBK99].

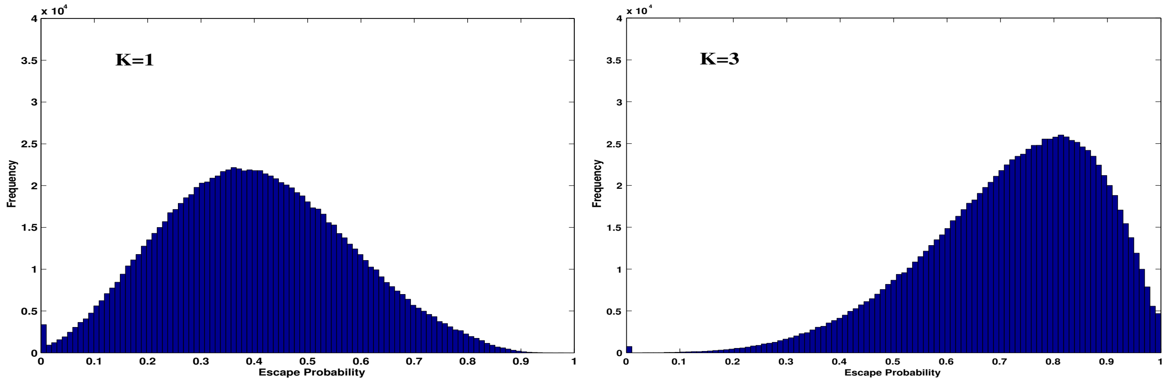

Figure 1: Distribution of the escape probability from the local optima of random 10×10 JSPs after accepting a random sequence of 1 (left figure) and 3 (right figure) less-fit neighbor(s).

2 Attractor Basin Strength in the JSP

A local optimum s is a solution possessing no higher-fitness neighbors with respect to some move operator. The attractor basin of s is the set of solutions S such that $\forall s' \in S$, s results with a non-zero probability when greedy descent is applied to s'; basin membership is stochastic because greedy descent is typically randomized. To escape a local optimum, it is generally necessary to accept a sequence of monotonically less-fit neighbors such that greedy descent terminates at a different local optimum with a non-zero probability. The question is then: How *many* non-improving moves are required? Or, in terms of ILS, how large of a large-step move is required to escape local optima.

Consider a local optimum s, and suppose that we accept a random sequence of monotonically less-fit neighbors of length at *most* k, yielding s'; local maxima can be encountered before k moves have been accepted. If next-descent initiated from s' terminates at a local optimum s'' with a makespan different from s, then we have escaped the attractor basin of s. We then define the escape probability of a local optimum s for a given k as the fraction of 100 independent trials that result in an s'' with a makespan different from s. For each of our random 10×10 instances, we computed the escape probability for 10,000 random local optima over a range of k. The aggregate distributions for $k = 1$ and $k = 3$ are shown in Figure 1; results for individual instances are similar. We measured no correlation between the escape probability and solution quality, and observed similar results over a range of problem sizes.

3 Iterated Jump-And-Redescend

We use the relative weakness of attractor basins in the JSP to motivate the design of a new ILS algorithm, which we denote *I-JAR* for *I*terated *J*ump *A*nd *R*edescend. At each iteration of search, *I-JAR* simply accepts a random sequence of at most k monotonically less-fit neighbors of the current local optima s, resulting in a new solution s'. Randomized next-descent is then applied to s' to generate a new local optimum s'', which serves as the starting point for the next iteration. *I-JAR* is initiated from a random local optimum, and continues until some termination criterion is satisfied. To enable escape from local optima that are also local

		I-JAR			$TS_{Taillard}$		
Instance	Optimal Makespan	Min.	Mean	Max.	Min.	Mean	Max.
ta01	1231	1233	1242	1246	**1231**	*1237.37*	1244
ta02	1244	**1244**	*1244.73*	1247	**1244**	1250.6	1256
ta03	1218	**1218**	*1221.53*	1224	**1218**	1222.33	1223
ta04	1175	**1175**	*1180.4*	1181	**1175**	1180.83	1182
ta05	1224	1229	*1232.37*	1233	1228	1232.83	1236
ta06	1238	1239	*1243.23*	1246	1240	1243.33	1246
ta07	1227	1228	1228	1228	1228	1228	1228
ta08	1217	**1217**	*1218.13*	1222	**1217**	1220.13	1227
ta09	1274	**1274**	*1281.57*	1287	**1274**	1282.33	1289
ta10	1241	**1241**	*1243.57*	1244	1244	1250.17	1256

Table 1: Statistics for the makespans of the best solutions obtained by *I-JAR* and $TS_{Taillard}$ on 15×15 benchmark instances. Bold-faced entries indicate the solution is optimal. Italicized entries indicate the mean makespan is less than that of its competitor.

maxima or near local maxima, we accept 5 random moves from s with probability $p = 0.01$ for any given iteration. Unlike existing ILS algorithms for the JSP, e.g., see [Lou95], *I-JAR*'s large-step move operator is explicitly based on an analysis of the underlying search space, as opposed to detailed, problem-specific knowledge.

4 Performance

The performance of *I-JAR* is dictated primarily by k, the maximal length of the jump taken before next-descent is re-initiated. While the escape probability is proportional to k, so is the CPU time per iteration – due to increases in both k and the length of the redescent. Thus, any increase in k must be balanced by a decrease in the total number of iterations required to achieve a particular objective. Empirically, for 10×10 and 15×15 instances, $k = 2$ yields provides the best overall performance, yielding an escape for 60% of all iterations.

To demonstrate the effectiveness of *I-JAR*, we measure performance relative to Taillard's [Tai94] tabu search algorithm, which we denote $TS_{Taillard}$. Both *I-JAR* and $TS_{Taillard}$ are based on the *N1* operator and do not re-intensify search around high-quality solutions. We chose $TS_{Taillard}$ to perform a *controlled* experiment, in that the only variable is the core meta-heuristic. Although beyond the scope of the present paper, we have also shown that re-intensification significantly and *equally* improves the performance of both algorithms [Wat03].

Performance is measured relative to ten moderately difficult 15×15 benchmark instances, denoted `ta01` – `ta10`. For each instance, we ran 30 independent trials of *I-JAR* and $TS_{Taillard}$ for 10 and 30 million iterations each, respectively; the cost-per-iteration ratio is also roughly 3:1 for these instances. Statistics for the makespans of the best solutions obtained during each trial are shown in Table 1. Both algorithms achieve excellent absolute performance, frequently locating optimal solutions. For some instances (e.g., `ta01`) there are slight differences in mean performance, although in no case are the differences statistically significant. Our results clearly demonstrate that the ILS meta-heuristic can achieve performance that is competitive with tabu search on moderately difficult benchmark problems; we also show similarly strong performance on larger, more difficult benchmark instances using Nowicki and Smutnicki's *TSA* tabu search algorithm [NS96] and a version of *I-JAR* based on the same move operator.

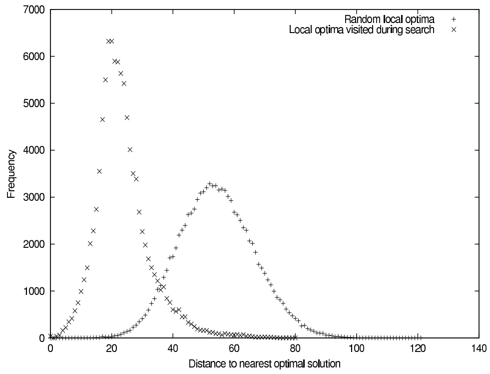

Figure 2: The distribution of the distance to the nearest optimal solution for (a) random local optima and (b) solutions visited by *I-JAR*.

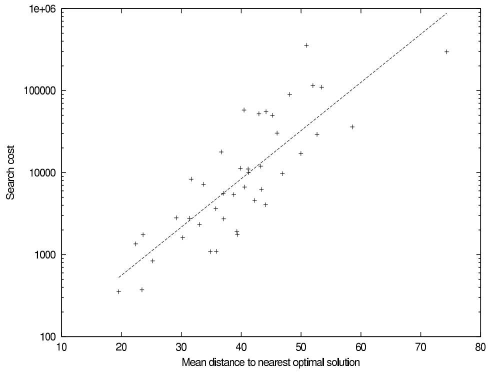

Figure 3: Scatter-plot of $d_{ijar\text{-}opt}$ versus search cost (c_{Q2}) for 10×10 random JSPs; the least-square fit line is super-imposed.

5 Modeling Algorithm Performance

Our results indicate that an effective mechanism for escaping attractor basins is sufficient to ensure high-performance ILS for the JSP. However, we know little about the global search strategy employed by *I-JAR*. We now consider two questions: "What search space features are responsible for the overall cost of search?" and "What are the high-level, qualitative run-time dynamics?" Specifically, we model the performance of *I-JAR*, as quantified by either the mean ($\overline{c}$) or median (c_{Q2}) number of iterations required to locate an optimal solution, depending on the context.

One approach to developing performance models of local search is to identify a *static* feature of the underlying search space that is highly correlated with search cost. Such static cost models typically take the form of a linear regression model, such that model accuracy can be quantified as the regression r^2, i.e., the proportion of the total variability in search cost accounted for by the model. For tabu search in the JSP, we recently showed that many well-known and widely search space features (e.g, the number of optimal solutions) are relatively weak indicators of problem difficulty [WBHW03]. In contrast, we showed that the mean distance between random local optima and the nearest optimal solution is highly correlated with local search cost; we denote the latter by $d_{lopt\text{-}opt}$.

Intuitively, $d_{lopt\text{-}opt}$ measures the *effective* (because it accounts for the number and distribution of optimal solutions) size of the sub-space containing the set of all local optima. Because search is restricted to the set of local optima, we would also expect $d_{lopt\text{-}opt}$ to be indicative of search cost in *I-JAR*. To test this hypothesis, we computed $d_{lopt\text{-}opt}$ for those 42 instances of our 10×10 problem set with $\leq 1{,}000{,}000$ *optimal* solutions; the computation is prohibitive for the remaining instances. A regression model of $d_{lopt\text{-}opt}$ versus $log_{10}(c_{Q2})$ yields an r^2 value of 0.2683. Although $d_{lopt\text{-}opt}$ only accounts for roughly 25% of the variability of search cost in *I-JAR*, other search space measures are even weaker indicators of problem difficulty.

Clearly, either (1) $d_{lopt\text{-}opt}$ is not an accurate indicator of the size of the local optima sub-space or (2) the size of the local optima sub-space is not completely indicative of search cost. Consider instead the distance to the nearest optimal solution (d_{opt}) for local optima visited by

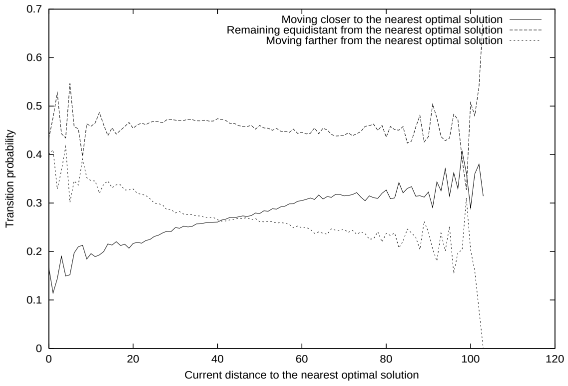

Figure 4: Aggregate transition probabilities for search under *I-JAR* for a typical 10×10 JSP.

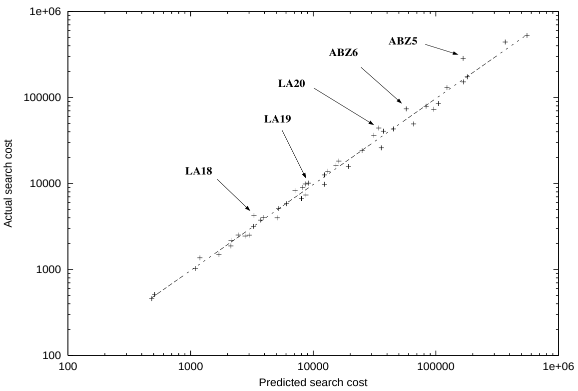

Figure 5: Predicted versus actual search costs for 10×10 random JSPs.

I-JAR during search. We show the distribution of d_{opt} for both random local optima and local optima visited by *I-JAR* for a typical 10×10 instance in Figure 2. The results indicate that random local optima are *not* representative of the local optima visited during search. This leads us to conjecture that the mean d_{opt} for local optima visited by *I-JAR* is a more accurate indicator of the size of the local optima sub-space than $d_{lopt\text{-}opt}$. To test this hypothesis, we computed the mean d_{opt} for local optima visited by *I-JAR* for those 42 10×10 instances with $\leq$ 1,000,00 optimal solutions, and denote the resulting value by $d_{ijar\text{-}opt}$. We estimate $d_{ijar\text{-}opt}$ from a set of solutions generated over a large number of independent trials. A regression model of $d_{ijar\text{-}opt}$ versus $log_{10}(c_{Q2})$ yields an r^2 value of 0.6859; the corresponding scatter-plot is shown in Figure 3. The resulting model yields a 250% increase in accuracy over the static $d_{lopt\text{-}opt}$ model, and accounts for roughly 2/3 of the variability in search cost.

It is still unclear *why* $d_{ijar\text{-}opt}$ is so highly correlated with search cost – which ultimately depends on the dynamic run-time behavior of *I-JAR*. We develop a Markov model of *I-JAR* by aggregating solutions based on their distance i from the nearest optimal solution. We define a set of states S_i, $2 \leq i \leq D$, such that a state S_i represents the set of all local optima that are distance i from the nearest optimal solution, and D represents the maximal distance from a local optimum to the nearest optimal solution. We denote the conditional probability of a transition from state S_j to state S_i by $P(S_i|S_j)$. The resulting Markov chain is a simple, generalized one-dimensional random walk, with a reflecting barrier at state S_D; we also impose $P(S_0|S_0) = 1$ to create an absorbing state S_0. We estimate both D and the set of transition probabilities $P(S_i|S_j)$ by analyzing the behavior of *I-JAR* over a large number of extended, independent trials. We identify a significant bias in the relative values of $\sum_{i=0}^{j-1} P(S_i|S_j)$ and $\sum_{i=j+1}^{D} P(S_i|S_j)$, as shown in Figure 4 for a typical 10×10 problem instance. Specifically, the search space induces a restoring force that consistently biases search *away* from optimal solutions, toward local optima that are roughly equidistant from the nearest optimal solution and solutions that are maximally distant from the nearest optimal solution. In [Wat03], we show that this bias is due to the choice of the underlying solution representation.

To validate our model, we simply compare the actual mean search cost $\overline{c}$ with that predicted by our Markov chain model. In Figure 5, we show a scatter-plot of the predicted versus actual $\overline{c}$ for the same subset of 42 10×10 instances; the r^2 value of the corresponding regression model is 0.9935. For reference, we also include results for those 10×10 benchmark instances with $\leq$ 1,000,000 optimal solutions. The accuracy of the model is exceptionally high, such that the actual $\overline{c}$ is always within a factor of 2 of the predicted $\overline{c}$. The model additionally accounts for the cost required to locate sub-optimal solutions to random JSPs. We also identify a relationship between the three performance models.

6 Additional Results and Conclusions

Our goal was to advance the state-of-the-art in the *analysis* of meta-heuristics. We showed that the attractor basins of local optima in the JSP are surprisingly weak. Based on this analysis, we introduced a new ILS algorithm, *I-JAR*, which can locate high-quality solutions to difficult benchmark instances. We demonstrated that the ILS meta-heuristic is competitive with tabu search on the JSP, and that complex, problem-specific large-step move operators are not necessary to achieve high-performance implementations of ILS. We concluded by introducing remarkably accurate performance models of *I-JAR*, which yield significant insights into both the dynamic run-time behavior of ILS and those search space features that influence problem difficulty. This research represents the first complete end-to-end design, evaluation, and analysis of any meta-heuristic, and yields the most accurate performance models of any meta-heuristic developed to date.

References

[LMS03] H.R. Lourenco, O. Martin, and T. Stuetzle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*. Kluwer, 2003.

[Lou95] H. Lourenco. Job-shop scheduling: Computational study of local search and large-step optimization methods. *European Journal of Operational Research*, 83:347–364, 1995.

[MBK99] D.C. Mattfeld, C. Bierwirth, and H. Kopfer. A search space analysis of the job shop scheduling problem. *Annals of Operations Research*, 86:441–453, 199.

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

[Tai94] E.D. Taillard. Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, 6(2):108–117, 1994.

[vLAL92] 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.

[Wat03] J.P. Watson. *Empirical Modeling and Analysis of Local Search Algorithms for the Job-Shop Scheduling Problem.* PhD thesis, Department of Computer Science, Colorado State University, 2003.

[WBHW03] J.P. Watson, J.C. Beck, A.E. Howe, and L.D. Whitley. Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence*, 143(2):189–217, 2003.