THE PROBABILISTIC ANALYSIS OF SOME COMBINATORIAL SEARCH ALGORITHMS

by

Richard M. Karp

Memorandum No. ERL-M581

28 April 1976

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

THE PROBABILISTIC ANALYSIS OF SOME COMBINATORIAL SEARCH ALGORITHMS[†]

Richard M. Karp
Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley

ABSTRACT -- The traveling-salesman problem, the set covering problem and the vertex coloring and maximum independent set problems for graphs are examples of combinatorial optimization problems that cannot be solved in polynomial time unless $P = NP$. Polynomial-time algorithms guaranteed to yield approximately optimal solutions to these problems are not known, and possibly do not exist. This paper gives a survey of a probabilistic approach, in which we assume that instances of these problems are drawn from specified probability distributions, and show that certain polynomial-time algorithms almost surely find near-optimal solutions.

1. THE LIMITATIONS OF WORST-CASE ANALYSIS

Many of the chief problems in the field of combinatorial algorithms appear intractable, in the sense that every known solution method experiences a combinatorial explosion in its worst-case running time as a function of the size of the input. Among these apparently intractable problems are the traveling-salesman problem, the problem of finding the chromatic number of a graph, the problem of finding a maximum number of mutually nonadjacent vertices in a graph, and the problem of testing whether a formula in the propositional calculus is satisfiable.

These problems, as well as many others, are equivalent, in the sense that the existence of an algorithm for solving any one of them within a polynomial time bound would imply that each of them is solvable within polynomial time. Such equivalences are established using the concepts of polynomial-time reducibility and NP-completeness [Cook (71), Karp (72), Karp (75a), Levin (73)]. It appears likely that none of these equivalent problems can be solved in polynomial time.

[†]Research supported by National Science Foundation grant DCR74-17680-A01. To appear in the Proceedings of the Symposium on Algorithms and Complexity: New Directions and Recent Results, April 7-9, 1976, Pittsburgh, Pennsylvania.

The evidence for this belief stems from our inability to find polynomial-time algorithms for these problems despite the intensive efforts of many workers, and from the theoretical result that a polynomial time algorithm for one of these problems would imply that $P = NP$ (cf. Cook (71), Karp (72), Karp (75a)); i.e., that polynomial-time algorithms exist for an unexpectedly wide class of combinatorial search problems.

Let us call a problem *NP-hard* if the existence of a polynomial-time algorithm for its solution implies $P = NP$. It has sometimes been suggested that, unless it turns out to our surprise that $P = NP$, all the NP-hard problems must be inherently resistant to practical solution. But, in fact, a proof that $P \neq NP$ would have only limited negative implications about the feasibility of attacking these problems. For an optimization problem such as the traveling-salesman problem, there would still be a possibility of finding a polynomial-time algorithm that gives approximately optimal solutions for all inputs, or optimal solutions for almost all inputs, or approximately optimal solutions for almost all inputs.

The investigation of polynomial-time algorithms guaranteed to give approximately optimal solutions to NP-hard optimization problems was pioneered by Ronald Graham [Graham (66)], and has been a very active area since 1973 (cf. Garey and Johnson (76b)). The definitional set-up is as follows. Consider a minimization problem (such as the traveling-salesman problem), which, for each problem instance I, asks for a solution of minimum cost. Let this minimum cost be $C^*(I)$. Consider an algorithm A that, on problem instance I, produces a solution of cost $C_A(I)$. Then, given a real number $r > 1$, we say that A solves the problem within the ratio r if there is a d such that, for all I,

$$C_A(I) \leq rC^*(I) + d \ .$$

This "guaranteed approximation" approach has yielded a number of successes, particularly in connection with various packing problems (cf. Garey and Johnson (76b)). On the other hand, the approach apparently fails to give positive results in some important cases. The most striking negative result [Garey and Johnson (76a)] is that, unless $P = NP$, no polynomial-time algorithm can solve the vertex-coloring problem for graphs within a ratio $r < 2$. At present, no polynomial-time algorithm is known that solves the vertex-coloring problem, the problem of finding a maximum independent set of vertices in a graph, or the set covering problem, within any bounded ratio. In the case of the traveling-salesman problem in the plane, the best ratio known to be achievable through a polynomial-time algorithm is $r = 3/2$ [Christofides (76)], which is probably not acceptable in practice. Thus, unless improved results can be obtained, the "guaranteed approximation" approach seems unable to identify polynomial-time algorithms that will be effective in solving these problems.

2. A PROBABILISTIC APPROACH

The present paper explores a relatively untested approach to the validation of algorithms for *NP*-hard problems. We retreat from worst-case analysis, and instead consider algorithms guaranteed to give optimal or near-optimal solutions on almost all problem instances. To formulate what "almost all" means, we must introduce a probability distribution over the set of problem instances of each size. The chief objection to the approach is that we seldom can know what probability distribution is realistic. Nevertheless, we feel that the approach yields valuable insights that are not accessible through worst-case analysis.

Let us associate with each problem instance I a *size* $|I|$; the size of a graph might be its number of vertices, the size of a traveling-salesman problem instance might be the number of cities, etc. For each size N, we assume as given a probability distribution S_N over the problem instances of size N. Let $X(\cdot)$ be some predicate which is true or false of each problem instance. Let q_N be the probability that $X(I)$ does *not* hold when I is drawn from S_N. Then we say that $X(I)$ *holds almost everywhere* (abbreviated a.e.) if $\sum_N q_N < \infty$. This is a very strong condition. It implies that, if we drew an infinite sequence of problem instances, one of each size, then, with probability 1, the predicate $X(\cdot)$ would be observed to fail only finitely often. On the other hand, it is an asymptotic condition, and says nothing about problem instances up to, say, size $N = 500$.

We shall be interested in polynomial-time algorithms that solve *NP*-hard problems (either exactly or within a specified ratio r) almost everywhere. The motivation for our approach can be understood by considering the simplex algorithm for solving linear programming problems. After more than two decades of intensive use, the simplex algorithm is known to solve large linear programming problems efficiently and reliably. Yet families of examples are known [Klee and Minty (70)] for which the running time of the simplex method grows exponentially with problem size. It is important to explore whether such catastrophic examples are rare. One might hope to do so by defining appropriate probability distributions and examining whether the simplex method, truncated to run within polynomial time, yields optimal solutions almost everywhere. Such an investigation remains to be carried out.

In the present paper we survey some results about the "almost everywhere" behavior of algorithms for finding traveling-salesman tours, Hamilton circuits, graph colorings, independent sets of vertices in graphs, and set coverings. Many of the results are new, and will be proved in forthcoming papers by the author. In each case we have made the simplest probability assumptions that lead to nontrivial questions. The algorithms we have chosen to analyze are also very simple. In every instance heuristics will occur to the reader that would undoubtedly improve the performance of the algorithms; unfortunately, the introduction of complex heuristics introduces probabilistic dependencies that are extremely difficult to analyze.

3. THE EUCLIDEAN TRAVELING-SALESMAN PROBLEM

An instance of the euclidean traveling-salesman problem is specified by a finite set of points in the plane; the solution consists of a polygon of minimum perimeter having the given points as its vertices. The problem is *NP*-hard [Garey, Graham and Johnson (76)]. The strongest known worst-case approximation result concerning the problem is due to N. Christofides, who gives an algorithm that runs in time $O(n^3)$, and solves the problem within $r = 3/2$. On the other hand, it has often been observed that persons can usually write down nearly optimum solutions to random euclidean traveling-salesman problems by inspection. We show here that such human behavior can be emulated by a simple algorithm.

We assume that a problem instance of size n is chosen by drawing n points independently from a uniform distribution over the unit square. We present an approximate solution algorithm based on partitioning the unit square into subregions within which optimum tours can be found exactly; the subtours are then joined to form a near-optimal tour through all the points.

Some preliminary remarks are needed. First, there is a dynamic programming algorithm capable of solving x-city traveling-salesman problems in $O(x \cdot 2^x)$ steps [Bellman (60), Held and Karp (72)]. Second, the traveling-salesman problem is not changed if we allow as feasible solutions arbitrary closed walks through the vertices, including those with crossing edges, and those in which certain vertices are repeatedly

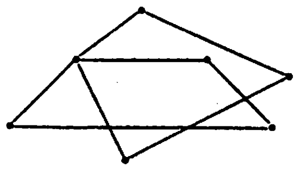

Fig. 1. A closed walk through seven points

traversed. It is easy to transform any such closed walk into a simple polygon with a smaller perimeter.

In specifying the algorithm, we mention a function t(n); t is any function that maps the positive integers into the positive rationals such that

(a) $t \sim \log_2 \log_2 n$ and

(b) for all n, $\frac{n}{t}$ is a perfect square.

Algorithm A

1. Subdivide the unit square into a regular grid of $\frac{n}{t(n)}$ subsquares, each of side $\sqrt{\frac{t(n)}{n}}$.
2. Using dynamic programming, construct an optimum tour through the set of points in each subsquare.
3. Regard each of the $\frac{n}{t(n)}$ subtours as a point, with the distance between subtours P_1 and P_2 equal to the shortest distance between a point on P_1 and a point on P_2. Construct a minimum-length spanning tree joining the subtours.
4. Construct a closed walk W that traverses each subtour once, and each tree edge twice. W represents the desired solution.

Theorem 1. Algorithm A runs within time $O(n \log n)$ (a.e.). For every $\epsilon > 0$, Algorithm A solves the euclidean traveling-salesman problem within $1 + \epsilon$ (a.e.).

Theorem 1 is based on the following result, which is given in [Beardwood, Halton and Hammersley (59)]. Let the random variable X_n denote the minimum perimeter of a polygon through n points drawn at random from a uniform distribution over the unit square.

Theorem 2. There is a constant β such that, for every $\epsilon > 0$,

$$\beta - \epsilon < \frac{X_n}{\sqrt{n}} < \beta + \epsilon \quad \text{a.e.}$$

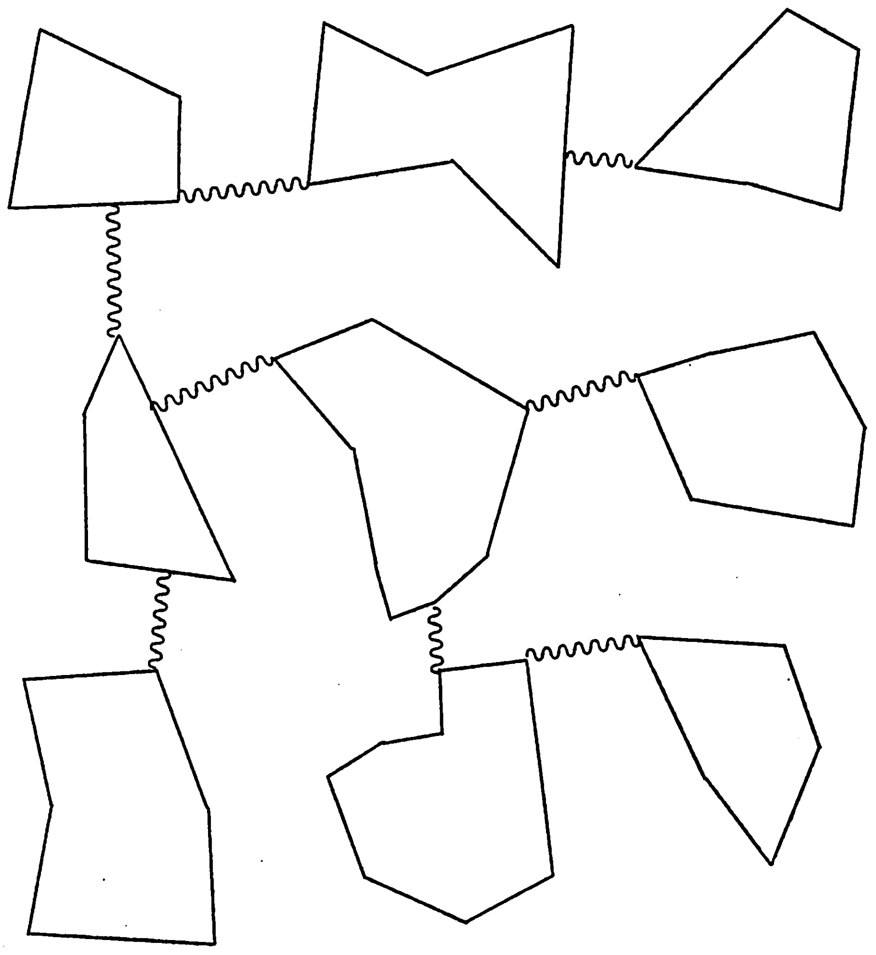

Fig. 2. Walk constructed by Algorithm A

Let Y_n be a random variable equal to the sum of the lengths of the n/t subtours constructed by the algorithm; let Z_n be the length of the minimum spanning tree joining the subtours. From Theorem 1 it follows that, for every $\epsilon > 0$,

$$(\beta-\epsilon) < \frac{Y_n}{\sqrt{n}} < (\beta+\epsilon) \quad \text{a.e.}$$

We can also show that there exists a constant γ such that

$$Z_n < \frac{\gamma\sqrt{n}}{t(n)} \quad \text{(a.e.)}$$

Theorem 2 follows readily from these remarks.

Algorithms based on similar partitioning principles, and with similar properties, apply to Steiner tree problems as well as traveling-salesman problems, and to cases where the points are in n-space instead of the plane, or where the distribution from which the points are drawn is non-uniform, or where distances are determined by the L_1 metric instead of the euclidean metric.

4. THE CONSTRUCTION OF HAMILTON CIRCUITS, INDEPENDENT SETS AND COLORINGS IN RANDOM GRAPHS

RANDOM GRAPHS

Let $p(n)$ be a function from the positive integers into $[0,1]$. Relative to this function, we can define a random n-vertex graph as one in which the vertex set is $\{1,2,\ldots,n\}$, and in which the edge set is constructed according to the following sampling experiment: for each pair i, j, include $\{i,j\}$ as an edge with probability $p(n)$, independent of what other edges are included. We speak of the *constant density model* when $p(n)$ is a constant p, and of the *constant average degree model* when $p(n) = c/n-1$, for some constant c, and all $n > c$.

The theory of random graphs goes back to [Erdös and Renyi (59)]. The subject is rich in remarkable theorems to the effect that certain properties of a random graph can be predicted with great certainty, simply from the knowledge of $p(n)$. For example, the following theorem tells us when we can expect a random graph to be connected.

Theorem 3 [Erdös and Renyi (59)]. Let $p(n) = \ln n + c$. Then the probability that a random n-vertex graph is connected is asymptotic to $e^{-e^{-2c}}$.

Corollary 1. If $p(n) \sim \alpha \ln n$, $\alpha < 1$, then a random n-vertex graph is not connected (a.e.). If $p(n) \sim \alpha \ln n$, $\alpha > 1$, then a random n-vertex graph is connected (a.e.).

HAMILTON CIRCUITS

A *Hamilton circuit* is a cycle passing through each vertex of a graph exactly once. The problem of deciding whether a graph has a Hamilton circuit is NP-complete. Nevertheless, Posa has given a polynomial-time algorithm that almost surely finds a Hamilton circuit when one exists [Posa (75)].

Theorem 4 (Posa). Suppose $p(n) \sim \alpha \ln n$.

Case 1. $\alpha < 1$. In this case, a random graph does not have a Hamilton circuit (a.e.).

Case 2. $\alpha > 1$. Then there is a polynomial-time algorithm that finds a Hamilton circuit (a.e.)

The first case in Theorem 4 is immediate from the work of Erdös and Renyi, since a graph must be connected in order to have a Hamilton circuit. More interesting is the algorithm referred to in Case 2. The algorithm fixes a vertex (call it vertex 1) and proceeds to construct longer and longer paths starting at vertex 1. Let P be a simple path from vertex 1 to vertex k; k is called the *free end point* of P. Let $e = \{k,\ell\}$ be an edge incident with k. Then any of three operations may be possible, each yielding either a Hamilton circuit or a new path.

(a) *closure* -- If $\ell = 1$ and P contains all the vertices, then $P \cup \{e\}$ is a Hamilton circuit;

(b) *extension* -- If ℓ does not occur in P, then e may be adjoined to P to form a longer path, with ℓ as its end point;

(c) *rotation* -- If $\ell \neq 1$ and ℓ occurs in P, then there is a unique edge e' in P such that $P \cup \{e\} - \{e'\}$ is a path.

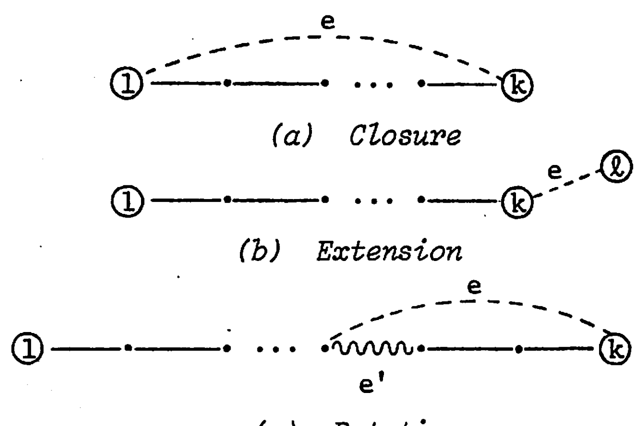

Fig. 3. Operations used in Posa's algorithm

Given a path of length i the algorithm repeatedly performs the rotation operation to obtain new paths of length i, continuing until extension to a path of length i+1 or (when i = n) closure to a Hamilton circuit becomes possible. The algorithm is constrained so that it will not form two paths of length i with the same free end point. This constraint ensures that the algorithm will run in polynomial time, but enhances the likelihood that the algorithm will stall because permissible rotations become exhausted before extension or closure can occur. Nevertheless, Posa shows that the algorithm almost surely finds a Hamilton circuit when $p(n) \sim \alpha \ln n$, $\alpha > 1$.

Experiments conducted by R. MacGregor at Berkeley indicate that the Posa algorithm is effective for practical values of n, as well as having favorable asymptotic properties. MacGregor's program generates n-vertex graphs by starting with n isolated vertices and adding edges until the graph is connected and each vertex has degree at least 2. Until that point no Hamilton circuit can exist. In each of sixty cases, with $n = 500$, the program found a Hamilton circuit at that point. On fifty-seven of the graphs the Posa algorithm succeeded immediately. In the other three cases the vertices had to be randomly renumbered, and the Posa algorithm rerun, before a Hamilton circuit was found.

INDEPENDENT SETS

A set S of vertices in a graph is *independent* if no two vertices in S are adjacent. The problem of finding a maximum-cardinality independent set in a graph is $\mathcal{NP}$-hard. No polynomial-time algorithm is known for solving this maximization problem within a fixed ratio r. In [Garey and Johnson (76a)] it is proven that the following statements are equivalent:

(a) for some r, there is a polynomial-time algorithm to solve the maximum independent set problem within the ratio r;

(b) for every $r > 1$, there is a polynomial-time algorithm for solving the maximum independent set problem within r.

We present a simple algorithm for finding independent sets, and analyze its typical behavior on random graphs. Given a graph G and a set of vertices S, define m(S) as the smallest-numbered vertex not in S, and not adjacent to any vertex in S; m(S) is undefined if no such vertex exists.

Sequential Algorithm for Finding Independent Sets

```
    S ← φ
begin while m(S) defined do
          S ← S ∪ {m(S)}
end
```

It is easy to construct graphs with n vertices for which the sequential algorithm delivers an independent set of size

1, although the largest independent set is of size n-1. By contrast, we shall show that, on random graphs, the sequential algorithm solves the maximum independent set problem within a small ratio (almost everywhere).

We analyze the behavior of the algorithm using two models of random graphs: the constant density model and the constant average degree model.

<u>Theorem 5</u>. Let $\gamma(n,p(n))$ be a random variable equal to the size of the independent set produced by the sequential algorithm applied to a random n-vertex graph.

(1) Constant-density model [Grimmett and McDiarmid (75)] For every $\epsilon > 0$

$$(1-\epsilon)\log_{\frac{1}{1-p}} n < \gamma(n,p) < (1+\epsilon)\log_{\frac{1}{1-p}} n \quad \text{(a.e.)}$$

(2) Constant average degree model. For every $\epsilon > 0$

$$(1-\epsilon)\frac{\ln(c+1)}{c}n < \gamma(n,\tfrac{c}{c-1}) < (1+\epsilon)\frac{\ln(c+1)}{c}n \quad \text{(a.e.)}$$

To determine how close the sequential algorithm comes to producing maximum independent sets, we examine the random variable $\beta(n,p(n))$, which is defined as the size of a largest independent set in a random n-vertex graph. When $p(n) = p$ and n is large, the value of the largest independent set can be pinpointed with amazing precision.

<u>Theorem 6</u> [Matula (76)]. Let

$$z(n,p) = 2\log_{\frac{1}{1-p}} n - 2\log_{\frac{1}{1-p}}\log_{\frac{1}{1-p}} n + 2\log_{\frac{1}{1-p}}\frac{e}{2} + 1 .$$

Then

$$|z(n,p) + \beta(n,p)| \leq 1 \quad \text{a.e.}$$

We can also estimate the size of the largest independent set when the constant average degree model is used. Let $H(\alpha)$ denote the entropy function:

$$H(\alpha) = -\alpha \ln \alpha - (1-\alpha)\ln(1-\alpha) .$$

<u>Theorem 7</u>. For every $\epsilon > 0$

$$\beta(n,\tfrac{c}{n-1}) \leq (1+\epsilon)\alpha n \quad \text{a.e. ,}$$

where α is determined by the relation

$$\frac{2H(\alpha)}{\alpha^2} = c .$$

Combining Theorems 5, 6 and 7, we may make the following claims about the sequential algorithm for finding independent sets.

Theorem 8.
(a) constant density model ($p(n) = p$). For every $\epsilon > 0$, the sequential algorithm solves the maximum independent set problem within $2+\epsilon$ (a.e.)
(b) constant average degree model ($p(n) = \frac{c}{n-1}$). For every $\epsilon > 0$, the sequential algorithm solves the maximum independent set problem within

$$(1+\epsilon)r(c) \quad \text{(a.e.)} ,$$

where $r(c)$ is a monotone increasing function which is less than 1.6 when $c < 1100$.

COLORINGS

A *coloring* of the graph G is a partition of the vertices of G into independent sets (called *color classes*). The *coloring problem* is to find a coloring with a minimum number of color classes; this minimum number is called the chromatic number of G, denoted $\chi(G)$.

The coloring problem is *NP*-hard. In [Garey and Johnson (76a)] it is shown that it is *NP*-hard to solve the coloring problem within a ratio $r < 2$. Moreover, no polynomial-time algorithm is known which solves the coloring problem within any fixed ratio r.

In [Johnson (74)] a *sequential algorithm* for graph coloring is considered. This algorithm simply examines the vertices in order, and places each one in the least color class that can legitimately accept it. A family of graphs is presented on which the performance of the algorithm is disastrous. A typical graph in the family has the vertex set $\{1,2,\ldots,n\} \cup \{1',2',\ldots,n'\}$ and the edge set $\{\{i,j'\} | i \neq j\}$. The chromatic number of the graph is 2, but the sequential algorithm will require n colors if the vertices are considered in the order $1,1',2,2',\ldots,n,n'$.

In [Grimmett and McDiarmid (75)] an analysis is made of the performance of the sequential algorithm on random graphs generated according to the constant density model ($p(n) = p$). Define two random variables: $\chi(n,p)$, the chromatic number of a random n-vertex graph and $\theta(n,p)$, the number of colors required by the sequential algorithm to color a random n-vertex graph. The following theorem summarizes their results.

Theorem 8. For every $\epsilon > 0$,

(a) $\chi(n,p) \geq (1-\epsilon)\frac{n}{2 \log_{\frac{1}{1-p}} n}$ (a.e.)

and

(b) $\theta(n,p) \leq (1+\epsilon)\frac{n}{\log_{\frac{1}{1-p}} n}$ (a.e.)

It follows that the sequential algorithm solves the coloring problem within $2+\epsilon$ (a.e.).

5. PROBABILISTIC ANALYSIS OF A TREE SEARCH ALGORITHM FOR THE SET COVERING PROBLEM

TREE SEARCH

Our set covering algorithm is based on a general strategy for structuring the solution of a combinatorial minimization problem as a search through a finite tree. The technique, known as implicit enumeration, is simply to break the problem recursively into cases, until all the cases become manageable.

Any instance of a combinatorial minimization problem can, in principle, be expressed as: minimize $f(x)$ subject to $x \in X$, where X, the set of feasible solutions, is a finite collection of objects such as tours, colorings or independent sets. For any $Y \subseteq X$, let $c(Y)$ denote $\min\{f(x) | x \in Y\}$. Given $\{X_i\}$, a collection of sets whose union is X, we have $c(X) = \min\{c(X_i)\}$.

Thus the original problem may be solved by splitting up the set of feasible solutions according to some criterion, and then solving the resulting subproblems, each of which has more constraints on its feasible solutions than the original problem has.

The same idea may be applied recursively to the subproblems. Thus, if $X_i = X_{i_1} \cup X_{i_2} \cup \cdots \cup X_{i_b}$, we may replace X_i by $\{X_{i_1}, X_{i_2}, \ldots, X_{i_b}\}$. This process is called *branching* or *expanding a node*. Repeated branching creates a tree of subproblems. Each path down the tree corresponds to a sequence of more and more severely constrained subproblems.

Repeated branching causes the number of subproblems to grow exponentially; thus, implicit enumeration methods for the exact solution of combinatorial optimization problems typically require exponential time. Here we explore a probabilistic approach to searching through an implicit enumeration tree. Roughly, the idea is to examine the subproblems (tree nodes) to determine their likelihood of yielding a near-optimal solution, and to avoid branching from unpromising nodes. In this way the number of nodes generated grows linearly, rather than exponentially, with problem size. Despite this severe pruning of the search tree, we show, under suitable probabilistic assumptions, that near-optimal solutions are obtained almost everywhere.

SET COVERING

We will use the set covering problem as a vehicle for exploring a probabilistic approach to implicit enumeration. An instance of the set covering problem is specified by a collection $\{S_1, S_2, \ldots, S_M\}$ of finite sets, each drawn from the set $\{1,2,\ldots,N\}$. We seek a set H of minimum cardinality subject to

$$H \cap S_J \neq \phi \ , \quad J = 1,2,\ldots,M \ .$$

The set-covering problem is *NP*-hard, and no known polynomial-time algorithm is guaranteed to solve the problem within a fixed ratio r.

The choice of a suitable probabilistic model for set covering problems is delicate. In [Gimpel (67)], a "constant density" model is proposed. The model involves two parameters: a probability p and a positive real number λ. A problem of size N has N elements, $[\lambda N]$ sets, and a fixed probability p that a given set contains any given element. It is then shown that random solutions are nearly optimal (a.e.).

Random Covering Algorithm

```
(1)  $H \leftarrow \phi$
(2)  While there exists J such that $H \cap S_J \neq \phi$ Do
        Begin Choose a random element $x \notin H$
           $H \leftarrow H \cup \{x\}$
        End
```

Theorem 9 [Gimpel (67)]. For every $\epsilon > 0$, the random covering algorithm solves the set covering problem within $1+\epsilon$ (a.e.).

We shall work with a different model, in which the construction of near-optimum solutions appears more difficult. The model involves as parameters a positive integer b and a positive real number λ. A problem of size N is specified as a $[\lambda N]$-tuple of b-element sets drawn from $\{1,2,\dots,N\}$; all such tuples are regarded as equally likely. The case $b = 2$ corresponds to the problem of finding a minimum number of vertices to cover all the edges of a random graph with N vertices and $[\lambda N]$ edges.

An implicit enumeration scheme for the covering problem may be constructed as follows. Given a covering problem (or subproblem) $\{S_1,S_2,\dots,S_m\}$, choose an index $j \in \{1,2,\dots,m\}$ at random. Let $S_j = \{X_1,X_2,\dots,X_b\}$. Since any feasible solution H must contain some element of S_j, we may replace the problem by b subproblems. In the i-th subproblem, it is assumed that $x_i \in H$; accordingly, x_i, along with all sets covered by X_i, is dropped from consideration.

Repeated branching creates a b-ary tree of subproblems. The terminal nodes of the tree correspond to cases in which all the sets have been eliminated. Solving the set covering problem amounts to finding a terminal node at a minimum distance from the root of the tree.

Define a *(m,n) subproblem* as one consisting of m b-element sets, and corresponding to a node at distance N-n from the root of the search tree (so that the sets are drawn from an n-element population). In analyzing tree search strategies we postulate that, whenever a (m,n) subproblem is first generated, it may be assumed to consist of a random m-tuple of b-element sets from an n-element population. This assumption would be exactly correct if the b subproblems generated by branching from any given subproblem were independent; we believe that the conditioning effects omitted by this assumption are of minor importance.

A *strategy* S for solving the covering problem is a rule for determining which node to expand in a partially formed tree; a strategy may take into account the pair (m,n) associated with each unexpanded node.

Theorem 10. Let S be any strategy that finds an optimal solution (a.e.). Then there is a $c > 1$ such that S expands at least c^N nodes (a.e.).

In view of this negative result, we concentrate on strategies that seek near-optimal solutions. One such strategy, called the *dovetailing strategy* $\mathcal{D}$, has remarkable properties. It is based on the concept of dominance. Let P_1 be an (m',n') subproblem, and P_2, an (m,n) subproblem. Then

P_1 *dominates* P_2 if $m' \leq m$, $n' \geq n$ and $(m',n') \neq (m,n)$. In such a case it is intuitively clear that P_1 is a more promising subproblem than P_2, and should be expanded before P_2 is expanded.

The dovetailing strategy sweeps down the search tree, expanding undominated nodes. Each sweep proceeds as follows.

```
    h ← N
    hmin ← the least n associated with an undominated
              unexpanded node
  While h ≥ hmin and there is an undominated unexpanded node
  (m,n) with n ≤ h Do
     Begin Among undominated unexpanded nodes (m,n) with
             n ≤ h, choose one for which n is maximum; let it
             be P = (m*,n*)
             Expand P
             h ← n*-1
     End
```

In each sweep, the dovetailing strategy expands at most N nodes. Also, if $Q = (m,n)$ is an unexpanded node at the beginning of a sweep then, during the sweep, some node $P = (m',n')$ is expanded such that either P dominates Q or $(m',n') = (m,n)$.

Theorem 11. Let S be any strategy. Let c and t be any positive integers. Then

$$\begin{aligned} &\Pr\{S \text{ finds a solution of cost} \leq c \text{ within } t \text{ node expansions}\} \\ &\leq \Pr\{\mathcal{D} \text{ finds a solution of cost} \leq c \text{ within } t \text{ sweeps}\} . \end{aligned}$$

Thus, up to a factor of N in running time, $\mathcal{D}$ is an optimal tree search strategy.

We present next a strategy B for which the number of nodes expanded is linear in N (a.e.), and which yields a nearly optimal solution (a.e.). The strategy B combines two aspects: bounded look-ahead and partial backtrack. The strategy involves two parameters: a positive integer L called the *look-ahead depth* and a real number $\mu > 1$ called the *selectivity*.

Given any node P in the search tree, call node Q a *L-th level descendant* of P if there is a path in the tree of length L from P to Q. Let $f(m,n,m'')$ denote the expected number of $(m'',n-L)$ subproblems arising as L-th level descendants of a random (m,n)-subproblem. Define

$m^*(m,n) = \min\{m' \mid \sum_{m''<m'} f(m,n,m'')) \geq \mu\}$.

If P is a (m,n) subproblem, Q is a (m',n-L) subproblem, and Q is a L-th level descendant of P, call Q a *good descendant* if $m' \leq m^*(m,n)$. In that case the path from P to Q is called a *good transition*. A path which is a concatenation of good transitions is called a *good path*. Note that the expected number of good descendants of P is $\geq \mu$.

Algorithm B conducts a depth-first search for a good path from the root of the search tree to a terminal node. If such a path is found then the algorithm terminates, and gives as output the feasible solution corresponding to that path. If no such path exists, let P be the end point of a longest good path. Let Q be a "best" L-th level descendant of P; i.e., Q is a (m',n-L) subproblem, where $m' > m^*(m,n)$, but each L-th level descendant of P corresponds to a (m'',n-L) subproblem with $m'' \geq m'$. Then Q is taken as a new root, and the search is resumed. Eventually a terminal node is reached, and the corresponding feasible solution is the output of the algorithm.

Theorem 12.

(a) Algorithm B, with parameters L and μ, runs in linear time (a.e.);

(b) There is a function $r(\lambda,b,L,\mu)$ such that Algorithm B, with parameters L and μ, solves the set covering problem within $r(\lambda,b,L,\mu)$ (a.e.);

(c) $\lim_{L\to\infty} r(\lambda,b,L,\mu) = r^*(\lambda,b)$, where $r^*(\lambda,b)$ is "small"; for example, $r^*(\lambda,5) \leq 1.08$, and is extremely close to 1 unless λ is huge.

Thus, Algorithm B is an efficient method of finding nearly optimal solutions to large random set covering problems in which each set has b elements. It is hoped that the "bounded look-ahead plus partial backtrack" approach will prove similarly successful in the solution of other tree search problems.

6. OPEN PROBLEMS

In this section we outline some next steps to be taken in the probabilistic analysis of combinatorial search methods.

1. Conduct empirical studies leading to more realistic models of the probability distributions from which real-life instances of combinatorial search problems are drawn.

2. Develop a reasonable probabilistic model of linear programming problems, and determine whether some variant of the simplex method yields optimal solutions in polynomial time (a.e.). Consider in particular a variant suggested by Weinberger and Yuval, in which a pivot column is chosen at random among those with negative reduced costs.

3. For some particular NP-complete problem, and some reasonable probability assumptions, prove that no polynomial-time algorithm solves the problem exactly (a.e.) unless $P = NP$.

4. Formulate a probabilistic model of directed traveling-salesman problems, and construct a polynomial-time algorithm to solve such problems (a.e.).

5. Formulate a probabilistic model of euclidean traveling-salesman problems in which the points are clustered, rather than being drawn independently from some distribution. Construct an adaptive partitioning algorithm that solves such problems within $1+\epsilon$ (a.e.), even without knowledge of the underlying distribution.

6. Prove or disprove: for every function $p(n)$ from the positive integers into $[0,1]$, the following is true (a.e.) of a random graph G:
   either (a) G is disconnected
   or (b) G has a vertex of degree 1
   or (c) The Posa algorithm finds a Hamilton circuit in G.

7. Develop results like Posa's for the Hamilton circuit problem in digraphs.

8. For the constant-density model, find polynomial-time algorithms that solve the maximum independent set and vertex coloring problems within $r < 2$ (a.e.).

9. Give a polynomial-time algorithm that finds a nearly maximum independent set in a random cubic graph (a.e.).

10. For the constant average degree model, determine how the chromatic number of a random graph is distributed. Give a polynomial-time algorithm to solve the coloring problem within some r (a.e.).

11. Analyze the behavior of Algorithm B for the model of set covering problems considered in this paper, without the use of a simplifying independence assumption.

12. Apply algorithms like our dovetailing algorithm and "bounded look-ahead with partial backtrack" algorithm

to other combinatorial optimization problems, and to heuristic search problems arising in artificial intelligence applications such as speech processing and scene analysis.

13. Consider the problem of partitioning the vertices of a 2n-vertex graph into two sets of size n, so as to minimize the number of edges running between the two sets. For the constant density or constant average degree model of random graphs, devise a polynomial-time algorithm to solve the problem within some r (a.e.).

REFERENCES

Beardwood (59) Beardwood, J., Halton, J.H. and Hammersley, J.M., "The Shortest Path Through Many Points", Proc. Camb. Phil. Soc. 55 (1959), 299-327.

Bellman (60) Bellman, R., "Combinatorial Processes and Dynámic Programming", Proc. Tenth Symp. in Applied Math., American Mathematical Society (1960), 217-250.

Christofides (76) Christofides, N., "Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem", Abstract, CMU Symp. on New Directions and Recent Results in Algorithms and Complexity (1976).

Cook (71) Cook, S.A., "The Complexity of Theorem-Proving Procedures", Proc. Third ACM Symp. on Theory of Computing (1971), 151-158.

Erdös (59) Erdös, P. and Renyi, A., "On Random Graphs, I", Publicationes Mathematicae 6 (1959), 290-297.

Garey (76) Garey, M.R., Graham, R.L. and Johnson, D.S., "Some NP-Complete Geometric Problems", Proc. Eighth ACM Symp. on Theory of Computing (1976).

Garey (76a) Garey, M.R. and Johnson, D.S., "The Complexity of Near-Optimal Graph Coloring", J. ACM (1976).

Garey (76b) Garey, M.R. and Johnson, D.S., "Performance Guarantees for Heuristic Algorithms: An Annotated Bibliography", Proc. Symp. on New Directions and Recent Results in Algorithms and Complexity, Academic Press (1976).

Gimpel (67) Gimpel, J.F., "A Stochastic Approach to the Solution of Large Covering Problems", IEEE Switching and Automata Theory (1967), 76-83.

Graham (66) Graham, R.L., "Bounds for Certain Multiprocessing Anomalies", Bell Syst. Tech. J. 45 (1966), 1563-1581.

Grimmett (75) Grimmett, G.R. and McDiarmid, C.J.H., "On Colouring Random Graphs", Math. Proc. Camb. Phil. Soc. 77 (1975), 313-324.

Held (62) Held, M. and Karp, R.M., "A Dynamic Programming Approach to Sequencing Problems", J. Soc. Indust. Appl. Math. 10 (1962), 196-210.

Johnson (74) Johnson, D.S., "Worst Case Behavior of Graph Coloring Algorithms", Proc. Fifth Southeastern Conf. on Combinatorics, Graph Theory and Computing, Utilitas Mathematica Publishing, Winnipeg (1974).

Karp (72) Karp, R.M., "Reducibility Among Combinatorial Problems", Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York (1972), 85-104.

Karp (75a) Karp, R.M., "On the Computational Complexity of Combinatorial Problems", Networks 5 (1975), 45-68.

Karp (75b) Karp, R.M., "The Fast Approximate Solution of Hard Combinatorial Problems", Proc. Sixth Southeastern Conf. on Combinatorics, Graph Theory and Computing, Utilitas Mathematica Publishing, Winnipeg (1975).

Klee (70) Klee, V. and Minty, G.J., "How Good is the Simplex Algorithm?", Mathematical Note No. 643, Boeing Scientific Research Laboratories (1970).

Levin (73) Levin, P.A., "Universal Sorting Problems" (Russian), Problemi Peredaci Informacii, Vol. IX (1973), 115-116.

Matula (76) Matula, D., "The Largest Clique in a Random Graph", to appear (1976).

Posa (76) Posá, L., "Hamilton Circuits in Random Graphs", to appear in Discrete Mathematics (1976).