ELSEVIER

Artificial Intelligence 81 (1996) 223-239

Artificial Intelligence

A study of complexity transitions on the asymmetric traveling salesman problem *

Weixiong Zhang [a,*], Richard E. Korf [b,1]

[a] *Information Sciences Institute and Computer Science Department, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292, USA*

[b] *Computer Science Department, University of California, Los Angeles, Los Angeles, CA 90024, USA*

Received May 1994; revised April 1995

Abstract

The traveling salesman problem (TSP) is one of the best-known combinatorial optimization problems. Branch-and-bound (BnB) is the best method for finding an optimal solution of the TSP. Previous research has shown that there exists a transition in the average computational complexity of BnB on random trees. We show experimentally that when the intercity distances of the asymmetric TSP are drawn uniformly from $\{0, 1, 2, \ldots, r\}$, the complexity of BnB experiences an easy–hard transition as r increases. We also observe easy–hard–easy complexity transitions when asymmetric intercity distances are chosen from a log-normal distribution. This transition pattern is similar to one previously observed on the symmetric TSP. We then explain these different transition patterns by showing that the control parameter that determines the complexity is the number of distinct intercity distances.

Keywords: Traveling salesman problem; Phase transitions; Problem solving; Combinatorial optimization; Complexity; Search; Branch and bound

1. Introduction

A phase transition of a complex system is a dramatic change of some system property when an order or control parameter crosses a critical value [29]. A simple example

* This research was supported by NSF Grant, IRI-9119825, a grant from Rockwell International, a GTE graduate fellowship (1992-93), and UCLA Chancellor's Dissertation Year Fellowship (1993-94).

* Corresponding author. E-mail: zhang@isi.edu. URL: http://isi.edu/isd/zhang. Some of this research was performed when this author was at the Computer Science Department, University of California, Los Angeles, CA 90024, USA.

[1] E-mail: korf@cs.ucla.edu.

SSDI 0004-3702(95)00054-2

of a phase transition is water changing from a liquid to a solid when the temperature drops below the freezing point. In computer science, for example, the probability that a random graph is connected, or contains a Hamiltonian circuit, increases sharply when the average graph connectivity exceeds a certain value [3]. In artificial intelligence, dramatic computational complexity transitions have also been observed [9,11]. In this paper, we refer to such transitions in the complexity of solving a problem as *computational complexity transitions*, or *complexity transitions* for short.

Complexity transitions of many combinatorial problems, in particular boolean satisfiability and the traveling salesman problem, have recently attracted much attention [6,7,13,21,26–28,30,31,33]. These studies provide a deeper understanding of the problems, and help us to identify difficult problem regions. In addition, they also lead to the development of new methods to solve difficult problems [24,34].

In this paper, we conduct a case study of complexity transitions on the traveling salesman problem. A number of real-world problems, including planning and scheduling problems, can be formulated and solved as the traveling salesman problem [14]. Specifically, we study complexity transitions of branch-and-bound (BnB) for finding an optimal solution of the asymmetric traveling salesman problem. Given n cities and a distance matrix that defines a distance between each ordered pair of cities, the *traveling salesman problem* (TSP) is to find a minimum-distance tour that visits each city exactly once, and returns to the starting city. When the distance matrix is symmetric, i.e. the distance from city i to city j is the same as that from j to i, the problem is the *symmetric* TSP. When the distance from city i to city j is not necessarily equal to that from j to i, it is the *asymmetric* TSP (ATSP). The optimal solutions of these problems can be found by branch-and-bound (BnB) [1,15]. BnB is a general problem-solving technique, and is particularly effective for solving combinatorial problems *optimally*. It is the best known method for finding optimal solutions of the ATSP. BnB includes best-first search (BFS) and depth-first branch-and-bound (DFBnB) as special cases.

Our research was motivated by a recent study of the average-case complexity of BnB on random trees [11,19,20,30,31,33]. The results show that there exists an exponential to polynomial complexity transition, in terms of tree depth, in the average-case complexity of BnB. The first goal of this research was to determine if such complexity transitions exist in real problems, such as the ATSP, and to what extent the analytical results on this random tree model apply to a real problem. We have reported our initial results previously in [30,33]. These results show that there is a complexity transition of BnB on the ATSP. If discrete intercity distances are chosen uniformly from $\{0,1,2,\ldots,r\}$, then the ATSP is easy to solve when r is small, it is difficult when r is large, and the complexity increases significantly when r exceeds a certain value. In short, the average-case complexity of BnB on the ATSP exhibits an easy–hard pattern as r increases.

This research was also motivated by differences between the transition pattern of BnB observed in our previous study on the ATSP, and that reported by Cheeseman et al. on the symmetric TSP [5,6]. In their experiments, Cheeseman et al. randomly generated intercity distances from a log-normal distribution with a fixed mean value, and used Little's algorithm [17] as the search method, which is also a branch-and-bound technique. They found that when the standard deviation σ of the distribution, or the

square root of the variance, is very small or very large, the symmetric TSP is easy to solve, i.e. only a small number of nodes of the search tree are expanded. However, when σ is intermediate, the problem is difficult. In other words, the complexity transition appears as an *easy–hard–easy* pattern as σ increases.

Because of the difference between symmetric and asymmetric intercity distances, different heuristic cost functions are used to solve the symmetric and asymmetric TSP [1]. For the symmetric TSP, the most effective cost function is based on the minimum spanning tree [22], while for the ATSP, the most effective cost function is the solution to the assignment problem [22]. In order to better understand the complexity transitions, and to further study the complexity of BnB. we use a log-normal distribution to generate intercity distances of the ATSP.

The paper is organized as follows. In Section 2 we discuss how to solve the ATSP using BnB, and the complexity transition of BnB on random trees. In Section 3, the main part of the paper, we present experimental complexity transitions of BnB on the ATSP with intercity distances drawn from uniform and log-normal distributions, and investigate the control parameter that determines the complexity. Finally, our summary appears in Section 4.

Some of our preliminary results, such as those in Sections 3.1 and 3.2, were previously reported in [30,32].

2. Tree search and complexity transitions

The search space of a combinatorial problem can usually be represented by a tree. This is because most combinatorial problems can be decomposed by the principle of inclusion and exclusion [1,18] in such a way that no duplicate subproblems are produced. The root node of the search tree corresponds to the original problem, and the interior nodes correspond to subproblems generated. In order to solve the problem more efficiently, lower-bound cost functions, or heuristic evaluation functions, are used to assign values to the nodes to estimate the minimal costs of solving the corresponding subproblems. Which subproblem is chosen for decomposition in each step gives rise to different search algorithms.

In Section 2.1, we take the ATSP as an example to illustrate these concepts of tree search. We then discuss the average-case complexity and complexity transitions of searching a random tree in Section 2.2.

2.1. Solving the ATSP by tree search

The most effective lower-bound cost function for the ATSP is the solution to the *assignment problem*, which is solvable in $O(n^3)$ time for n cities [1,22]. The assignment problem is to assign to each city i another city j, with the distance from i to j as the cost of this assignment, such that the total cost of all assignments is minimized. The assignment problem is a relaxation of the ATSP since the assignments need not form a single tour, but allow collections of disjoint subtours, such as i to j and j to i. Thus, it provides a lower bound on the distance of the ATSP tour. If the assignment problem

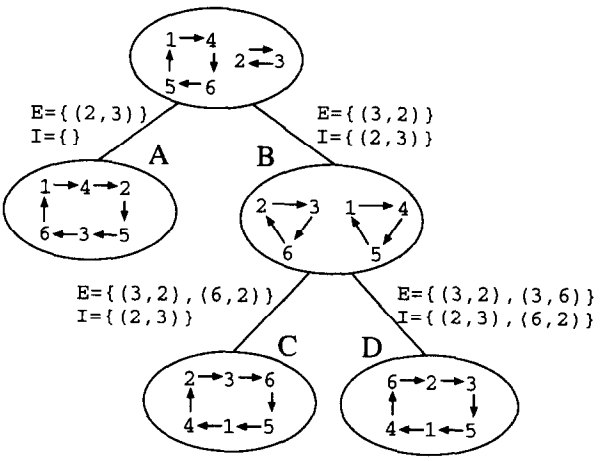

Fig. 1. An example of solving the ATSP.

solution happens to be a single complete tour, it is the solution to the ATSP as well. An example of solving the ATSP using the assignment problem as the cost function is illustrated in Fig. 1.

We first solve the assignment problem for the six given cities. Assume that the assignment problem solution contains two subtours shown in the root node of the tree. We then eliminate a subtour in the solution. If subtour $2-3-2$ is chosen to be eliminated, we have two choices. We may either exclude edge $(2,3)$ or edge $(3,2)$, each of which leads to a subproblem with an additional constraint, the excluded edge. We then solve the assignment problems of the corresponding subproblems. Assume that the solutions to the derived assignment problems are not complete tours. We use inclusion and exclusion to decompose them further, until the solution to a derived assignment problem is a complete tour. We avoid generating duplicate subproblems by including in the current subproblem any edges that were excluded by its previously generated sibling subproblems. In our example, suppose that we generate the first subproblem A by excluding edge $(2,3)$. The second subproblem B excludes edge $(3,2)$, but includes the edge $(2,3)$. Therefore, no subproblems generated under A can include edge $(2,3)$, but all subproblems under B must include edge $(2,3)$, guaranteeing that all their descendent subproblems will be mutually disjoint.

In general, let E denote the set of excluded edges, and I the set of included edges of a subproblem whose assignment problem solution is not a single complete tour. We choose a subtour with a minimum number of edges to eliminate. Assume that there are t edges in the subtour, $\{x_1, x_2, \ldots, x_t\}$, that are not in I. We then decompose the problem into t children, with the kth one having excluded arc set E_k and included arc set I_k, such that

$$E_k = E \cup \{x_k\}, \quad I_k = I \cup \{x_1, \ldots, x_{k-1}\}, \qquad k = 1, 2, \ldots, t. \tag{1}$$

Since x_k is an excluded edge of the kth subproblem, $x_k \in E_k$, and it is an included edge of the $(k+1)$st subproblem, $x_k \in I_{k+1}$, any subproblems generated from

the kth subproblem cannot contain edge x_k, but all subproblems obtained from the $(k+1)$st subproblem must include edge x_k. Therefore, no duplicate subproblems will be generated, and the search space is a tree of unique nodes. There are different schemes for decomposing a subproblem [1], but we adopted the method proposed by Carpaneto and Toth [4], as described in Eq. (1). In addition, a child subproblem is more constrained than its parent problem, by virtue of having more included and excluded edges, so that the cost of the child is at least as large as that of the parent. In other words, subproblem costs are monotonically nondecreasing along a path from the root.

In summary, the ATSP can be solved by BnB as a tree search. It maintains an upper bound α, which is the cost of the best complete tour found so far. The initial value of α is set to the cost of an approximate solution, such as the cost of the nearest neighbor tour [8, 14]. BnB takes the original problem as the current subproblem, and repeats the following steps. First, solve the assignment problem for the current subproblem. If the solution is not a single complete tour, then decompose the problem and generate all child subproblems according to Eq. (1). Next, select a subproblem that has been generated but not yet expanded as the next current subproblem. If the assignment cost of the current subproblem is greater than or equal to the current upper bound α, prune this branch of the tree, since it cannot lead to a complete tour with length less than α, and select another subproblem. If the optimal assignment of the current subproblem is a complete tour, and its length is less than α, update α to the length of this new complete tour. If the optimal assignment of the current subproblem is not a single complete tour, and its cost is less than α, decompose this subproblem. This subproblem selection and decomposition process continues until no unexpanded subproblems exists, or all unexpanded subproblems have costs greater than or equal to α, the length of the best complete tour found so far.

Which subproblem is selected for decomposition in each step gives rise to different implementations of BnB, particularly best-first search or depth-first BnB. Best-first search chooses a subproblem that has the minimum cost among all generated subproblems that have not yet been decomposed. Depth-first BnB selects a subproblem from the most recently decomposed subproblem.

2.2. *Complexity transitions of random tree search*

By relaxing the constraints on a problem, we can usually obtain a heuristic evaluation function which is a lower bound on the cost of solving a subproblem [23]. A lower-bound cost function can be used to construct a monotonic cost function, which assigns a cost to a child node in a search tree that is no less than the cost of its parent. This is done by taking the cost of a node as the maximum of all node costs on the path from the root to the node.

When node costs are monotonically nondecreasing, we can model the cost function by associating nonnegative costs with the edges of the search tree, such that the cost of an edge is the difference between the cost of the child node and the cost of its parent. A node cost is then the sum of the edge costs on the path from the root to the node. To analyze the average-case complexity of BnB, the following random tree was proposed and used in [19, 20, 31, 33].

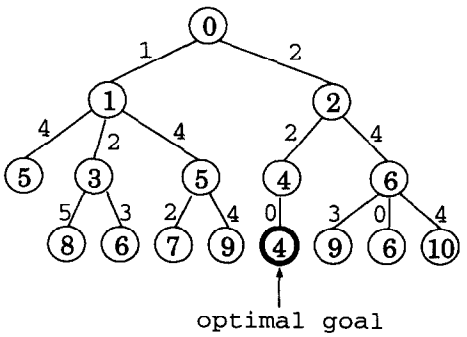

Fig. 2. An example of random tree.

Definition 2.1. An *incremental random tree*, or *random tree* $T(b,d)$, is a tree with depth d, and independent and identically distributed (i.i.d) random branching factors with mean b. Nonnegative edge costs are bounded i.i.d. random values. The cost of a node is the sum of the edge costs along the path from the root to that node. An *optimal goal node* is a node of minimum cost at depth d.

Fig. 2 shows an example of a random tree, where the numbers on the edges and the numbers in the nodes are the edge costs and the resulting node costs, respectively.

The expected number of nodes expanded by BnB to find an optimal goal node of a random tree is governed by the expected number of children of a node that have the same cost as their parent, which are referred to as *same-cost children*. If p_0 is the probability that an edge has zero cost, and b is the mean branching factor, then bp_0 is the expected number of same-cost children of a node.

Lemma 2.2 ([19,20,31,33]). *On a random tree $T(b,d)$, as $d \to \infty$, both best-first search and depth-first branch-and-bound expand $\theta(\beta^d)$ expected number of nodes when $bp_0 < 1$, where β is a constant between 1 and b. Best-first search expands $O(d^2)$ expected number of nodes, and depth-first branch-and-bound expands $O(d^3)$ expected number of nodes when $bp_0 \geqslant 1$.*

Lemma 2.2 shows that there exists a complexity transition, from exponential to polynomial in the search depth, when the expected number of same-cost children bp_0 increases from below one to above one. This is summarized by Fig. 3.

We need to emphasize that a random tree is an abstraction of a practical tree-search problem, making many assumptions to enable a tractable average-case analysis. For example, branching factors of different nodes in a random tree are independent and identically distributed, referred to as the i.i.d. assumption, which is rarely true in a real problem. Furthermore, edge costs of a random tree are independent from each other, which is also rarely true in practice. In general, to bridge the gap between an analytical model and a practical problem, empirical results are required.

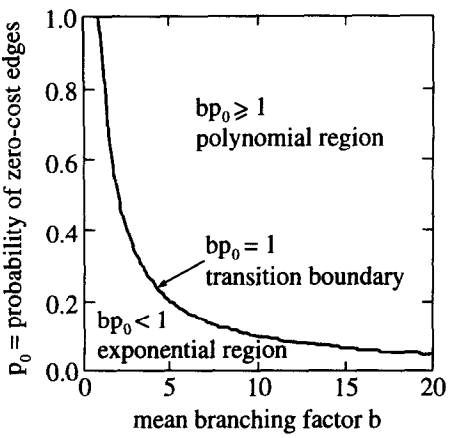

Fig. 3. Complexity transition of random-tree search.

3. Complexity transitions on the ATSP

Do complexity transitions exist in real search problems, such as the ATSP? How applicable are analytic results of complexity transitions on random trees to the ATSP? How do problem parameters affect complexity transitions? This section addresses these questions.

3.1. Transitions under uniform distributions

The node costs in the search tree of the ATSP are the solution costs of the corresponding assignment problems. Assume that intercity distances are uniformly chosen from $R = \{0, 1, 2, \ldots, r\}$, for some positive integer r. Consider the relationship between the intercity distance range r, and the number of edges in the search tree that have zero cost. The probability that two sets of n values from R have the same total sum is smaller if r is larger. Thus, the probability that two sets of n edges in the assignment problem solutions to two subproblems have the same total cost decreases as r increases. When r is small compared to the number of cities, the probability that the assignment problem cost of a child subproblem in the search tree is equal to the assignment problem cost of its parent is large. In other words, the probability p_0 of zero-cost edges in the search tree is larger when r is smaller, so that the average number of same-cost children bp_0 may be greater than one, if the branching factor b does not decrease significantly when r increases. Therefore, the problem may be easy to solve when r is small. Conversely, the probability that the assignment problem cost of a child is equal to that of its parent is smaller when r is relatively larger, as is the probability p_0 of zero-cost edges in the search tree. Consequently, the problem may be difficult to solve when r is large.

The above argument suggests that there may exist a complexity transition of BnB on the ATSP as r changes, following the complexity transition of random-tree search shown in Fig. 3. We experimentally tested this prediction by experiments on 100-city ATSPs. In our experiments, we randomly generated 1000 problem instances for each different

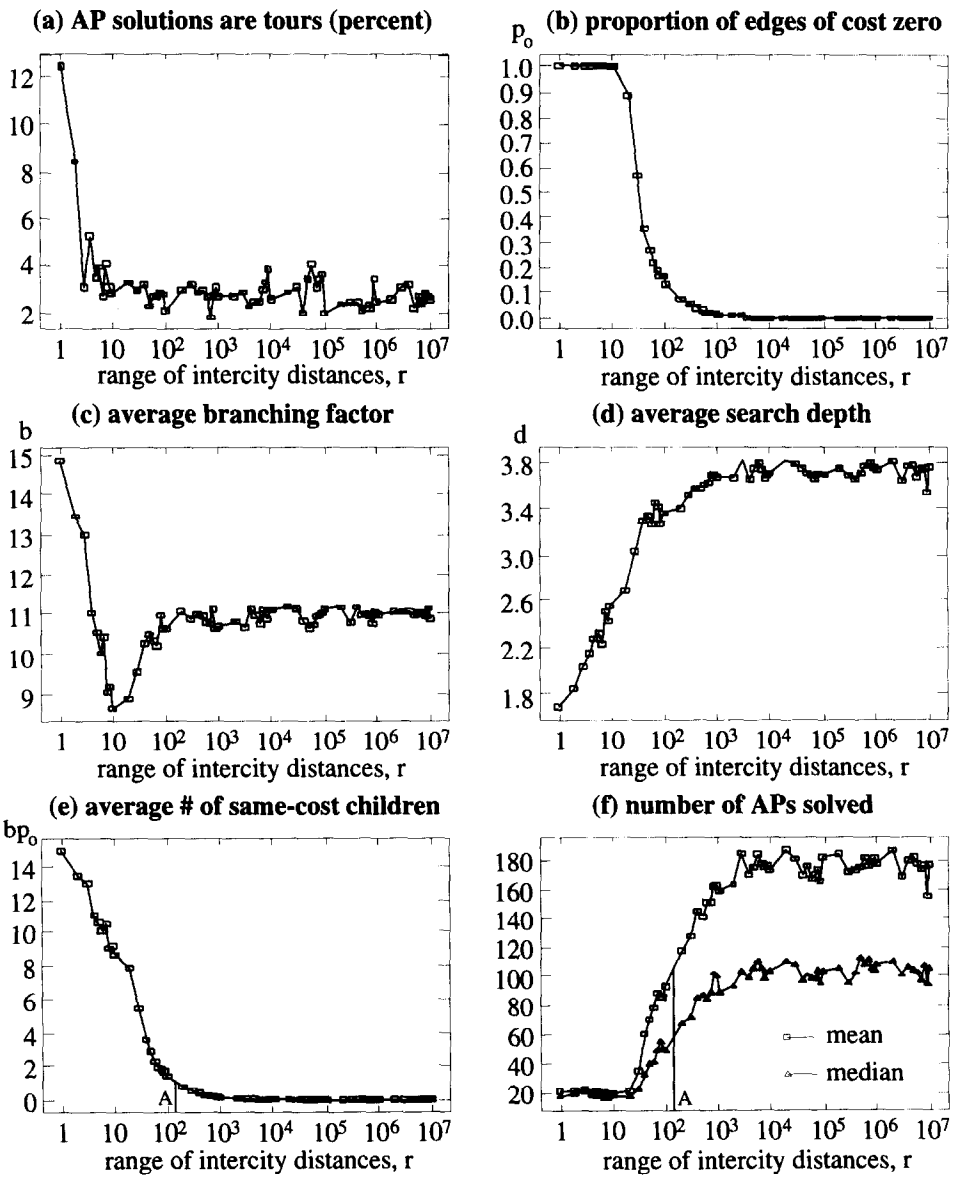

Fig. 4. Complexity transition on the 100-city ATSP, uniform distribution on $\{0, 1, 2, \ldots, r\}$.

value of r, solved them using depth-first BnB, and averaged the results for each value of r. Fig. 4(a) shows the percentage of problem instances whose assignment problem solutions are complete tours, where the horizontal axis is the intercity-distance range r, on a logarithmic scale. Overall, this percentage is small, especially for $r \geqslant 5$. We further examined four important parameters of the search trees of the ATSP instances whose assignment problem solutions are not complete tours. These include the proportion of

edges in the search tree that have cost zero, which corresponding to the probability p_0 that a node has the same cost as its parent, the average branching factor b, the average search depth d, and the average number of same-cost children of a node bp_0. These four parameters are graphed in Figs. 4(b) through 4(e).

Fig. 4(b) shows that when $r \leqslant 10$, p_0 is almost one, so that most nodes in the search tree have the same cost, and almost every leaf node is an optimal goal node. On the other hand, when $10 < r < 1000$, increasing r causes p_0 to decrease sharply, and approach zero. Figs. 4(c) and 4(d) illustrate the average branching factor b and average search depth d, respectively, which determine the size of the search tree. When $r \leqslant 10$, b increases but d decreases as r decreases, meaning that the search tree gets bushier and shallower as r decreases. b and d remain relatively constant when $r > 100$. Fig. 4(e) shows that $bp_0 > 1$ when $r \leqslant 130$, and $bp_0 < 1$ when $r > 130$, indicated by point A. Following the complexity transition of random-tree search in Fig. 3, this means that the problem should be easy to solve when $r \leqslant 130$, but should be difficult when $r > 130$.

Fig. 4(f) presents the mean and median numbers of assignment problems (AP) solved, or search tree nodes generated, under different values of intercity-distance range r, showing a complexity transition. When $r \leqslant 20$ the problem is easy, but is difficult when $r > 1000$. Similar complexity transitions have also been observed on 200-, 300-, 400- and 500-city ATSPs. Fig. 5 shows the complexity transition on the 200-city ATSP.

The transition from easy problem instances to difficult ones, however, is not as dramatic as we expected. This may be due to the following factors. First, bp_0 decreases gradually as r increases for $r < 1000$, making the complexity increase slowly. Secondly, the size of the search tree increases slowly as b and d increase with r for $10 < r < 1000$ (Figs. 4(c) and 4(d)). Thirdly, the search trees of the ATSPs have relatively small depths, while the complexity transitions of random tree search are asymptotic results as tree depth approaches infinity. Finally and more importantly, the edge costs and branching factors of different nodes in the search tree are not random variables, so that the i.i.d. assumptions made for random trees are violated. Nevertheless, point A in Fig. 4(f), which corresponds to point A of Fig. 4(e) for $bp_0 = 1$, is in the middle of the complexity transition.

As suggested by Section 2, if we take the particular value of r such that $bp_0 = 1$ as the transition point, point A in Figs. 4(e) and (f), then this transition point increases with the problem size, or the number of cities. Point A' in Figs. 5(a) and 5(b) is the transition point on the 200-city ATSP, corresponding to $bp_0 = 1$. Compared to point A in Fig. 4, point A' in Fig. 5 has a larger value of intercity-distance range $r = 450$.

3.2. Transitions under log-normal distributions

We now examine the complexity of BnB for solving ATSP with intercity distances drawn from a log-normal distribution [10]. It is a continuous distribution, whose density function[2] is

[2] This is called a two-parameter density function [10].

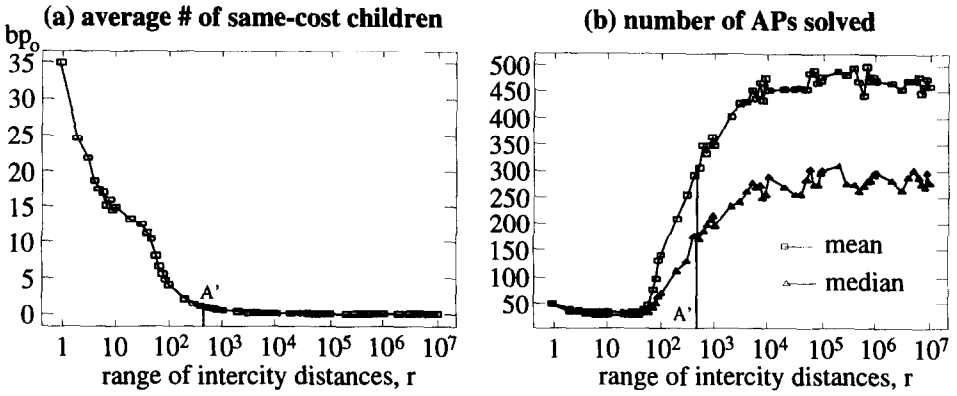

Fig. 5. Complexity transition on the 200-city ATSP, uniform distribution on $\{0, 1, 2, \ldots, r\}$.

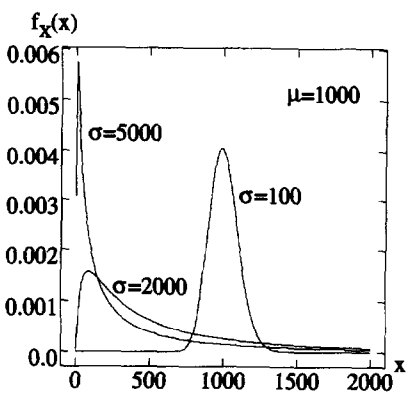

Fig. 6. Density functions of the log-normal distribution with mean 1000.

$$fx(x) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\gamma x} \exp\left[-\dfrac{1}{2}\left(\dfrac{\log x - \lambda}{\gamma}\right)^2\right], & \text{for } x > 0, \\ 0, & \text{for } x \leqslant 0. \end{cases} \tag{2}$$

The mean μ and the standard deviation σ of the distribution are

$$\mu = e^{\lambda+\gamma^2/2}, \qquad \sigma = \sqrt{e^{2\lambda+\gamma^2}(e^{\gamma^2} - 1)}. \tag{3}$$

Fig. 6 illustrates the density function with a constant mean of $\mu = 1000$, and three different values of the standard deviation σ.

When the standard deviation σ of the distribution is very small, most values from the distribution are located near its mean value, so that most of them are the same after being discretized. Thus, when σ is small, many intercity distances drawn from this distribution are equal, and the proportion p_0 of zero-cost edges in the search tree is large, following our previous arguments. One observation on the log-normal distribution is that

it is biased toward small costs when σ increases. When σ is very large, most intercity distances are almost zero with a few very large intercity distances, which are unlikely to be chosen in the solution to an assignment problem. Consequently, the proportion p_0 of zero-cost edges in the search tree is large in this case as well. Thus, we should expect that the ATSP is easy to solve when σ is small and large, but is difficult when σ is intermediate.

We verified our arguments with experiments, in which we fixed the mean of the distribution to a constant, $\mu = 1000$, and used the standard deviation σ as a parameter to adjust the distribution. We generated 1000 problem instances of the 100-city ATSP for each different value of σ. Fig. 7 shows our experimental results for depth-first BnB. Fig. 7(a) is the percentage of instances whose assignment problem solutions are complete tours, as a function of σ, on a logarithmic scale. This percentage grows with σ, indicating more easy problems were generated as σ increases, supporting our arguments. Similar to the experiments with a uniform distribution, we also examined four parameters of the search trees of the problem instances whose assignment problem solutions are not complete tours. Figs. 7(b) to 7(e) show the proportion or probability p_0 that a node has the same cost as its parent, the average branching factor b, the average search depth d, and the average number of same-cost children of a node bp_0, respectively.

Fig. 7(b) shows that when the standard deviation σ increases for $\sigma \geqslant 0.4$, the proportion of zero-cost edges in the search tree decreases, to near zero, then increases to one, and finally remains at one. Figs. 7(c) and 7(d) show the average branching factor b and average search depth d, which determine the search tree size. Fig. 7(e) shows that when σ increases, the average number of same-cost children bp_0 first decreases, then remains at zero, and finally increases. bp_0 drops below one when σ exceeds 8 (point B in Fig. 7(e)), and then grows above one when σ exceeds 6000 (point C in Fig. 7(e)). Fig. 7(f) displays the mean and median numbers of assignment problems (AP) solved, or tree nodes generated, which follow a similar easy–difficult–easy transition pattern as that of the symmetric TSP [6]. Fig. 7(f) shows that the problem is relatively easy when σ is small or large, but is difficult when σ is intermediate.

However, the behavior of BnB in the region when $\sigma < 0.4$ does not completely match our intuition. When σ is very small, say $\sigma < 0.4$, most intercity distances are equal to the mean of the distribution. In this case, most subproblems in the search tree should have the same assignment cost, giving a high proportion of edges of cost zero in the tree. In addition, the assignment problem algorithm intends to find solutions with small cycles, so that the branching factor of the search tree should be small and the search tree should be deep in this case. The average search depth in Fig. 7(d) when $\sigma < 0.4$ confirms our intuition, but the average branching factor in Fig. 7(c) when $\sigma < 0.4$ does not.

Similar to the case when intercity distances are uniformly distributed, the transitions from easy problem instances to difficult ones, and from difficult instances to easy ones in Fig. 7(f), are not very sharp. This is mostly due to the fact that edge costs in the search tree are not independent. Nevertheless, points B and C of Fig. 7(e), which correspond to points B and C of Fig. 7(d) for $bp_0 = 1$, are both located at the middle of the respective complexity transitions.

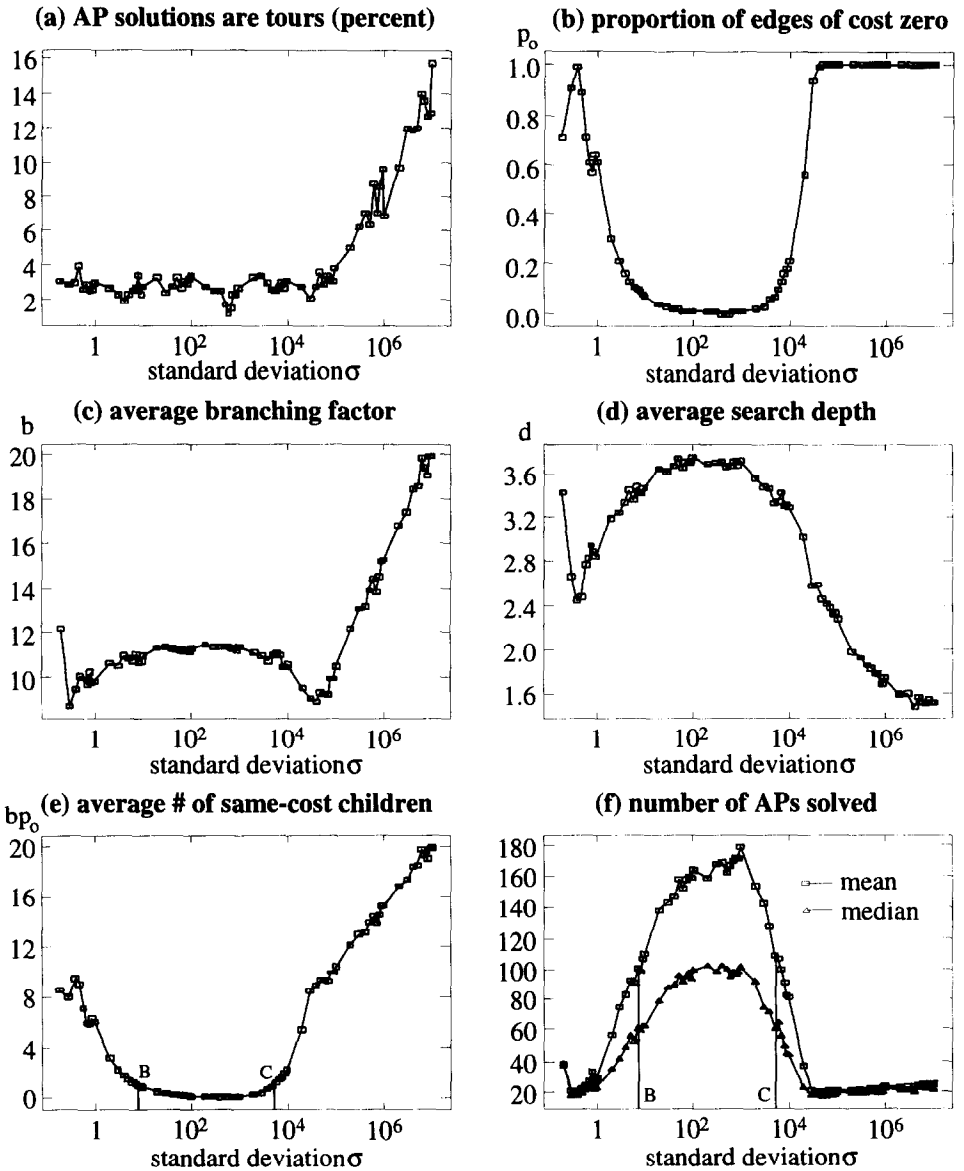

Fig. 7. Complexity transition on the 100-city ATSP, log-normal distribution with a fixed mean of 1000.

3.3. *Identifying the control parameter*

Our experiments in Sections 3.1 and 3.2 indicate that complexity transitions of the ATSP depend on the distributions of intercity distances. What factor causes different complexity transition patterns on the ATSP under different intercity-distance distributions? In other words, what is the control parameter?

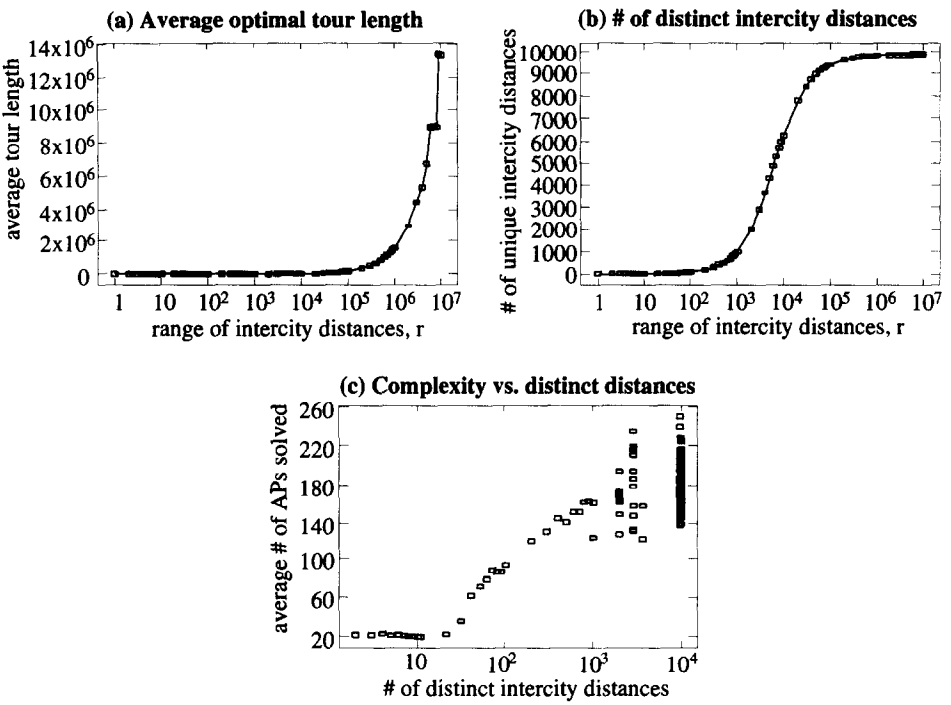

Fig. 8. Tour length, distinct intercity costs and complexity, uniform distribution on $\{0, 1, 2, \ldots, r\}$.

The complexity of the ATSP depends on the number of different intercity distances, their values, and the probability that a distance takes a particular value.

When intercity distances are uniformly chosen from $\{0, 1, 2, \ldots, r\}$, both the optimal ATSP tour length and the total number of distinct intercity distances increase with the intercity-distance range r. For the 100-city ATSP, the maximum number of distinct intercity distances is 10000, which is the total number of entities in the 100 by 100 distance matrix. We examined these two parameters using the same 1000 problem instances of the 100-city ATSP for each different r as we used for Fig. 4 of Section 3.1. Fig. 8(a) shows the average tour length, and Fig. 8(b) displays the average number of distinct intercity distances, as a function of the range r. The results in these figures confirm that the ATSP tour length and the number of distinct intercity distances grow with r.

We then investigated the relationship between the number of distinct intercity distances and the problem complexity, the total number of assignment problems solved. The numbers of distinct intercity distances of two particular problem instances that are generated using the same intercity-distance range r are not necessarily the same. Furthermore, the number of distinct intercity distances of two problem instances that are generated using two different r may be the same. Therefore, we rearranged the problem instances generated from all values of r that we used, according to their numbers of distinct intercity distances. Since some numbers of distinct intercity distance may be represented by only a few problem instances, we chose those numbers of distinct

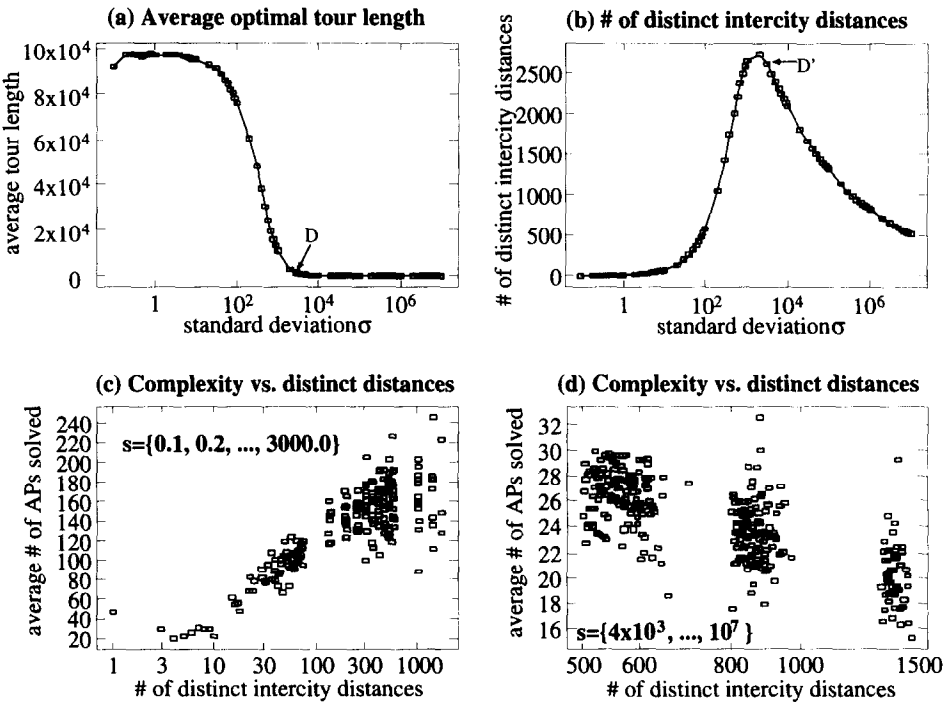

Fig. 9. Tour length, distinct intercity costs and complexity, log-normal distribution with a fixed mean of 1000.

intercity distances that are represented by more than 30 problem instances, and plotted in Fig. 8(c) the average number of assignment problems solved as a function of the number of distinct intercity distances. Fig. 8(c) shows that the complexity of the ATSP increases with the number of distinct intercity distances. For example, when the number of distinct intercity distances is less than 40, the average total number of assignment problems solved is less than 25, and when the number of distinct intercity distances is greater than 100, the total number of assignment problems solved is greater than 120. In short, when intercity distances are chosen from a uniform distribution, the number of distinct intercity distances serves as the control parameter of the complexity transition. The problem is easy to solve when the number of distinct intercity distances is small, but is difficult when this number is large.

We now consider the log-normal distribution. Fig. 9(a) and Fig. 9(b) show the average tour length and the number of distinct intercity distances, averaged over the same 1000 problem instances of the 100-city ATSP for each standard deviation σ as those used for Fig. 7 in Section 3.2, as a function of σ. Fig. 9(a) shows that when $\sigma < 10$, the average tour length is close to 100×1000, indicating that the intercity distances included in the ATSP tour are close to the mean of the distribution, which is 1000. When $\sigma \geqslant 3000$ (after point D in Fig. 9(a)), the average tour length is less than 1000, and approaches zero quickly, indicating that small intercity distances are included in the ATSP tour, although large ones exist. This is mostly due to the fact that in this case most intercity distances are very small, and only a few large ones exist.

Fig. 9(b) shows that the average number of distinct intercity distances increases with σ when $0.1 \leqslant \sigma \leqslant 2000$. However, it decreases with σ when $2000 \leqslant \sigma \leqslant 10,000,000$ (after point D' in Fig. 9(b)). This further indicates that more small intercity distances are generated for $\sigma > 2000$ in order to keep the mean constant. Most of these small distances are the same after being discretized. This is also the reason that the complexity drops quickly when $\sigma \geqslant 2000$ in Fig. 7(f).

We further examined the relationship between the number of distinct intercity distances and the number of assignment problems solved. Similar to the case of the uniform distribution, we rearranged the problem instances according to their numbers of distinct intercity distances, and report the results for the numbers represented by more than 30 problem instances. Since many intercity distances generated with $\sigma > 3000$ are equal to a few small values, and the large ones are rarely used in the optimal assignment, we separated the problem instances that were generated using $\sigma \leqslant 3000$ from those using $\sigma > 3000$. Fig. 9(c) and Fig. 9(d) show the average number of assignment problems solved as a function of the number of distinct intercity distances for the problem instances generated with the standard deviation $\sigma \leqslant 3000$ and $\sigma > 3000$, respectively. Fig. 9(d) shows that when σ is large, the average number of assignment problems (AP) solved is less than 35, meaning that the problem is easy to solve. Fig. 9(c) shows that the problem is easy when the number of distinct intercity distances is small, but is more difficult when this number is large. In summary, when intercity distances are chosen from the log-normal distribution, if the problem instances that are generated under an extremely large standard deviation are excluded, the number of distinct intercity distances can be used as the control parameter of the complexity transition.

However, the change from easy problem instances to difficult instances, in both cases of uniform and log-normal distributions, is not as sharp as we expected. This is more likely due to the same reason as described in Section 3.1, namely that the branching factors and edge costs in the search tree are not independent of each other.

We would like to emphasize that the meaning of the above results is that when the number of distinct intercity distances is relatively large, solving the ATSP requires computation that may be exponential *in the search depth* on average. This does not imply, however, that the average-case complexity of the ATSP is exponential *in the number of cities* when the number of distinct intercity distances is relatively large. In fact, it is still an open problem whether or not the ATSP can be solved in polynomial or exponential average time in the number of cities, and the opinion of experts on this question is divided [2, 12, 16, 25].

4. Conclusions

We have experimentally demonstrated the existence of complexity transitions of branch-and-bound for finding an optimal solution of the ATSP. We found that when discrete intercity distances were chosen uniformly from $\{0, 1, 2, \ldots, r\}$, the complexity exhibits an easy–hard transition as r increases. We also observed that when the intercity distances were drawn from a discretized log-normal distribution, the complexity displays easy–hard–easy transitions as the standard deviation of the distribution grows.

This transition pattern is similar to the complexity transition observed on the symmetric TSP [5,6]. Furthermore, we explained why there exist two different transition patterns on the ATSP by hypothesizing that the control parameter that determines the transitions is the total number of distinct intercity distances of the ATSP. The complexity transition follows an easy–hard transition as the number of distinct intercity distances increases.

This research indicates that analytical results of tree-search complexity transitions can be applied to real search problems, such as the ATSP, helping us to predict the complexity transitions of the problem. However, the transition between easy and difficult regions is not as sharp as predicted by the analytical results. This is most probably due to the fact that the independence assumptions in the analysis are not valid in real problems.

Acknowledgements

Thanks to Peter Cheeseman, Tad Hogg and Colin Williams for stimulating discussions related to this work. We are also indebted to the two anonymous reviewers for their excellent comments that greatly improve the content and presentation of this paper. Thanks also to William Cheng for *tgif*, David Harrison for *xgraph*, and the Free Software foundation for *gcc*, *gdb*, and *gnuemacs*.

References

[1] E. Balas and P. Toth, Branch and bound methods, in: E.L. Lawler et al., eds., *The Traveling Salesman Problem* (Wiley, Chichester, England, 1985) 361–401.

[2] M. Bellmore and J.C. Malone, Pathology of traveling-salesman subtour-elimination algorithms, *Oper. Res.* **19** (1971) 278–307.

[3] B. Bollobás, *Random Graphs* (Academic Press, New York, 1985).

[4] G. Carpaneto and P. Toth, Some new branching and bounding criteria for the asymmetric traveling salesman problem, *Management Sci.* **26** (1980) 736–743.

[5] P. Cheeseman, Personal communication (1991–1992).

[6] P. Cheeseman, B. Kanefsky and W.M. Taylor, Where the really hard problems are, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 331–337.

[7] J.M. Crawford and L.D. Auton, Experimental results on the crossover point in satisfiability problems, in: *Proceedings of AAAI-93*, Washington, DC (1993) 21–27.

[8] A. Frieze, G. Galbiati and F. Maffioli, On the worst-case performance of some algorithms for the asymmetric traveling salesman problem, *Network* **12** (1982) 23–39.

[9] B.A. Huberman and T. Hogg, Phase transitions in artificial intelligence systems, *Artif. Intell.* **33** (1987) 155–171.

[10] N.L. Johnson and S. Kotz, *Distributions in Statistics: Continuous Univariate Distributions*—1 (Wiley, New York, 1970).

[11] R.M. Karp and J. Pearl, Searching for an optimal path in a tree with random costs, *Artif. Intell.* **21** (1983) 99–117.

[12] R.M. Karp and J.M. Steele, Probabilistic analysis of heuristics, in: E.L. Lawler et al., eds., *The Traveling Salesman Problem* (Wiley, Chichester, England, 1985) 181–205.

[13] T. Larrabee and Y. Tsuji, Evidence for a satisfiability threshold for random 3CNF formulas, in: *Working Notes of AAAI 1993 Spring Symposium: AI and NP-Hard Problems*, Stanford, CA (1993) 112–118.

[14] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, *The Traveling Salesman Problem* (Wiley, Chichester, England, 1985).

[15] E.L. Lawler and D.E. Wood, Branch-and-bound methods: a survey, *Oper. Res.* **14** (1966) 699–719.

[16] J.K. Lenstra and A.H.G. Rinnooy Kan, On the expected performance of branch-and-bound algorithms, *Oper. Res.* **26** (1978) 347–349.
[17] J.D.C. Little, K.G. Murty, D.W. Sweeney and C. Karel, An algorithm for the traveling salesman problem, *Oper. Res.* **11** (1963) 972–989.
[18] C.L. Liu, *Introduction to Combinatorial Mathematics* (McGraw-Hill, New York, 1968).
[19] C.J.H. McDiarmid, Probabilistic analysis of tree search, in: G.R. Gummett and D.J.A. Welsh, eds., *Disorder in Physical Systems* (Oxford Science, 1990) 249–260.
[20] C.J.H. McDiarmid and G.M.A. Provan, An expected-cost analysis of backtracking and non-backtracking algorithms, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 172–177.
[21] D.G. Mitchell, B. Selman and H.J. Levesque, Hard and easy distributions of SAT problems, in: *Proceedings AAAI-92*, San Jose, CA (1992) 459–465.
[22] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, NJ, 1982).
[23] J. Pearl, *Heuristics* (Addison-Wesley, Reading, MA, 1984).
[24] J.C. Pemberton and W. Zhang, Epsilon-transformation: exploiting phase transitions to solve combinatorial optimization problems, *Artif. Intell.* **81** (1996) 297–325 (this volume).
[25] D.R. Smith, Random trees and the analysis of branch and bound procedures, *J. ACM* **31** (1984) 163–188.
[26] C.P. Williams and T. Hogg, Using deep structure to locate hard problems, in: *Proceedings AAAI-92*, San Jose, CA (1992) 472–477.
[27] C.P. Williams and T. Hogg, Extending deep structure, in: *Proceedings of AAAI-93*, Washington, DC (1993) 152–157.
[28] C.P. Williams and T. Hogg, Exploiting the deep structure of constraint problems, *Artif. Intell.* **70** (1994) 73–117.
[29] K.G. Wilson, Problems in physics with many scales of length, *Scientific American* **241** (1979) 158–179.
[30] W. Zhang and R.E. Korf, An average-case analysis of branch-and-bound with applications: summary of results, in: *Proceedings AAAI-92*, San Jose, CA (1992) 545–550.
[31] W. Zhang and R.E. Korf, Depth-first vs. best-first search: new results, in: *Proceedings AAAI-93*, Washington, DC (1993) 769–775.
[32] W. Zhang and R.E. Korf, A unified view of complexity transitions on the traveling salesman problem, in: *Working Notes of AAAI 1994 Workshop on Experimental Evaluation of Reasoning and Search Methods*, Seattle, WA (1994).
[33] W. Zhang and R.E. Korf, Performance of linear-space search algorithms, *Artif. Intell.* **79** (1995) 241–292.
[34] W. Zhang and J.C. Pemberton, Epsilon-transformation: exploiting phase transitions to solve combinatorial optimization problems—initial results, in: *Proceedings of AAAI-94*, Seattle, WA (1994).