Theorem 2: There exists no algorithm that decides for any arbitrarily given uniformity predicate P whether or not the predicate induces more than one segment in any image.

Proof: Let $Q(x_1, \cdots, x_m)$ be any Diophantine polynomial. Also let the predicate

$$P_Q: \{1, \cdots, t\}^k \to \{true, false\}, \qquad k \geq 2$$

be defined as follows: $P_Q(i_1, \cdots, i_k) = true$ if and only if $Q(\hat{x}_1, \cdots, \hat{x}_m) \neq 0$ for all $\hat{x}_1, \cdots, \hat{x}_m < B$, which satisfy $(\hat{x}_1, \cdots, \hat{x}_m) \neq (0, \cdots, 0)$, where $B = \sum_{j=1}^{k} i_j$. (Note that the value of $Q(0, \cdots, 0)$ might be of any magnitude.) Then there exists a picture with more than one segment with respect to P_Q if and only if $Q(x_1, \cdots, x_m)$ has a nontrivial [i.e., $\neq (0, \cdots, 0)$] nonnegative integer solution $(\hat{x}_1, \cdots, \hat{x}_m)$. The proof of this last statement follows.

For the "if" direction of the proof, assume that

$$Q(x_1, \cdots, x_m) = 0$$

has a nontrivial nonnegative integral solution. Then let $A = \min \{\hat{y} \mid \hat{y} = \max \{\hat{y}_1, \cdots, \hat{y}_m\},\ \hat{y} > 0,\ Q(\hat{y}_1, \cdots, \hat{y}_m) = 0$, and $\hat{y}_1, \cdots, \hat{y}_m$ are nonnegative integers$\}$. Now consider the picture f_A defined as follows:

$$f_A(l) = \begin{cases} 0 & \text{if } A = 1 \text{ and } l = 1 \text{ or } 3 \\ 1 & \text{if } A = 1 \text{ and } l = 2 \text{ or } 4 \\ 1 & \text{if } A \geq 2 \text{ and } 1 \leq l \leq 2A \\ \text{undefined} & \text{otherwise.} \end{cases}$$

By assumption $Q(x_1, \cdots, x_m) = 0$ has a nonnegative integral solution different from $(0, \cdots, 0)$, and therefore $A > 0$. If $A = 1$ then $P_Q(f_A(1), f_A(2)) = P_Q(f_A(3), f_A(4)) = P_Q(0, 1) = true$, but $P_Q(f_A(1), f_A(2), f_A(3), f_A(4)) = false$. Thus, f_A must have two segments with respect to P_Q. (Recall that each segment must have a grid of size 2 at least.) On the other hand, if $A \geq 2$ then $P_Q(f_A(s_1), \cdots, f_A(s_2)) = true$ if and only if $1 \leq s_1 < s_2 \leq 2A$ and $s_2 - s_1 < A$. Thus, in such a case f_A must have a segmentation of at least cardinality two with respect to P_Q. One such possible segmentation is the one in which one of the segments has the grid $(1, \cdots, A)$ and the other has the grid $(A + 1, \cdots, 2A)$.

To show the "only if" direction of the proof assume that there exists a linear picture f with segmentation of cardinality greater than one with respect to P_Q. Let B_1 and B_2 be the sums of the values of f at the grid points of any two adjacent such segments, respectively. Then by the definition of P_Q the equation $Q(x_1, \cdots, x_m) = 0$ has no nontrivial integral solution $(\hat{x}_1, \cdots, \hat{x}_m)$ that satisfies $\hat{x}_1, \cdots, \hat{x}_m < \max \{B_1, B_2\}$. On the other hand, the equation has at least one such solution $(\hat{y}_1, \cdots, \hat{y}_m)$ that satisfies

$$\max \{B_1, B_2\} \leq \max \{\hat{y}_1, \cdots, \hat{y}_m\} \leq B_1 + B_2$$

because P_Q is false when considered for the union of the grids of these two segments. Moreover, both B_1 and B_2 must be greater than zero because otherwise P_Q would have been true for the union of the grids of these two segments.

The result now follows because $Q(x_1, \cdots, x_m) = 0$ has a nonnegative integral solution if and only if $Q(0, \cdots, 0) = 0$ or P_Q induces a segmentation of cardinality greater than one for some picture. □

III. Conclusions

The segmentation of (textured) images is of major concern today for scene analysis. The usual approach is an ad hoc one, i.e., using heuristic methods. Given some particular image(s), one would choose some predicate, assumed to be uniform and attempt to segment. Basically, this approach is domain specific and it is probably unsuitable even for only one domain of pictures. The whole field of digital image processing is still trying to grow and mature and there is no formal knowledge regarding what tasks are feasible and if feasible how expensive they might be. This correspondence provides some basic results regarding the complexity of two problems within the general context of image segmentation. We would like to hope that some taxonomy for the general field of digital image processing could be defined and that each problem within such a classification could be characterized according to its complexity. Then any proposal for some specific scene analysis procedure will be more realistic.

References

[1] O. D. Faugeras, "An overview of probabilistic relaxation theory and applications," in *Digital Image Processing and Analysis*, J. C. Simon, Ed. The Netherlands: Reidel, 1981.

[2] R. Fowler, M. S. Paterson, and S. Tanimoto, "The complexity of packing and covering in the plane and related intersection graph problems," Dep. Comput. Sci., Univ. of Washington, Seattle, TR 80-05-02, May 1980.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability–A Guide to the Theory of NP-Completeness.* San Francisco, CA: W. H. Freeman, 1979.

[4] K. Gödel, "Uber formal unentscheibare sätze der principia mathematica and verwandter systeme," *Monatschefte Math. Phys.*, vol. 38, pp. 173-198, 1931.

[5] A. R. Hanson, C. C. Parma, and E. M. Hanson, "Experiments in schema-driven interpretation of a natural scene," in *Digital Image Processing and Analysis*, J. C. Simon, Ed. The Netherlands: Reidel, 1981.

[6] R. M. Haralick, "Statistical and structural approaches to textures," *Proc. IEEE*, vol. 67, pp. 786-804, May 1979.

[7] R. M. Haralick and L. G. Shapiro, "The consistent labeling problem: Parts I & II," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 173-184, Apr. 1979, and vol. PAMI-2, pp. 193-203, May 1980.

[8] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley, 1978.

[9] D. Hilbert, "Mathematische probleme vortrag, gehalten auf dem int. math.–Kongrass zu Paris 1900," in *Nachr. Akad. Wiss. Gottingen Math.-Phys.*, 1900, pp. 253-297; also in *Bull. Amer. Math. Soc.*, vol. 8, pp. 437-479, 1901-1902.

[10] Y. Matijasevic, "Enumerable sets are diophantine," *Dokl. Akad. Nauk SSSR*, vol. 191, pp. 279-282, 1970.

[11] T. Pavlidis, *Structural Pattern Recognition.* New York: Springer-Verlag, 1977.

[12] A. Rosenfeld and A. Kak, *Digital Picture Processing.* New York: Academic, 1976.

[13] J. M. Tenebaum *et al.*, "Prospects for industrial vision," SRI Int., TR 175, Nov. 1978.

[14] H. Wechsler, "Texture analysis–A survey," *Signal Processing*, vol. 2, pp. 271-282, 1980.

A Counterexample to a Diameter Algorithm for Convex Polygons

BINAY K. BHATTACHARYA AND GODFRIED T. TOUSSAINT

Abstract–**Recently, Snyder and Tang [1] proposed an algorithm for finding the diameter of a convex polygon. In this note a family of convex polygons is described for which their algorithm fails. It is also**

Manuscript received February 12, 1981; revised October 22, 1981.

The authors are with the School of Computer Science, McGill University, Montreal, P.Q., Canada.

pointed out that the diameter of an *arbitrary* simple n-vertex polygon can be computed in $O(n)$ time.

Index Terms—**Algorithm, artificial intelligence, computational complexity, computational geometry, convex hull, convex polygon, image processing, pattern recognition, region growing, scene analysis, simple polygon.**

I. Introduction

In this note we assume that we are given a simple polygon $P = (p_1, p_2, \cdots, p_n)$, i.e., we are given a list of vertices in order, along with their Cartesian coordinates. We assume the polygon is in *standard form*, i.e., the vertices are distinct and no three consecutive vertices are collinear. A pair of vertices p_ip_{i+1} defines an edge of the polygon where $i = 1, 2, \cdots, n$ and $p_{n+1} = p_1$. If for every vertex p_k the angle determined by the edges $p_{k-1}p_k$ and p_kp_{k+1} is convex, then the polygon is *convex.*

Recently, Snyder and Tang [1] proposed several approximate and exact algorithms of varying complexity for finding the diameter of arbitrary simple polygons. The *diameter* of P, denoted by $D(P)$, is defined as follows:

$$D(P) = \max_{i,j} \{d(p_i, p_j)\}$$

where $d(p_i, p_j)$ is the Euclidean distance between vertices p_i and p_j. One of their $O(n)$ algorithms is claimed to give an exact answer when the polygons are convex. Since their algorithm is short and simple we paraphrase it below in full in the interest of completeness.

Algorithm DIAMCON

Step 1: Select an arbitrary starting vertex; call it p_0.

Step 2: Perform a linear search of the vertices of P, testing adjacent vertices in turn, searching for the vertex with a maximum distance from p_0. Call the new vertex p_1.

Step 3: With p_1 as an "anchor" point, starting at p_0, search clockwise or counterclockwise for the vertex furthest from p_1. This search is carried out only in the direction in which the distance is increasing. If the distance decreases in both directions, EXIT with p_0 and p_1 as the extrema determining the diameter; ELSE continue.

Step 4: Find a new point p_2, such that $d(p_1, p_2) > d(p_1, p_0)$, and assign $p_0 \leftarrow p_1$, $p_1 \leftarrow p_2$; then GO TO Step 3.

The manner in which the algorithm works is illustrated in Fig. 1, where the algorithm starts at vertex "1" and finishes with the correct answer at vertices "3" and "4." In the next section we describe a class of convex polygons for which algorithm DIAMCON fails.

II. A Class of Counterexamples

Consider the Euclidean plane E^2. Given two points p_i, p_j, $i \neq j$ in E^2, the disk of p_i and p_j, denoted by DISK (p_i, p_j) is that subset of E^2 defined as follows:

$$\text{DISK}\,(p_i, p_j) \equiv \{x\colon\ [d^2(x, p_i) + d^2(x, p_j)]^{1/2} \leqslant d(p_i, p_j)\}$$

where d, as before, denotes Euclidean distance. Clearly, p_i and p_j lie on a circle and determine its diameter.

Another subset of E^2 (a superset of DISK (p_i, p_j)) denoted by LUNE (p_i, p_j) is defined as the intersection of the interior of two circles, each with radius $d(p_i, p_j)$, centered at p_i and p_j. More formally,

$$\text{LUNE}\,(p_i, p_j) \equiv \{x\colon\ \max[d(x, p_i), d(x, p_j)] < d(p_i, p_j)\}.$$

Let $LD(p_i, p_j)$ be defined as

$$LD(p_i, p_j) = \text{LUNE}\,(p_i, p_j) \sim \text{DISK}\,(p_i, p_j)$$

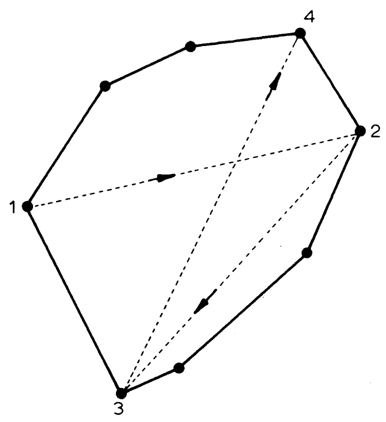

Fig. 1. A convex polygon illustrating how the algorithm of Snyder and Tang [1] works.

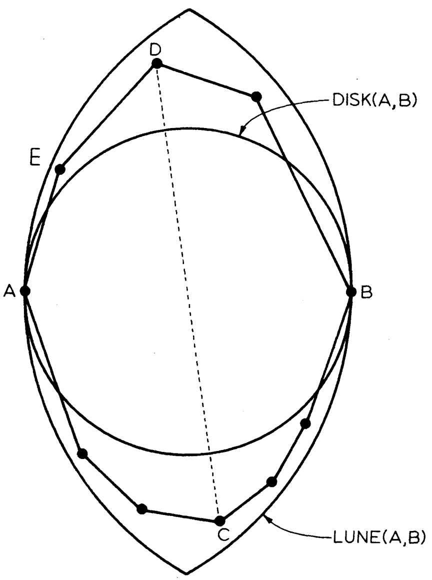

Fig. 2. A convex polygon for which the algorithm of Snyder and Tang [1] fails.

where $\sim$ denotes set difference. In other words $LD(p_i, p_j)$ is the region in between the lune and the disk.

Definition: A convex polygon P is said to be *lunar* if there exist two vertices $p_i, p_j \in P$ such that all the remaining vertices lie in $LD(p_i, p_j)$. These vertices will be referred to as *lunar vertices.*

It is now easy to see that algorithm DIAMCON will fail if it operates on a *lunar* polygon, selects a *lunar* vertex in Step 1, and the diameter is not determined by the pair of lunar vertices. Fig. 2 illustrates one such counterexample. If the algorithm selects either vertex A or vertex B in Step 1, it will exit with A and B as the extrema and $d(A, B)$ as the diameter, clearly a much smaller distance than the true diameter $d(C, D)$. One might wonder with what probability the algorithm would select a *lunar* vertex in Step 1 if a vertex was chosen at random with probability $1/n$. This requires knowing how many vertices of a *lunar* polygon can be *lunar.* In the Appendix it is shown that a *lunar* polygon has no more than two *lunar* vertices and thus the probability is only $2/n$. This is no consolation, however, because while the above conditions for algorithm DIAMCON to fail are sufficient, they are by no means necessary. The algorithm may still fail if it operates on a *lunar* polygon and selects a *nonlunar* vertex in Step 1. To see this insert a new vertex $A' \in LD(A, B)$ an ϵ-distance away from A

and refer to Fig. 2. A' is not *lunar* since $A \notin$ LUNE(A', B). Yet if the algorithm starts at A' it incorrectly exits with A, B as the diameter pair. Neither is the algorithm guaranteed to work if P is *nonlunar.* Consider a polygon consisting of the vertices A, C, B, D', E in Fig. 2, where D' is located just outside LUNE(A, B) on the line through C, D, and E is adjacent to A and D and lies in $LD(A, B)$. This polygon is *nonlunar* and yet if the algorithm starts at A it will again yield the incorrect answer A, B.

III. Concluding Remarks

In image processing researchers in the past have used either the brute-force $O(n^2)$ method [5] or an *approximate* $O(n)$ algorithm [6] for computing the diameter of an n-vertex simple polygon [7]. However, the tools are available today for computing the *exact* diameter of a simple polygon in $O(n)$ time. While Dobkin and Snyder [3] present an $O(n)$ algorithm for computing the *exact* diameter of a *convex* polygon, it is shown in [8] that their algorithm can also fail. However, Shamos [2] presents an $O(n)$ algorithm which appears to work for all convex polygons. For a *simple* polygon McCallum and Avis [4] give an $O(n)$ algorithm for obtaining its convex hull. Once the hull is obtained the algorithm in [2] can be used for a total running time of $O(n)$. This approach should be quite useful in scene analysis using *regions* [1] since *regions* are simple polygons that can have a very large number of vertices. Thus the extra overhead in running time incurred by the algorithms in [2] and [4], over and above the brute-force method, should be more than compensated for by the fact that they run in linear rather than quadratic time.

Appendix

The following notation is used in the lemmas and theorem below. The boundary of a set such as DISK(a, b) is denoted by bd(DISK(a, b)). The closed line segment between two points a, b is denoted by $[a, b]$; the open line segment by (a, b). The x and y coordinates of a point p are denoted by x_p and y_p, respectively.

Lemma A1: Given two points a, b in E^2, if two additional points p, q are placed in such a manner that

1) one of them lies on one side of the line L through a and b, and the other lies on the opposite side of L,

2) the line segment $[p, q]$ intersects (a, b), and

3) both p and q lie in the exterior of DISK(a, b), then it follows that at least one of the two points a or b must lie in the interior of DISK(p, q).

Proof: Without loss of generality it is assumed that L is the horizontal line, p lies above L, and the intersection of $[p, q]$ and $[a, b]$ lies to the left of c, the center of DISK(a, b). Refer to Fig. 3. Since p and q lie outside DISK(a, b), angle $(apb) < 90^\circ$ and angle $(aqb) < 90^\circ$. Hence, angle (apb) + angle $(aqb) < 180^\circ$. Since the sum of the four internal angles of the quadrilateral $apbq$ equals 360°, angle (qap) + angle $(pbq) > 180^\circ$. This implies that either angle (qap) or angle (pbq) is greater than 90°. Hence, one of the points (a or b) lies inside DISK(p, q). Q.E.D.

Let the half-lune of two points a and b, denoted by HLUNE(a, b), be defined as the intersection of LUNE(a, b) with the open half-plane determined by the line L through a and b. We then have the following lemma where again we assume without loss of generality that L is the horizontal line, and HLUNE(a, b) lies above L.

Lemma A2: Given two points p and q that lie in HLUNE(a, b), if a lies in LUNE(p, q), then the line segment $[p, q]$ must have negative slope.

Proof: That $[p, q]$ can be neither horizontal nor vertical is obvious from Fig. 4. Consider then the case when the slope of $[p, q]$ is positive. Without loss of generality we assume p dominates q and refer to Fig. 4. Three possibilities exist for

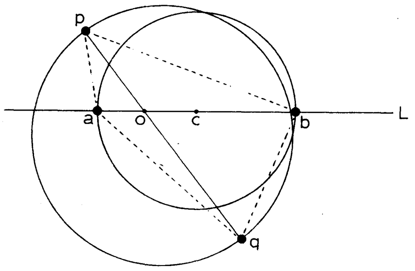

Fig. 3. Illustrating the proof of Lemma A1.

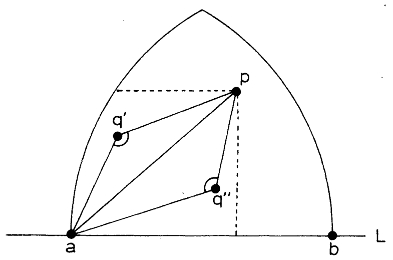

Fig. 4. Illustrating the proof of Lemma A2.

q: either it lies on (a, p), above (a, p), or below (a, p). In the first case $d(p, a) > d(p, q)$, and therefore a lies outside LUNE(p, q), a contradiction. In the second and third cases q' and q'' illustrate the possible corresponding locations for q. In both cases pqa forms a triangle in which the interior angle $pqa > 90^\circ$. Thus, it follows that $d(a, p) > d(q, p)$ and a lies outside LUNE(p, q), a contradiction. Q.E.D.

Theorem: A *lunar* polygon has no more than two *lunar* vertices.

Proof: Let P be a lunar polygon. By definition P must contain two lunar vertices a and b such that all the remaining vertices lie in $LD(a, b)$. It must be shown that: 1) no single vertex p, other than a and b, can form a *lunar* pair with either a or b, and 2) no two additional vertices p, q such that $p \neq a$, b, $q \neq a, b$ can form a *lunar* pair. Let the line L through a, b be the horizontal line and let it divide the sets of interest into upper and lower half-sets. In other words, let $UHLD(a, b)$ denote the intersection of $LD(a, b)$ with the open upper half-plane determined by L. Similarly, let $LHLD(a, b)$ denote the lower half of $LD(a, b)$.

Case 1: Consider a vertex p of P such that $p \neq a, b$. By definition, $P \in LD(a, b)$. Therefore, $d(p, a) < d(a, b)$ and b lies outside LUNE(a, p). Therefore, p, a cannot form a *lunar* pair. Similarly, $d(p, b) < d(a, b)$ and a lies outside LUNE(b, p). Therefore, p, b cannot form a *lunar* pair.

Case 2: Consider two vertices p, q, $p \neq a, b$, $q \neq a, b$. By definition, $p, q \in LD(a, b)$. Two subcases arise: 1) one vertex lies in $UHLD(a, b)$ and the other in $LHLD(a, b)$, or 2) both lie in the same $HLD(a, b)$. Consider subcase 1) and without loss of generality assume $p \in UHLD(a, b)$ and $q \in LHLD(a, b)$. It follows that $[p, q]$ intersects (a, b) and by Lemma A1 either a or b must lie in DISK(p, q). Therefore, p, q cannot be a *lunar* pair in this case. For subcase 2) assume without loss of generality that both $p, q \in UHLD(a, b)$. We will show that both a and b cannot simultaneously lie in LUNE(p, q). Assume that $a \in$ LUNE(p, q). Since $UHLD(a, b) \in UH$LUNE(a, b) it follows from Lemma A2 that $[p, q]$ must have negative slope. Without loss of generality let $x_p < x_q$ and let $y_p > y_q$ and refer to Fig. 5. Vertices p, q, b form a triangle where angle $pqb > 90^\circ$. Therefore, $d(p, b) > d(p, q)$ implying that b must lie outside LUNE(p, q). Therefore, p and q cannot form a *lunar* pair and this completes the proof of the theorem.

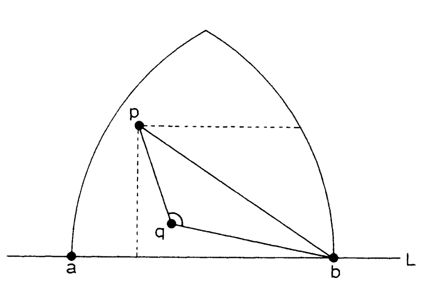

Fig. 5. Illustrating the proof of the theorem.

REFERENCES

[1] W. E. Snyder and D. A. Tang, "Finding the extrema of a region," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 266-269, May 1980.

[2] M. I. Shamos, "Geometric complexity," in *Proc. 7th ACM Symp. Theory of Comput.*, May 1975, pp. 224-233.

[3] D. P. Dobkin and L. Snyder, "On a general method for maximizing and minimizing among certain geometric problems," in *Proc. 20th Annu. Symp. Foundations Comput. Sci.*, San Juan, Puerto Rico, Oct. 1979, pp. 9-17.

[4] D. McCallum and D. Avis, "A linear algorithm for finding the convex hull of a simple polygon," *Inform. Process. Lett.*, vol. 9, pp. 201-206, Dec. 1979.

[5] J. Jacobsen, "Biomedical computer analysis of cells," in *Proc. 17th Annu. Conf. Eng. in Med. Biol.*, 1964, p. 117.

[6] M. A. Fischler, "Fast algorithms for two maximal distance problems with applications to image analysis," *Pattern Recog.*, vol. 12, pp. 35-40, 1980.

[7] G. T. Toussaint, "Pattern recognition and geometrical complexity," in *Proc. 5th Int. Conf. Pattern Recog.*, Miami Beach, FL, Dec. 1980, pp. 1324-1347.

[8] D. Avis, G. T. Toussaint, and B. K. Bhattacharya, "On the multimodality distances in convex polygons," *Comput. Math. Appl.*, to be published.

Comments on "A Counterexample to a Diameter Algorithm for Convex Polygons"

W. E. SNYDER AND D. A. TANG

In our paper [1] we described an algorithm which efficiently found the diameter of arbitrary regions, convex or concave, simple or not, filled (boundary not identified) or not. In our enthusiastic desire to describe this very general and very efficient algorithm, we described, for completeness, some more simple algorithms to handle some of the more simple and "uninteresting" (to us) cases, such as convex polygons.

Unfortunately, one of those "more simple" cases was in fact not so simple, and Bhattacharya and Toussaint[1] have very accurately pointed out a condition under which one of the simple techniques we described fails. We are very grateful to them for that observation. It should be noted, however, that this flaw does not propagate to our general algorithm, which converges to the diameter for arbitrary regions.

As one further note, it has been our experience that it is unreasonable to assume the boundaries of regions in digital pictures form simple polygons, since the sampling process often results in collinear points.

Manuscript received November 16, 1981.

The authors are with the Department of Electrical Engineering, North Carolina State University, Raleigh, NC 27607.

[1] B. K. Bhattacharya and G. T. Toussaint, this issue, pp. 306-309.

REFERENCES

[1] W. E. Snyder and D. A. Tang, "Finding the extrema of a region," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 266-269, May 1980.

An Empirical Comparison of Backtracking Algorithms

CYNTHIA A. BROWN AND PAUL W. PURDOM, JR.

Abstract—**In this paper we report the results of experimental studies of zero-level, one-level, and two-level search rearrangement backtracking. We establish upper and lower limits for the size problem for which one-level backtracking is preferred over zero-level and two-level methods, thereby showing that the zero-level method is best for very small problems. The one-level method is best for moderate size problems, and the two-level method is best for extremely large problems. Together with our theoretical asymptotic formulas, these measurements provide a useful guide for selecting the best search rearrangement method for a particular problem.**

Index Terms—**Backtracking, constraint satisfaction, search algorithms, tree search.**

I. INTRODUCTION

An important task of computer scientists is devising general algorithms that can be used to solve any problem from a large set of related problems. Such sets of problems can be divided into two classes, sometimes called "easy" and "hard." The easy sets are those for which each problem in the set can be solved within a time which is a polynomial function of the problem size. An example of such an easy set is the computation of the shortest path between two nodes in a graph. An individual problem in the set consists of a graph whose arcs have nonnegative labels and a distinguished pair of vertices. There are well-known general methods for solving any problem in this set in a time proportional to the square of the number of nodes in the graph [7]. For naturally occurring easy problem sets, the degree of the polynomial time bound is usually no greater than three, so rapid solution of large problems is possible.

A hard problem set is one for which the best known algorithm takes more than polynomial time for some sequence of problems in the set. (For some hard problem sets, there are solution methods with small average time.) Many important hard problem sets are NP complete. Garey and Johnson [8] give a thorough discussion of the NP complete class and list many NP complete problem sets.

An examination of the problems listed by Garey and Johnson shows that most of them have a natural representation as a predicate of the form

$$P \equiv \bigwedge_{1 \leqslant i \leqslant m} R_{i}(w_1, \cdots, w_v) \qquad (1)$$

where each R is a relation that is simple in the sense that it depends on only a few of the variables and each w has a finite

Manuscript received March 25, 1981; revised November 3, 1981. This work was supported in part by the National Science Foundation under Grant MCS 7906110.

The authors are with the Department of Computer Science, Indiana University, Bloomington, IN 47405.