Adaptive Search Algorithms and Fitness-Distance Correlation*

J. Christopher Beck* Jean-Paul Watson†

*Cork Constraint Computation Center, Department of Computer Science
University College Cork, Cork, Ireland
{c.beck}@4c.ucc.ie

†Department of Computer Science, Colorado State University
Fort Collins, CO 80523-1873 USA
watsonj@cs.colostate.edu

1 Introduction

Several constructive heuristic search algorithms dynamically adapt their search strategy during execution by learning the relative importance or weight of solution components. We hypothesize that the performance of such techniques depends on the strength of the *fitness-distance correlation* (FDC) in the space of solutions. FDC is the correlation between the quality of a solution and its distance to an optimal solution. In problems with strong FDC, components in good solutions are likely to occur in optimal solutions. Thus, the learned weights will tend to bias the search toward optimal solutions. In problems with no FDC, the learned weights are essentially random, as there is no correlation between the presence of a particular solution component and its occurrence in an optimal solution. In a problem with negative FDC, good solutions share little with optimal solutions, causing adaptive search algorithms to learn weights that bias search *away* from optimal solutions. In this paper, we test our hypothesis in two problem domains: an idealized problem domain that allows complete control over the fitness-distance correlation and the job shop scheduling problem.

Researchers have hypothesized that *Fitness-Distance Correlation* (FDC)[BKM94] is correlated with problem difficulty for *local* search algorithms. To compute the FDC for a problem instance, a simple local search algorithm (e.g., steepest-descent) is used to generate N random local optima. The quality of a local optima i is denoted by $F(i)$ and the distance between i and the nearest globally optimal solution is denoted by $dist_i$. FDC for the instance is then defined as the Pearson's correlation between $dist_i$ and $F(i)$ in the N samples.

We consider the impact of FDC on the performance of two constructive, adaptive search algorithms: *Ant Colony Optimization* (ACO) and *Adaptive Probing* (AP). Consider a problem whose solution requires a value be assigned to each of N attributes a_i. Both ACO and AP

*This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075.

incrementally construct solutions by assigning values to the a_i; the process of building a complete solution is referred to as a *probe*. In ACO [DDCG99], value assignments are a function of both a learned weight and a pre-determined preference for each candidate value; the preference is dictated by a fixed external, problem-specific heuristic. Search proceeds by iteratively performing sets of k probes. After each probe set, the learned weights are updated based in part on the values of each a_i present in the best of the k resulting solutions. ACO has been applied to a number of combinatorial optimization problems including the traveling salesman [DDCG99] and quadratic assignment [SD99] problems, in addition to scheduling problems [MMS00].

AP [Rum02] differs from ACO in that the weights are updated after every probe, the weight updating procedure is more complex and statistically grounded, and no external heuristic is used (although one could be incorporated [Rum02]). In AP, the initial probes proceed by randomly and uniformly assigning values to the a_i; the process is terminated once all values of a variable have been assigned at least once. The results are then used to construct a set of preliminary weights, which are used to bias future probes. After the initial probing, weights are updated after each probe. The weight learning mechanism assumes that each decision in a probe is statistically independent and defines a "cost" for each decision such that the total cost is equal to the sum of individual decision costs. Weights are learned by observing the distribution of solution costs obtained by different probes and using the difference between the predicted and observed solution cost to estimate cost distributions for individual decisions. The estimates are then used to probabilistically bias decision-making while probing.

ACO and AP were chosen for two reasons: first, to our knowledge, no research has investigated the relationship between FDC and problem difficulty for constructive algorithms. Second, for such adaptive algorithms to be successful it seems clear that there must be some relationship between sub-optimal and optimal solutions. The adaptive algorithms require the problems to have some structure to learn from in order to eventually find an optimal solution. A priori, FDC appears to be a reasonable measurement of such structure.

2 Experiment 1: An Idealized Problem Domain

We first apply ACO and AP to an idealized problem domain that allows us to control for the expected FDC of problem instances. We specify solutions using vectors of n bits. To generate a problem instance, we identify a random set of bit vectors as optimal solutions; we denote the resulting set by S^*. Let $HD(s, S^*)$ denote the minimal Hamming distance from a solution s to an optimal solution $s^* \in S^*$. We define the cost of any $s^* \in S^*$ equal to 0, and set the cost, $c_{FDC+}(s)$, of a sub-optimal solution s in an instance with perfect FDC equal to $HD(s, S^*)/n$. Similarly, we set the cost $c_{FDC-}(s)$ of sub-optimal solutions in an instance with perfect negative FDC to be $(n - HD(s, S^*))/n$. To generate instances with intermediate FDC, we interpolate between these two extremes, such that the overall cost function for a solution in our idealized problem domain is given as follows:

$$c(s) = \begin{cases} 0 & \text{if } s = S^* \\ \alpha \times c_{FDC+}(s) + (1-\alpha) \times RAND(s) & \text{if } s \neq S^* \wedge \beta = 0 \\ \alpha \times c_{FDC-}(s) + (1-\alpha) \times RAND(s) & \text{if } s \neq S^* \wedge \beta = 1 \end{cases} \tag{1}$$

where $\alpha, \in [0,1]$, $\beta \in \{0,1\}$, and $RAND(s) \in [0,1]$; the latter value is generated by using the bit vector s as the random seed. The random noise component is added to achieve more realism in our model. Clearly, when $\alpha = 0$, the cost assignments to sub-optimal solutions is purely random. While α determines the strength of the FDC, β is a two-valued parameter governing its direction: $\beta = 0$ and $\beta = 1$ induce positive and negative FDC, respectively.

We consider problem instances with $n = 20$ and $|S^*| = 3$; results with $|S^*| = 10$, 100, and 1000 are qualitatively similar. Problem sets containing 10,000 instances apiece were generated for $\alpha \in [0,1]$ in steps of 0.1 and for $\beta \in \{0,1\}$, resulting in a total of 21 problems sets (the sets for $\alpha = 0, \beta = 1$ and $\alpha = 0, \beta = 0$ are identical). For ACO, we let $k = 5$. Following standard practice [DDCG99], the value for bit i is assigned to 1 with probability $p_i(1)$ given by:

$$p_i(1) = \frac{[w_i]^\gamma [hv_i]^\delta}{[w_i]^\gamma + (1-w_i)^\gamma + [hv_i]^\delta + (1-hv_i)^\delta} \tag{2}$$

where γ and δ are constant parameters, hv_i is the external heuristic preference for assigning bit i the value of 1, and w_i is the learned weight for assigning bit i equal to 1; we let $\gamma = \delta = 1$. We set hv_i equal to the fraction of optimal solutions in which bit i is assigned a 1 with probability h; otherwise, we set hv_i equal to a random value sampled from the unit interval. Intuitively, h is an independent variable corresponding to the strength of the external heuristic.

The w_i for each bit i are initially set to 0.5. After each set of k probes, the w_i are updated as follows. First, all weights are decreased according to $w_i = (1-\rho) \times w_i$, where ρ is the constant learning rate; we let $\rho = 0.01$. Following Merkle et al. [MMS00], we then increase the weights for bit values appearing in the best solution generated in the last probe set according to $w_i = w_i + \rho \times c^{-1}$, where c is the cost of the best solution. The same rule is also applied for the best solution found in any set of k probes. Finally, after each set of probes, the best solution found so far is replaced by the best solution in the current probe set with probability 0.01, independent of relative costs. Such a replacement strategy helps to prevent premature convergence of the learned weights on high-quality sub-optimal solutions [MMS00].

Because AP uses an additive weight model, after the initial probing phase the value selected for bit i is based on the probability that the value's contribution to the overall weight (i.e., solution cost) is less than that of the other value. The algorithm maintains estimates of the mean and standard deviation of the local weights for each value. After initial probing, weights are updated according to: $w_i = w_i + \rho \times (c(s) - c_{est}(s))/n$, where ρ specifies the learning rate (which we take as 0.2 in our experiments, following Ruml [Rum02]), $c(s)$ is the actual cost of solution s, $c_{est}(s)$ is the estimated cost of the solution s, and n is the number of bits.

We execute both ACO and AP for a total of 25,000 probes on each problem instance, and report results relative to two baseline algorithms. For ACO, we measure the mean relative error between the number of problems solved (to optimality) versus the number solved by a non-adaptive heuristic probing algorithm (HP); the latter simply uses the external heuristic preference vectors hv_i generated with the corresponding h value to bias the probing. Because AP uses no external heuristic, the appropriate non-adaptive baseline is random probing (RP). The resulting relative performance of both ACO and AP is shown in Figure 1; points above $y = 0$ indicate superior performance by ACO (AP) relative to HP (RP). The mean FDC of instances in each problem set are shown on the x-axis, generated by sampling 25,000 solutions for each

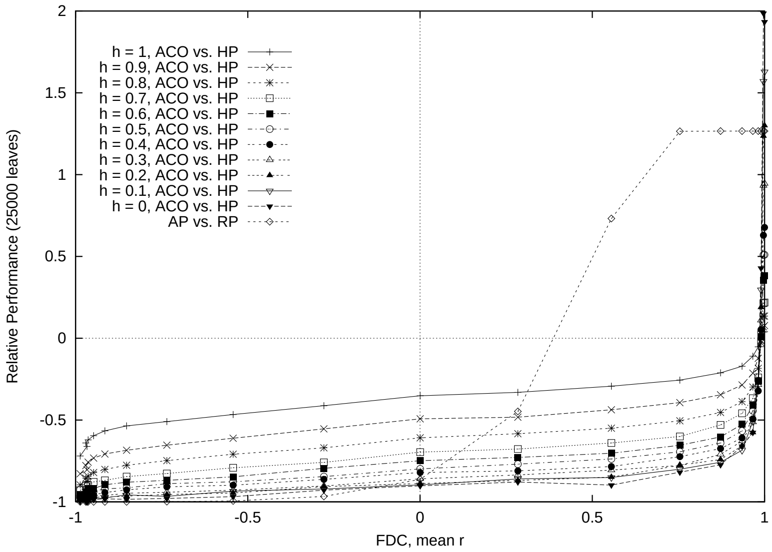

Figure 1: The relative performance of ACO vs. heuristic probing and AP vs. random probing after 25,000 probes as both FDC and heuristic quality are varied.

problem instance; we can only control for the *expected* FDC with our problem generator.

AP provides strong performance down to FDC ≈ 0.4, consistently outperforming random probing. For FDC ≤ 0.4, both relative and absolute performance are poor. We show ACO results across a range of h. Overall, strong FDC leads to strong performance and moderate or negative FDC leads to poor performance, and a stronger external heuristic generally yields stronger performance. These results support our hypothesis as to the importance of FDC in the performance of ACO and AP. With no or negative FDC, both algorithms perform very poorly, solving no problems. With a high FDC, performance of both algorithms is very good.

3 Experiment 2: Job Shop Scheduling

We now turn from our idealized problem domain to a more realistic problem domain: job shop scheduling (JSP). We report only results for ACO, as the results for AP are very similar. We use a set of 1,000 random 6×6 instances to analyze the relationship between FDC and search cost for ACO on the JSP. We define the distance between any two solutions in the JSP as the disjunctive graph distance [MBK99], and use a constraint programming algorithm [BF00] to enumerate sets of optimal solutions. The choice of distance measure has the potential to impact our results. The disjunctive graph measure is the most widely used distance measure for the JSP; Nowicki and Smutnicki [NS03] discuss other possibilities.

The performance of all ACO implementations for the JSP reported to date is relatively weak in comparison to other well-known meta-heuristics, e.g., tabu search. Consequently, we adapt a high-performance ACO algorithm introduced by Merkle et al. [MMS00] for the Resource-Constrained Project Scheduling Problem, of which the JSP is a specialization; we denote the resulting algorithm by *ACO-MMS*. In pilot experiments, we found that while *ACO-MMS*

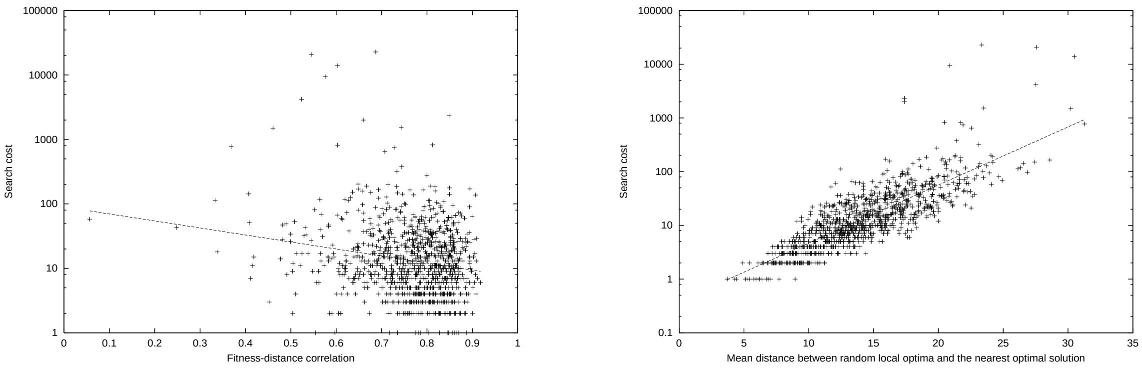

Figure 2: Scatter-plots FDC (left) and $d_{lopt\text{-}opt}$ (right) vs. search cost for 6×6 JSPs.

outperforms other reported ACO implementations for the JSP (specifically [vdZM99] and [CDMT94]), the performance was still poor. In response, we implemented a hybrid version of *ACO-MMS* in which solutions resulting from each probe are post-processed by via steepest-descent local search under a critical-path neighborhood [MBK99]. The hybrid algorithm, which we denote *ACO-MMS-LSO*, significantly outperforms the original algorithm, and is able to locate optimal solutions to all of our 6×6 instances in reasonable run-times. In our experiments, we let $k = 10$ and $\rho = 0.01$. Weights are updated after each set of k probes using both the best solution obtained in the current round and the overall best found so far. The best-so-far solution is replaced with the best solution of the current round with probability $p = 0.01$ [MMS00].

We define search cost under *ACO-MMS-LSO* as the mean number of k probe sets required to locate an optimal solution, with statistics taken over 100 independent trials; we denote the result by c_{Q2}. To test the hypothesis that FDC is indicative of problem difficulty, we measured the correlation between FDC and $log_{10}(c_{Q2})$ for our 6×6 instances; the corresponding scatter-plot is shown in the left side of Figure 2. The resulting r-value is only -0.1893, providing strong evidence against our hypothesis that problem difficulty is correlated with FDC.

4 Discussion and Future Work

A conflict exists between our results on the idealized problem domain and the JSP. We currently have no explanation for this divergence, although we do posit two conjectures. First, the ACO algorithm for the JSP uses local search to post-process the solutions obtained by probing; the modification was necessary to achieve reasonable performance on the test problems. However, the use of locally optimal solutions to update the weights may significantly impact the search space features relevant to algorithm performance. Second, inherent differences between the idealized problem and the JSP may lead to variable effects of FDC on algorithm performance. The obvious candidate is the presence of constraints in the JSP, as not every assignment of values to solution attributes results in a feasible solution. The existence of such constraints is obviously important to problem difficulty and may induce search space features that overwhelm any effect of FDC on algorithm performance.

We previously demonstrated that the mean distance between random local optima and the nearest optimal solution, which we denote $d_{lopt\text{-}opt}$, is highly correlated with problem difficulty for *local* search algorithms in the JSP [WBHW03]. Unexpectedly, $d_{lopt\text{-}opt}$ is also highly correlated with search cost for *ACO-MMS-LSO*. We show a scatter-plot of $d_{lopt\text{-}opt}$ versus $log_{10}(c_{Q2})$ in the right side of Figure 2; the corresponding r-value is 0.8154. This result is the first demonstration of such a correlation with an algorithm that is not based on local search. However, it should be noted that the hill-climbing component of *ACO-MMS-LSO* is a local search technique and therefore might be impacting our results. Clearly, further investigation is warranted.

References

[BF00] J.C. Beck and M.S. Fox. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. *Artificial Intelligence*, 117(1):31–81, 2000.

[BKM94] K.D. Boese, A.B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16:101–113, 1994.

[CDMT94] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Comp. Science*, 34(1):39–53, 1994.

[DDCG99] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[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.

[MMS00] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 893–900, 2000.

[NS03] E. Nowicki and C. Smutnicki. Some new ideas in TS for job-shop scheduling. In *Adaptive Memory and Evolution: Tabu Search and Scatter Search*. Kluwer, 2003.

[Rum02] W. Ruml. *Adaptive tree search*. PhD thesis, Harvard University, 2002.

[SD99] T. Stuetzle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D Corne et al., editor, *New Ideas in Optimization*. McGraw-Hill, 1999.

[vdZM99] S. van der Zwaan and C. Marques. Ant colony optimisation for job shop scheduling. In *Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life*, 1999.

[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.