Estimating the Hardness of Optimisation

Sylvie Thiébaux[1], **John Slaney**[2] and **Phil Kilby**[1]

Abstract. Estimating computational cost is a fundamental issue in time-bounded computation. We present a method for estimating the hardness of optimisation problems (find a minimal cost solution to instance I) by observing that of the corresponding decision problems (has I a solution of cost less than threshold T). Provided T is not too close to the optimal, decision is typically much easier than optimisation, yet knowing the hardness of the former is useful for predicting the latter. The present paper reports an experimental investigation of this idea, with encouraging results. An investment of a few percent of the work required for optimisation suffices for estimation within a small factor, even using a very simple implementation of the method.

1 INTRODUCTION

Many combinatorial problems found in scheduling, network design, planning and the like have both an *optimisation* form (find a minimal cost solution) and a corresponding *decision* form (is there a solution of cost less than threshold T). The close relationship between these two forms of the problems is explored in [4] and [13], with an emphasis on their respective computational cost or *hardness*. As illustrated in Figure 1, which we borrow from [13], the hardness of decisions is polynomial in the hardness of optimisation and exponential in the distance separating the decision threshold T from the cost of the optimal solution. This has been observed in many domains, and tends to hold not only on average over problem instances as shown in the figure, but also for individual instances.

In [13] this relationship is exploited to obtain an estimate of the hardness of decisions using an estimate of the hardness of optimisation. However, the converse idea of observing decision problems in order to estimate the hardness of finding an optimal solution has never been investigated. We see two good reasons why it ought to be. Firstly, it appears to be more useful, since a good prediction of the hardness of optimisation is a key issue in time-bounded reasoning. Secondly, the decision problems tend to be much easier than the optimisation ones, and it makes better sense to use the easy to estimate the hard than *vice versa*.

To illustrate how this could work, we begin with an example. Figure 2, which is something like a slice through Figure 1, shows the hardness of computing decisions for a *single* typical problem instance taken from the same blocks world domain as used for Figure 1. The x axis is the decision threshold T and the y axis the hardness of deciding whether there is a plan shorter than T. The measure of hardness is the number of branches explored by the domain-specific solver described in [13]. For each threshold, the solver was rerun 8 times with random tie-breaking, resulting in the observed spread

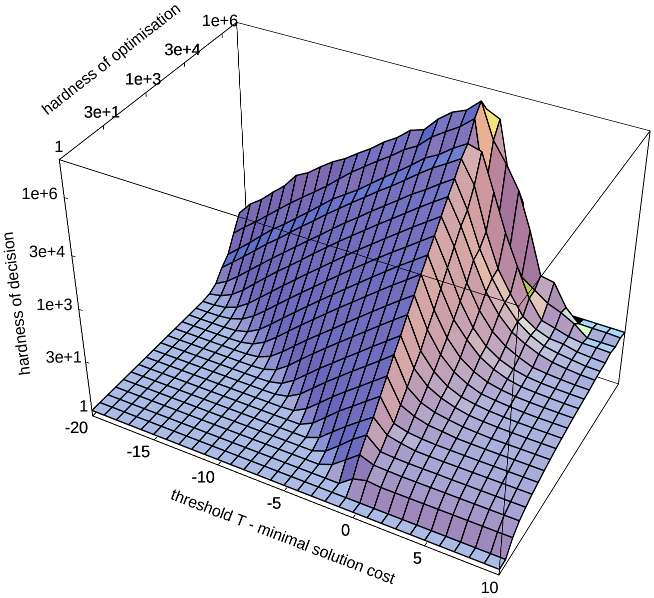

Figure 1. Average hardness of decision plotted against difference between threshold T and optimal solution cost, and hardness of optimisation. 6000 instances of a blocks world planning domain were used to produce the figure.

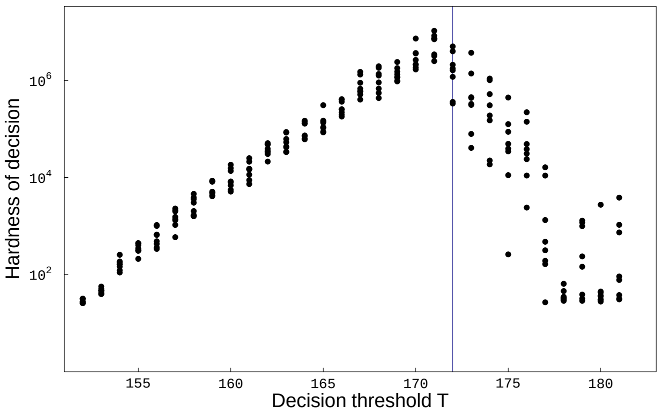

Figure 2. Hardness of decisions for a single blocks world planning instance, plotted against the decision threshold (required plan length).

of data points. The vertical line indicates the optimal (shortest) plan length, which happens to be 172 moves in this case. To the left of the vertical line, the decisions are negative and become exponentially harder to derive as the threshold approaches the optimal. To the right, the search produces a plan and again the hardness increases exponentially with closeness to the optimum. The variance is much greater on the right than on the left, for much the same reason that

[1] CSIRO Math. & Info. Sciences, PO Box 664, Canberra, ACT 2601, Australia. e-mail: First.Last@cmis.csiro.au
[2] Computer Science Laboratory, Australian National University, Canberra, ACT 0200, Australia. e-mail: John.Slaney@anu.edu.au

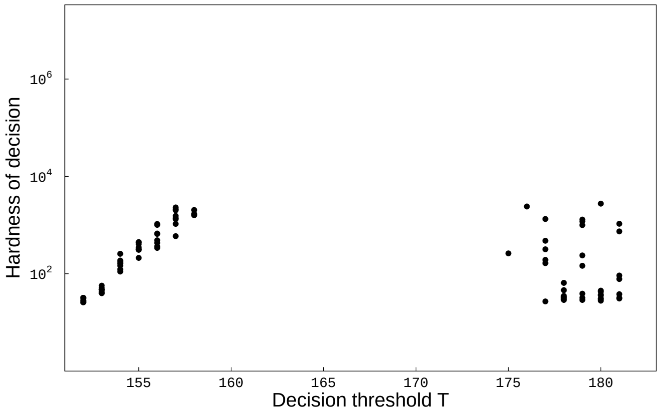

Figure 3. A basis for estimation? Data points from Figure 2 obtainable within a limit of 3000 backtracks per search.

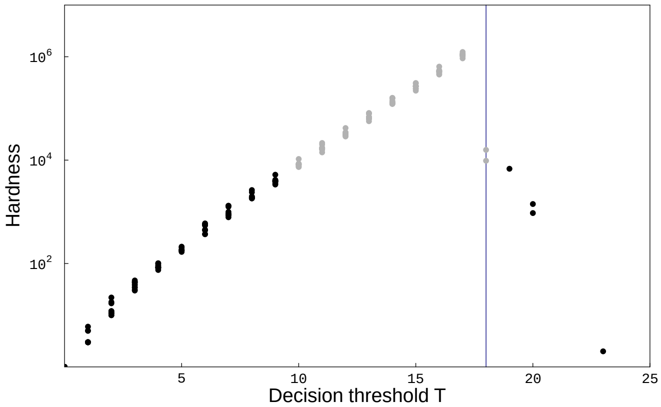

Figure 4. Another example: hardness of decisions for a single instance of minimum ones. The black points are those available cheaply.

random restart is effective on underconstrained problems but not on overconstrained ones: there is randomness in how much of the search tree will be traversed before the first solution is found. As noted in [4], the hardness of optimisation—i.e. of computing an optimal solution and proving it optimal—is approximately that of solving the two decision problems with threshold at either side of the optimal (here with $T = 171$ and $T = 172$).

Briefly, our approach is to estimate the position of the peak in Figure 2 by observing the data points in the low part of the curve. Figure 3 shows those data points from Figure 2 obtainable very cheaply—in three orders of magnitude less time than is required for optimisation. A natural idea is to estimate the curve of hardness for the negative decisions and that for the positive decisions, extrapolate both hardness curves and note the point where they cross. The x coordinate of this point gives an estimate of the optimum solution and the y coordinate an estimate of the hardness of computing it. Indeed, it is not hard to arrive at a hardness curve on the negative side: the linear regression fitted to the logs of the negative observations is a good start. On the positive side, however, curve fitting is so unreliable as to be almost useless because of the extremely large variance. Instead, therefore, we use a function learned from previous experience with the domain (blocks world in this example) to estimate the amount by which the best observed solution (175 in this case) exceeds the optimum. The hardness curve extrapolated to the point where T reaches the estimated optimum value gives a reasonably good estimate of the height of the peak. More exactly, as already noted, the hardness of optimisation is close to the sum of the hardness of decisions for $T = \text{optimal}$ and $T = \text{optimal} - 1$.

1. Find an underestimate LB and an overestimate UB of the optimal solution cost.
2. While global resources remain do
   (a) Choose some threshold T in the range $LB \ldots UB$.
   (b) Set a local resource bound B.
   (c) Attempt to solve the decision problem with threshold T within bound B and record the hardness if successful.
   (d) Extrapolate a curve $N(x)$ from the accumulated data on negative decisions.
   (e) Estimate the solution cost o of the optimal solution on the basis of the accumulated data on positive decisions.
   (f) Estimate the hardness of optimisation on the basis of the predicted peak $N(o-1)$.

Figure 5. The estimation algorithm

To demonstrate that this idea is by no means confined to blocks world or even to planning, Figure 4 shows another example taken from minimum ones, a logic domain.

2 THE METHOD

The above examples are suggestive, but examples do not make an algorithm. In fact, we shall not present a specific algorithm, but rather a method or algorithm scheme which must be instantiated to the case of each particular problem domain to which it is applied. We shall subsequently examine the performance of three instantiations of the general method, designed for different domains: blocks world planning (BW), the travelling salesman problem (TSP) and minimum ones (MIN-ONES).

Overall, the generic algorithm, outlined in Figure 5, is an anytime procedure which runs until a global resource bound is reached. This could be a fixed time limit, or simply the user's patience, but in practice we find it most useful to impose both a static limit and also a cutoff when the time already spent in estimation exceeds some proportion of the estimated hardness of optimisation. This acts as a rough classifier of whether the instance is too easy to be worth (further) estimation.

Several comments are in order before we proceed to examine instances of the method. Firstly, there are optimisation problems for which step 1 is intractable.[3] For example, in general propositional STRIPS planning, even deciding whether an upper bound exists is PSPACE-hard. For such problems, of course, our method does not always work. However, in a wide range of cases, there exist computationally inexpensive procedures which deliver suboptimal solutions suitable for setting UB. As for LB, even in the worst case it is always possible to choose 0 as a value. Hence the requirement that step 1 be performed quickly does not impose a serious limitation in practice.

Secondly, the template does not specify how the threshold T and the local resource bound B are to be selected. Naturally, the selection should be designed to yield new information, though this does not preclude choosing the same threshold several times which may

[3] It may even be impossible, in undecidable cases, as for example where the optimisation problem is to find the shortest resolution proof of an arbitrary set of first-order clauses.

prove valuable if the search method uses random tie-breaking. In the experiments reported below, we did not reason deeply about these settings, but evidently much more reasoning is possible. For example, background knowledge from previous experiments can be used to maximise the expected utility of these choices in the manner of [7].

Thirdly, the calculation of the curve of expected hardness of negative decisions and the use of the given information about positive decisions to arrive at an estimate of the optimum can be more or less sophisticated. A reasonable default for the former is to use a linear least-squares fit to logarithms of the observed values, though clearly it makes sense to give higher values more weight than lower ones for this purpose. Again, experience with a given domain may be used to learn some adjustment of the linear projection. As for estimating the optimum, in this paper we are not concerned to investigate that issue intensively; in our experiments, we simply learn the (average) amount by which the best known positive solution is expected to exceed the optimal. That is not to deny the importance of more refined techniques, starting with learning a probability distribution for the expected optimum rather than just an average value. However, we present our experimental results as compelling evidence that even crude techniques are good enough to yield excellent predictions across a range of domains.

3 EXPERIMENTS

In order to investigate the effectiveness of our approach, we conducted experiments on randomly generated instances of three problems: BW, TSP and MIN-ONES. The decision problems for these show a marked easy-hard-easy pattern as in Figure 2, resulting in rather good predictions especially for hard instances.

In our experiments, we wish to compare the estimated hardness of optimisation with the actual hardness. We therefore generated random instances of sufficient size for some of them to be difficult, but small enough for optimal solutions to be obtained for the comparison.

As a performance measure, we report the *proportional error* of the estimate as a function of the time at which the estimation is interrupted. For the purpose of comparing actual and predicted hardness, proportional error is defined as the ratio of the larger of these two figures to the smaller, and time is defined as the number of branches the estimator has explored, as a percentage of the number explored by the optimal solver.

We give results for the hardest 10% of problem instances among those generated, since in practice these are the ones whose hardness is worth predicting. Because our estimator aborts an instance when it decides that it has spent too much time on it in proportion to the cost of optimisation, seriously incorrect estimates are possible if a hard instance gets misclassified as trivial in the first few iterations of the estimation loop, due to unexpectedly low values for UB and consequent initial under-estimation of the optimal solution cost. We additionally report the number of instances so misclassified.

3.1 Blocks World (BW)

For these experiments, we consider the most basic version of BW planning in which both the initial state and the goal state are completely specified and in which the table has infinite capacity. The solution cost is the number of steps (move block x from y to z) in the plan. Optimal BW planning is NP-hard [6] though approximation within a factor of 2 is possible in linear time [12] and the average case performance of the approximation algorithms is much better than their worst case.

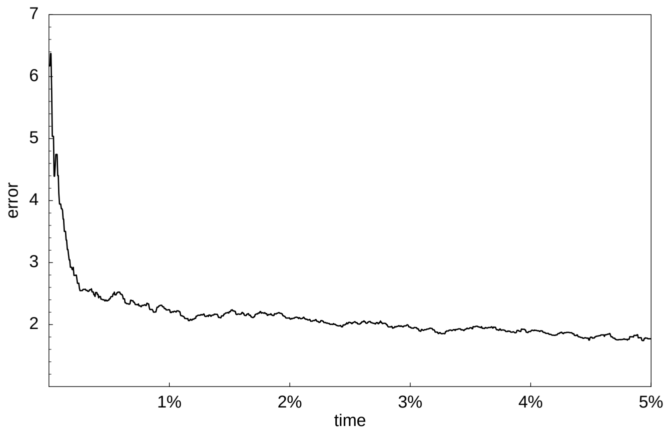

Figure 6. Average proportional error of estimates on hard BW instances with 125 blocks, plotted against time invested in estimation.

Our sample consisted of 1000 BW problem instances with 125 blocks, randomly generated with uniform distribution [12]. To solve them optimally, we used the iterative deepening algorithm outlined in [13]. For easy negative cases of the decision problem, we used a similar iterative deepening search with the decision threshold as a hard bound on the depth of the search tree. For positive cases, however, this method is inappropriate as it would amount to solving the (hard) optimisation problem as a step towards reaching the easy positive decisions. On the positive side, therefore, we ran a depth-first search with the specific decision threshold as its depth bound, without iterative deepening.

The upper bound UB of the window $(LB \ldots UB)$ was obtained using the linear time approximation algorithm GN2 [12], which overestimates the plan lengths by less than 10% on average. For LB we used the number of misplaced blocks plus the number of singleton deadlocks (see [12] for definitions) which gives a fairly tight lower bound for problems of the size considered and is computable in quadratic time.

The local resource bound B was set initially to a low figure (100 branches) and increased by 200 after each pair of iterations. T was chosen alternately to be one greater than the highest threshold so far giving a negative decision, and one less than the lowest giving a positive one. The iterative deepening method used for negative decisions repeats those for thresholds $\langle LB, \ldots, T-1\rangle$ on the way to T; because it uses random tie-breaking, it yields new measures of hardness for thresholds less than T, providing useful extra data points, as already illustrated in Figure 2.

Figure 6 shows the average error of predictions made by our anytime algorithm as a function of the time for the 10% hardest instances. All of these instances required the optimal solver to backtrack over a million times. It is clear that the average prediction very quickly settles down to be within a factor of about 2 of the actual hardness. This occurs when the backtracks used for prediction amount to some 1% of those needed for optimisation. On easier instances, the time required for the prediction to converge on a figure is less absolutely, but proportionally more of the optimal solution time, and the prediction is similarly better in absolute terms but proportionally less good.

None of the 10% hardest instances were misclassified by the algorithm. Of 418 instances requiring over 100,000 backtracks, only 16 were mis-classified.

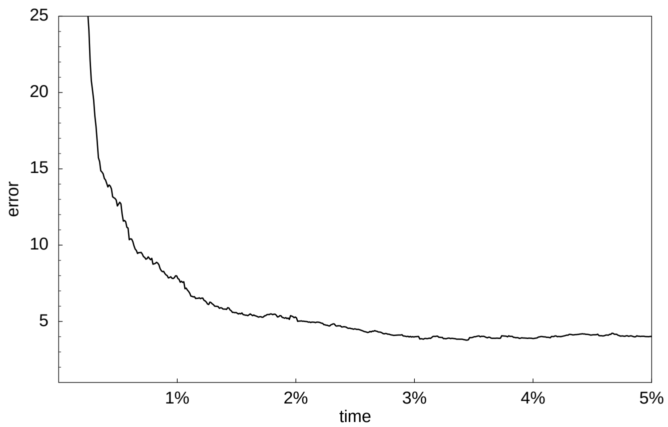

Figure 7. Average proportional error of estimates on hard geometric TSP instances with 20 cities, plotted against time invested in estimation.

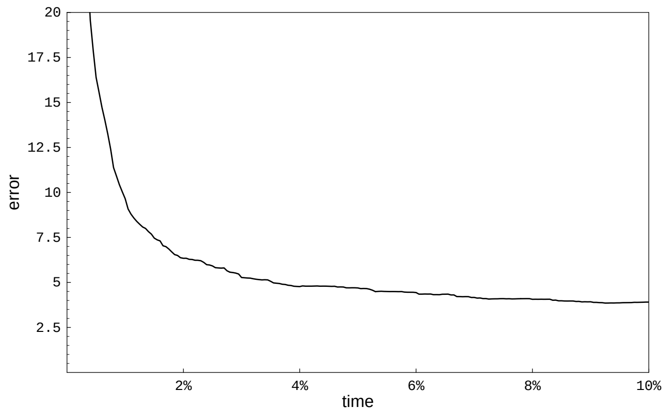

Figure 8. Average proportional error of estimates on hard MIN-ONES instances with 100 variables, plotted against time invested in estimation.

3.2 Travelling Salesman Problem (TSP)

The problem here is to find a minimal-cost Hamiltonian circuit in a complete undirected graph with edge weights. In the version of the problem considered in this paper, the points are chosen to lie on a plane and the edge weights are taken to be the distances between them in Euclidean space. There is a large literature on this and closely related problems, from both the AI and OR communities.[4]

As in the case of BW, the optimisation problem is NP-hard. With arbitrary edge weights, TSP is not approximable within a constant in polynomial time, though with the addition of the triangle inequality it is approximable within a factor of 3/2 and with Euclidean distances as the weights, as we use here, it admits a polynomial time approximation scheme [1]. Interestingly, the minimal solution costs for randomly selected TSP instances on a square cluster very strongly around a known mean as the number of cities becomes large [3]. This means that for large numbers of cities, the length of the optimal tour is highly predictable, though the hardness of optimisation is not.

We generated 1000 problem instances consisting of 20 points randomly placed on a square. We give results for uniform distribution, though we also tried some degree of clustering of the points and found that the technique still works, albeit with some degradation in the quality of the estimation. As a solver we used a branch and bound algorithm with the well-known "regret" heuristic. The first solution found, on the leftmost branch of the search tree, was treated both as UB and as the estimated optimum,[5] so all of the subsequent work went into gathering information concerning negative decisions. For LB we took $UB/10$, which is a low underestimate and is always greatly improved on within a few iterations of the anytime algorithm.

As in the case of BW, we increased the local resource bound B arithmetically between iterations. We used the predicted hardness of decisions to select as the next decision threshold the highest value likely to yield a negative decision within the limit B. This selection was made using a least squares fit on the highest 50% of solution costs for which hardness data were available.

Figure 7 shows the average error of predictions made by our algorithm as a function of the time for the 10% hardest instances. These required at least 400,000 node expansions for their optimal solution. None of them were misclassified by the algorithm as too easy to be worth predicting.

[4] See [11] for an overview of the field.

[5] This simplifying assumption may appear rather drastic, but as a matter of fact, with only 20 cities, it gives tolerably accurate results and is therefore legitimate in this context.

3.3 Minimum Ones (MIN-ONES)

MIN-ONES is the problem of finding a model of a 3-SAT instance with the minimum number of variables assigned to true. This problem is not approximable in polynomial time within $n^{1-\epsilon}$ for any $\epsilon > 0$, where n is the number of variables [9]. The decision version of MIN-ONES is a special case of DISTANCE-SAT [2], which consists in finding the model of a 3-SAT instance disagreeing with a given partial interpretation on at most T variables. These decision problems are harder than SAT, in that tractable restrictions of SAT (e.g. Horn, Krom) do not give rise to tractable restrictions for DISTANCE-SAT [2].

Assuming the fixed clause length model of random 3-SAT, we found a clause to variable ratio of 1.8 to give the hardest MIN-ONES instances on average (this is consistent with the observations in [2]). Note that this ratio is much lower than that associated with the well-known phase transition in solvability for 3-SAT, meaning that these instances almost certainly have many solutions. For our experiments, we therefore generated 1000 problem instances with 100 variables and 180 clauses. To solve them optimally, we used an extension of the Davis-Putnam algorithm similar to those in [2], but with the simple MOMS branching heuristic in place of the more sophisticated ones considered there.

LB was set to 0. This is an extreme underestimate, of course, but with such a low bound the decision problem is so easy that little time is lost by it. To obtain UB, we generated an arbitrary model of the SAT instance using the Davis-Putnam algorithm. Although finding such a model is NP-hard in the worst case, the present SAT instances are so small and underconstrained, that finding solutions is trivial in practice. The local resource bound B was initialised to the resource spent in finding UB and increased geometrically between iterations. The successive values of the decision threshold T were determined much as in the case of BW.

Figure 8 shows the average error of predictions made by our algorithm as a function of the time for the 10% hardest instances. Solving them optimally required between 1.5 and 10 million backtracks. Again, none of them were misclassified by the algorithm as too easy to be worth predicting. An important difference between MIN-ONES and the two previous problems is that the optimal solution size is harder to predict accurately. For that reason, the algorithm has to invest more time before the proportional error gets reduced to a figure comparable to those in the other examples. We expect the percentage of time required to achieve an acceptable estimate would decrease significantly with larger problems.

4 CONCLUSION

We have presented a method for estimating the computational cost of combinatorial optimisation problems, by relating the hardness of optimisation to that of the corresponding decision problems and by observing easy cases of the latter with the decision threshold some distance away from the optimal. As the examples show, this method has to be fine-tuned to suit each particular problem and the optimisation and decision algorithms used. With that caveat, however, it is clear that most encouraging results are attainable. We have shown that investing 5%, or even 1%, of the computation time in examining easy decision problems can result in estimates accurate to within much less than an order of magnitude. Moreover, both the accuracy and the proportion of the time required tend to improve as the problem size increases.

It must be stressed that the implementations used for the experiments reported here are by no means highly engineered. One of the most interesting future developments of our work would be to pursue the suggestion in section 2 above of generating probability distributions both for the estimated optimal solution and for the estimated computational cost of proving it optimal. This would decrease the importance of errors in the former, at the very least rendering the method more robust.

Naturally, our method will not suit all optimisation problems and algorithms. For instance, if there is little variance in hardness between instances of the same size, as is the case for maximum satisfiability (MAX-SAT), simple predictors may provide better results. Also, the method is inappropriate where overconstrained decisions are just as hard as optimisation. A typical case is number partitioning (NPP) with the state of the art CKK algorithm [10]: when the decision threshold is below the optimal partition size, CKK does not prune the search tree significantly more than when optimising [5].

There is a large literature from the last ten years on detecting hard instances of intractable problems. It has become commonplace to associate hardness with the boundaries at which transitions occur from under-constrained to over-constrained instances [8]. It is, however, equally well known that there are large differences in hardness between instances, even where the positioning of the instances relative to the phase transition is the same. Even worse, the variance of hardness is largest in the transitional region, where the really hard problems are. This means that while phase transitions furnish some information about the hardness of instances *on average*, they reveal very little about the hardness of any *particular* instance. Moreover, it has proved difficult to transfer results about average hardness in purely random instances to sets of instances with structure such as might occur in real-world domains. Our method, by contrast, takes each specific problem instance as it stands, largely independently of any other instances there may be. For that reason, it can accommodate highly structured problems relatively easily, as well as being much more informative in each particular case than techniques based on average behaviour.

As hinted in [4] each instance of an optimisation problem has associated with it a series of decision problems ranging from under- to over-constrained, with the critically constrained ones on the boundary where the solution is optimal. Thus each such problem instance generates in effect a little phase transition of its own, whose properties we are measuring by sampling in the regions where the decision problems begin to approach the boundary. It turns out — we present our results as evidence — that for the purposes of predicting the peak of the hardness curve for an instance, a little sampling goes a long way.

REFERENCES

[1] S. Arora, 'Polynomial time approximation scheme for euclidean TSP and other geometric problems', in *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS-96)*, pp. 2–11, Burlington (VE), (October 1996). IEEE Computer Society.

[2] O. Bailleux and P. Marquis, 'DISTANCE-SAT: complexity and algorithms', in *Proceedings of the 16th American Conference on Artificial Intelligence (AAAI-99)*, pp. 642–647, Orlando (FL), (july 1999). AAAI Press/The MIT Press.

[3] J. Beardwood, J. Hamilton, and J. Hammersley, 'The shortest path through many points', *Proceedings of the Cambridge Philosophical Society*, **55**, 299–327, (1959).

[4] I. Gent and T. Walsh, 'The TSP phase transition', *Journal of Artificial Intelligence*, **88**(1-2), 349–358, (December 1996).

[5] I. Gent and T. Walsh, 'Phase transitions and anealed theories: Number partitioning as a case study', in *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, ed., W. Wahlster, pp. 170–174, Budapest (Hungary), (august 1996). J. Wiley.

[6] N. Gupta and D. S. Nau, 'On the complexity of blocks world planning', *Journal of Artificial Intelligence*, **56**(2-3), 223–254, (August 1992).

[7] E. Hansen and S. Zilberstein, 'Monitoring the progress of anytime problem-solving', in *Proceedings of the 13th American Conference on Artificial Intelligence (AAAI-96)*, pp. 1229–1234, Portland (OR), (August 1996). AAAI Press/The MIT Press.

[8] T. Hogg, B.A. Huberman, and C.P. Williams, 'Phase transitions and the search problem', *Journal of Artificial Intelligence*, **81**(1-2), 1–15, (March 1996).

[9] P. Jonsson, 'Near-optimal nonapproximability results for some NPO PB-complete problems', *Information Processing Letters*, **68**(5), 249–253, (December 1998).

[10] R. Korf, 'From approximate to optimal solutions: A case study of number partitioning', in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 266–272, Montréal (Canada), (1995). Morgan Kaufmann.

[11] E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, J. Wiley, Chichester, 1985.

[12] J. Slaney and S. Thiébaux, 'Linear time near-optimal planning in the blocks world', in *Proceedings of 13th American Conference on Artificial Intelligence (AAAI-96)*, pp. 1208–1214, Portland (OR), (august 1996). AAAI Press/The MIT Press.

[13] J. Slaney and S. Thiébaux, 'On the hardness of decision and optimisation problems', in *Proceedings of the European Conference on Artificial Intelligence (ECAI-98)*, ed., H. Prade, pp. 244–248, Brighton (UK), (august 1998). J. Wiley.