AN EXTREMELY FAST MINIMUM SPANNING CIRCLE ALGORITHM[+]

B. John Oommen

SCS-TR-65
November 1984

[+] School of Computer Science, Carleton University, Ottawa, Ontario, Canada. K1S 5B6

This research project was partially supported by the Natural Sciences and Engineering Research Council of Canada.

AN EXTREMELY FAST MINIMUM SPANNING CIRCLE ALGORITHM+

B. John Oommen
School of Computer Science
Carleton University
Ottawa, K1S 5B6
CANADA

Abstract

We consider the century old problem of finding the Minimum Spanning Circle (MSC) of N points in the plane. We propose a computational scheme which incorporates the ideas which motivate three of the best algorithms known. The technique converges in finite time and is extremely fast.

KEYWORDS: Minimal Spanning Circle, Sylvester's Minmax Location Problem, Chrystal-Pierce Algorithm

+ Partially supported by the Natural Sciences and Engineering Research Council of Canada.

I. INTRODUCTION

Let S be a set of N points in the plane. It is required to find the circle of minimum radius which encloses all the points. We shall refer to this circle as the Minimum Spanning Circle (MSC) of the set S. Obviously, the circle is unique. Let its center and radius be Θ^* and R^* respectively.

The MSC problem was first presented by Sylvester in 1857 [9]. Various algorithms have been proposed since then [2-10]. For the sake of brevity we refer the reader to an excellent paper by Hearn and Vijay [7] which surveys the literature and qualitatively compares the various algorithms which solve the problem.

There are three main properties of the MSC which motivate the three families of algorithms reported in the literature. To view our algorithm in the right perspective, we shall first consider these properties and briefly describe the principles that underly the corresponding families.

The first property of the MSC is that if it touches exactly two points in S, then these two points must lie on the diameter of the circle. However, if the MSC touches exactly K points ($K \geq 3$), then there exists three points among these K which form a non-obtuse angled triangle. The MSC is the circumcircle of this triangle.

This property of the MSC led to the first algorithm which computed it. Amazingly enough, the solution was independently discovered in the late 1800's by Sylvester, Chrystal and Pierce, and was proposed as algorithmically as one could expect [4-7]. The algorithm bears the name of the latter scientists, and has been extensively studied. It converges in finite time.

We now consider a rather interesting property of any spanning circle. This property is that any spanning circle is entirely defined by its center. Indeed this is true, for, if we are given any point Θ, the circle with center Θ and radius R spans S, where R is the distance of the fartherest point from Θ. Using this property Jacobsen [8] proposed an algorithm which converged asymptotically to Θ^*, the centre of the MSC. In every iteration, he used the principles of nonlinear programming to decide on a set of feasible directions along which the center of the circle could be moved so as to diminish the radius of the spanning circle.

The third interesting property of the MSC was the one recently reported by Castells and Melville [2]. Let $G_{A,R}$ be the circle with center A and radius R. Further, let

$$D_R = \bigwedge_{A \varepsilon S} G_{A,R} \tag{1}$$

Castells and Melville observed that D_R obeys the following relationship.

$$\begin{aligned} D_R &= \emptyset, \quad \text{the null set}, && \text{if } R < R^* \\ &\supseteq \{\Theta^*\} \quad , && \text{if } R > R^* \\ &= \{\Theta^*\} \quad , && \text{if } R = R^* \end{aligned} \tag{2}$$

Note that this says that the MSC is completely defined by its radius, R^*. Using this property, the authors of [2] have devised an "unusual" (but ingenious) algorithm which starts with any initial radius R=a and processes the region D_a to yield an updated radius R = b $\leq$ a for which $D_b \subseteq D_a$. This algorithm, too, converges asymptotically.

In this paper, we propose an algorithm which uses all these three approaches simultaneously. However, it retains the convergences property of the Chrystal-Pierce algorithm in that it always converges in finite time. Our solution is quite similar to that due to Chrystal-Pierce. However, as opposed to the latter, we do not merely use the information contained in the angles between the lines joining the points in S. In every iteration of the algorithm we also utilize information contained in the distance between the points and thus attempt to maximize the rate of convergence.

In the next section we shall present the Chrystal-Pierce algorithm in its most refined form. We shall then proceed to propose our solution.

II. THE CHRYSTAL-PIERCE ALGORITHM

Although Sylvester, Chrystal and Pierce were the ones who first proposed a solution to the problem, the algorithm which bears their name today, is a refined version of their solution. The main lemmas that led to their solution are two geometrical results related to the MSC of a triangle. For the sake of brevity we shall merely state the results.

Lemma I

Let ABC be any acute triangle. Then there exists no circle G containing the points A, B and C, smaller than the circumcircle of ABC.

Lemma II

Let $S = \{P_i \mid 1 \leq i \leq N\}$. Let G be the circumcircle of $P_1P_2P_3$, which has all the other points of S interior to the circle. If $\angle P_1P_2P_3$ is obtuse and there exists a $P_s \in S$ such that $\angle P_1P_sP_2 < \angle P_1P_2P_3$, then there exists a circle G_1 smaller than G which spans S.

Collary of Lemma II

Let $S_1 = S - \{P_1, P_3\}$, and let,

$$\angle P_1P_sP_3 = \min_{P_j \in S_1} \left[\angle P_1P_jP_3 \right]$$

If $\angle P_1P_2P_3 \geq \angle P_1P_sP_3 \geq \pi/2$, , G_1 is the circle whose diameter is P_1P_3.

The Chrystal-Pierce Algorithm uses the above results extremely efficiently. Starting with any two points A and B, the algorithm computes the point C which minimizes the $\angle$ ADB over all D in S-{A,B}. If the resulting triangle ABC is non-obtuse, the circumcircle of ABC is the required circle. If the triangle is obtuse, and the angle ACB is obtuse, the circle with diameter AB is the required circle. If neither of these cases are true, the point which has the obtuse angle (say, B) is eliminated from all further computations and the procedure repeats using the edge AC.

Chakraborty and Chandhuri improved the above procedure by suggesting a technique to obtain the best pair of points to start the whole process. Further, they observed that after any iteration, we can eliminate from all further computations not only the obtuse vertex, B, but also all $Q \in S-\{A,C\}$ for which $\angle AQB > \pi/2$.

We shall now algorithmically present the Chrystal-Pierce Solution. We assume we have a procedure GetFirstTwo, which has as its input the set S and yields as its output the best two vertices A and B which are to be used to start the algorithm. We also assume that the user may be satisfied with an approximation of the MSC. In other words, rather than wait till the algorithm halts and yields Θ^*, and R^* which are the centre and radius of the MSC respectively, the user may be satisfied with a circle of radius R^+ whose center is Θ^+, where R^+ is sufficiently close to R^* (i.e., the relative value of the change in radius is not significant). A Boolean variable Satisfied is set to TRUE if the user is satisfied with the current solution, and the algorithm terminates. With regard to eliminating a point in subsequent computations, we shall "mark" the vertex to be eliminated in the current iteration and perform all subsequent computations only with "unmarked" points.

ALGORITHM CHRYSTAL-PIERCE

Input: The set of points, S

Output: The MSC of S. The user has the capability of deciding on the accuracy of the solution, if he does not want the exact solution.

Assumption: Available to the user is a procedure GetFirstTwo, whose input is S and whose output is the best two vertices, A and B according to the Chakraborti-Chaudhuri criterion.

Method:

```
Call GetFirstTwo (S,A,B)
Unmarked = S
Done = Satisfied = FALSE
REPEAT UNTIL ((Done) OR (Satisfied))
   C = point which minimizes $\angle$ ADB for all D$\in$Unmarked-{A,B}
   Unmarked = Unmarked - {D | $\angle$ ADB > $\pi/2$}
   If $\angle$ ACB is obtuse THEN
      Done = TRUE
      MSC = Circle with AB as diameter
   ELSEIF triangle ABC is acute THEN
      Done = TRUE
      MSC = circumcircle of ABC
   ELSE
      Done = FALSE
      MSC = circumcircle of ABC
      recompute Satisfied
      IF $\angle$ ABC is obtuse THEN
         Unmarked = Unmarked - {B}
         B = C
      ELSE
         Unmarked = Unmarked - {A}
         A = C
      ENDIF
   ENDIF
ENDREPEAT
Print Out $\Theta$ and R of the MSC
END
```

END ALGORITHM CHRYSTAL-PIERCE

The algorithm requires at most O(N) computations per iteration. Further, since in every iteration we eliminate at least one vertex the exact MSC is obtained in at most N iterations. Experimentally [7], it is seen that for randomly generated points the algorithm is O(N).

III. AN ALGORITHM WHICH USES DISTANCE AND ANGLE INFORMATION

Any one who has worked with the Chrystal-Pierce algorithm will know that these scientists optimally used the information that they computed -- namely the angles between the vertices (or the cosines of the angles between the vertices [3]). However, there is a lot of information that is latent in the distances between the vertices.

We can ask ourselves: What is the main drawback of the Chrystal-Pierce algorithm? Indeed, the fact that if the edge AB was used in any iteration, the next iteration was <u>constrained</u> to use either of the two vertices A or B. Thus, if $\angle$ ABC was obtuse, the vertex B was eliminated and the procedure continued with the edge AC. That is to say that even though there may have been another edge XC which could have been used in the next iteration, the algorithm did not use it, nor did it concern itself with the problem of getting an edge "better" than AC.

By processing the information contained in the distance between the vertices, we shall try to find the "best" edge that has to be used in the next iteration. The way we tackle the problem is as follows.

Let Θ and R be the centre and radius of the circle passing through A, B and C. We shall merely consider the case when the triangle ABC is obtuse, and $\angle$ ACB is acute. In this case, the algorithm has to proceed for at least one more iteration. With no loss of generality, in the following discussions we assume that B is the obtuse angle. This means that if BB' is the diameter of the circle, A will be on one side of BB' and C will lie on the other (see Figure I). We shall process the set of points in S-{A,B} and attempt to obtain a circle of radius R centered at Θ' which touches exactly two points in S, and yet spans S. The edge joining the latter two points is used as the starting edge for the subsequent iteration.

Theorem I

Let Θ and R be the center and radius of the circumcircle of the obtuse triangle ABC, whose obtuse angle is $\angle$ ABC. Let BB' be the diameter of the circle. Let YZΘ be the arc of the radius R drawn from C. If Θ' is any point on YZΘ, the circle with center Θ' and radius R will touch the vertex C, and have A and B inside it.

Proof

Consider Figure I. Since Θ is the center of the circumcircle of ABC,

$$|\Theta A| = |\Theta B| = |\Theta C| = R.$$

Let YZΘ and VΘW be the arcs of radii R drawn from C and A respectively. Let the tangents to these arcs be PΘQ and SΘT respectively. Since the tangents to the arcs are normal to the corresponding arcs, Θ' will definitely lie inside the arc VΘW. Thus $|\Theta' A| < R$. Arguing in a similar way we can show that $|\Theta' B| < R$. This proves the theorem.

2. The above construction of finding the vertex of minimum perturbation can be viewed as an augmentation of Jacobsens [8] technique and the one due to Castells and Melville [2]. By Theorem I, we move Θ in the feasible direction along the arc ΘZY (see Figure II). Further we move it to the point V_X. Let us consider the arc ΘV_X as the set intersection of various arcs as,

$$\mathrm{Arc}(\Theta V_X) = \bigwedge_{\substack{D \varepsilon \mathrm{Semicircle} \\ BCB'}} \left[\mathrm{Arc}(\Theta V_D) \right] \tag{3}$$

Then, if $\Theta = \Theta^*$ and $R = R^*$ (the center and radius of the MSC),

$$\bigwedge_{\substack{D \varepsilon \mathrm{Semicircle} \\ BCB'}} \left[\mathrm{Arc}(\Theta^* V_D) \right] = \{ \Theta^* \} \tag{4}$$

which is remarkably identical to the formulation of Castells and Melville [2], given in equations (1) and (2).

3. Let us suppose we perturb Θ to the new point V_X. If the circle centered at V_X and radius R encloses <u>all</u> the points in the semicircle BAB', (and not just A and B), then, clearly, rather than work with the edge AC, we can more productively work with the edge CX. This is because of the fact that whereas the circle with center Θ and radius R touches <u>three</u> points in S, the circle with center V_X and radius R touches <u>exactly two</u> points, namely X and C, and yet spans S.

4. Let X be the vertex of minimum perturbation with respect to C. Similarly, let Y be the vertex of minimum perturbation with respect to A. If the circle of radius R centered at V_X

spans all the points in the semicircle BAB', the edge used in the next iteration is CX. If this circle does not span all of the points in BAB', we next consider the circle of radius R centered at V_Y, and check if this spans all the points in BCB'. If it does, the edge AY is used as the best edge for the next iteration. If neither of these hold true, clearly the best edge to be used is AC - which is the edge which the Chrystal-Pierce algorithm would have chosen.

5. It must be observed that it is a trivial matter to check if a point D is in the semicircle BCB'. Indeed, if $\hat{s}$ is the vector normal to BB' (in the direction of C), then, the dot product

$$\hat{s} \cdot \vec{\Theta D} \begin{cases} > 0 & \text{if D is in the semicircle BCB'} \\ < 0 & \text{if D is in the semicircle BAB'} \\ = 0 & \text{if D is on BB'} \end{cases}$$

Using the above results we present a procedure GetNextTwo, whose inputs are S and the three vertices A, B and C. The procedure outputs the vertices V and W to be used in the next iteration of the algorithm.

PROCEDURE GetNextTwo (S, B, A, C, V, W)

<u>Input</u>: The set of points S, and the points A, B and C. B is the obtuse angle of the triangle ABC.

<u>Output</u>: The vertices V and W. The edge VW is the starting edge for the next iteration.

<u>Assumption</u>: Θ and R are the centre and radius of the circumcircle of ABC. These are assumed global.

<u>Method</u>

```
Compute the image of perturbation $V_D$ for all D in
semicircle BCB'

X = vertex which minimizes $|\Theta V_D|$.  Its image of
perturbation is $V_X$

IF $|V_X-E|<R$ for all E in semicircle BAB' THEN
   V = C
   W = X
ELSE
   Compute the image of perturbation $V_E$ for all E in
   semicircle BAB'

   Y = vertex which minimizes $|\Theta V_E|$.  Its image of
   perturbation is $V_Y$

   IF $|V_Y-D|<R$ for all D in semicircle BCB' THEN
      V = A
      W = Y
   ELSE {The Case when AC is the best edge}
      V = A
      W = C
   ENDIF
ENDIF
RETURN
END
```

END PROCEDURE GetNextTwo

Using the procedure GetNextTwo we now present our algorithm, ALGORITHM MSC, which processes the set of points S, and yields as its output the MSC of the set. After each iteration the algorithm calls the above procedure, GetNextTwo, which yields the edge to be used in the next iteration.

ALGORITHM MSC

Input: The set of points, S.

Output: The MSC of S. The user has the capability of deciding on the accuracy of the solution, if he does not want the exact solution.

Assumption: As in Algorithm Chrystal-Pierce, the user is accessible to a procedure GetFirstTwo which yields him the best two vertices according to the Chakraborti-Chaudhuri criterion. Also accessible to the algorithm is the procedure GetNextTwo presented above.

Method:

```
Call GetFirstTwo (S,A,B)
Unmarked = S
Done = Satisfied = FALSE
REPEAT UNTIL ((Done) OR (Satisfied))
   C = point which minimizes $\angle$ ADB for all D$\in$Unmarked-{A,B}
   Unmarked = Unmarked - {D | $\angle$ ADB > $\pi/2$}
   If $\angle$ ACB is obtuse THEN
      Done = TRUE
      MSC = Circle with AB as diameter
   ELSEIF triangle ABC is acute THEN
      Done = TRUE
      MSC = circumcircle of ABC
   ELSE
      Done = FALSE
      MSC = circumcircle of ABC
      recompute Satisfied
      IF $\angle$ ABC is obtuse THEN
         Unmarked = Unmarked - {B}
         Call GetNextTwo(S,B,A,C,A,B)
      ELSE
         Unmarked = Unmarked - {A}
         Call GetNextTwo(S,A,B,C,A,B)
      ENDIF
   ENDIF
ENDREPEAT
Print Out $\Theta$ and R of the MSC
END
```

END ALGORITHM MSC

To show the advantage of our algorithm over the traditional Chrystal-Pierce Algorithm we have shown in Figure III the value of θ in the <u>second</u> iteration using both the algorithms. θ_A is the position of the center obtained using the Chrystal-Pierce algorithm and θ_B is the center obtained using Algorithm MSC. The advantage is obvious.

In terms of complexity, Algorithm MSC cannot do worse than Algorithm Chrystal-Pierce. In every iteration we require O(N) computations, and furthermore, we also eliminate the same number of vertices. However, unlike the latter algorithm, we do not merely compute the best edge in the <u>initialization</u> process. At every stage in the algorithm the best two vertices are used as a starting point to determine the best approximation of the MSC.

CONCLUSIONS

The problem of computing the Minimum Spanning Circle of a set of points was proposed in 1857. Subsequently, three families of algorithms have emerged which solve the problem. In this paper we present an algorithm which converges in <u>finite</u> time and which uses the motivating ideas of all the above mentioned families. The technique is possibly the fastest known solution to the problem.

ACKNOWLEDGEMENTS

I would like to sincerely thank my colleagues at the School of Computer Science, Carleton University, for the numerous discussions we have had. In particular I would like to thank Mike Atkinson, Irwin Reichstein and Jorg Sack very specially.

REFERENCES

1. Blumenthal, L.M. and G.E. Wahlin, 1941. On the Spherical Surface of Smallest Radius Enclosing a Bounded Subset of n-Dimensional Euclidiean Space. Am. Math. Soc. Bull. 47, 771-777.

2. Castells, C., and R.C. Melville, 1983. An Unusual Algorithm for Finding the Minimum Spanning Circle of a Convex Polygon. Technical Report 17, Department of Electrical Engineering and Computer Science, Johns Hopkins University, Maryland.

3. Chakraborty, R.K., and P.K. Chaudhuri, 1981. Note on Geometrical Solutions for Some Minimax Location Problems. Trans. Sci. 15, 164-166.

4. Chrystal, G., 1885. On the Problem to Construct the Minimum Circle Enclosing n Given Points in the Plane. Proc. Edinburgh Math. Soc. 3, 30-33.

5. Elzinga, J., and D.W. Hearn, 1972a. The Minimum Covering Sphere Problem. Mgmt. Sci. 19, 96-104.

6. Elzinga, J., and D.W. Hearn, 1972b. Geometrical Solutions for Some Minimax Location Problems. Trans. Sci. 6, 379-394.

7. Hearn, D.W., and J. Vijay, 1982. Efficient Algorithms for the (Weighted) Minimum Circle Problem. Operations Research, 30, 777-795.

8. Jacobsen, S.K., 1981. An Algorithm for the Minimax Weber Problem. Eur. J. Opnl. Res. 6, 144-148.

9. Sylvester, J.J., 1857. A Question in the Geometry of Situation. Quart. J. Pure Appl. Math. 1, p.79.

10. Sylvester, J.J., 1860. On Poncelet's Approximate Linear Valuation, of Surd Forms. Philosophical Magazine 20 (Fourth Series), 203-222.

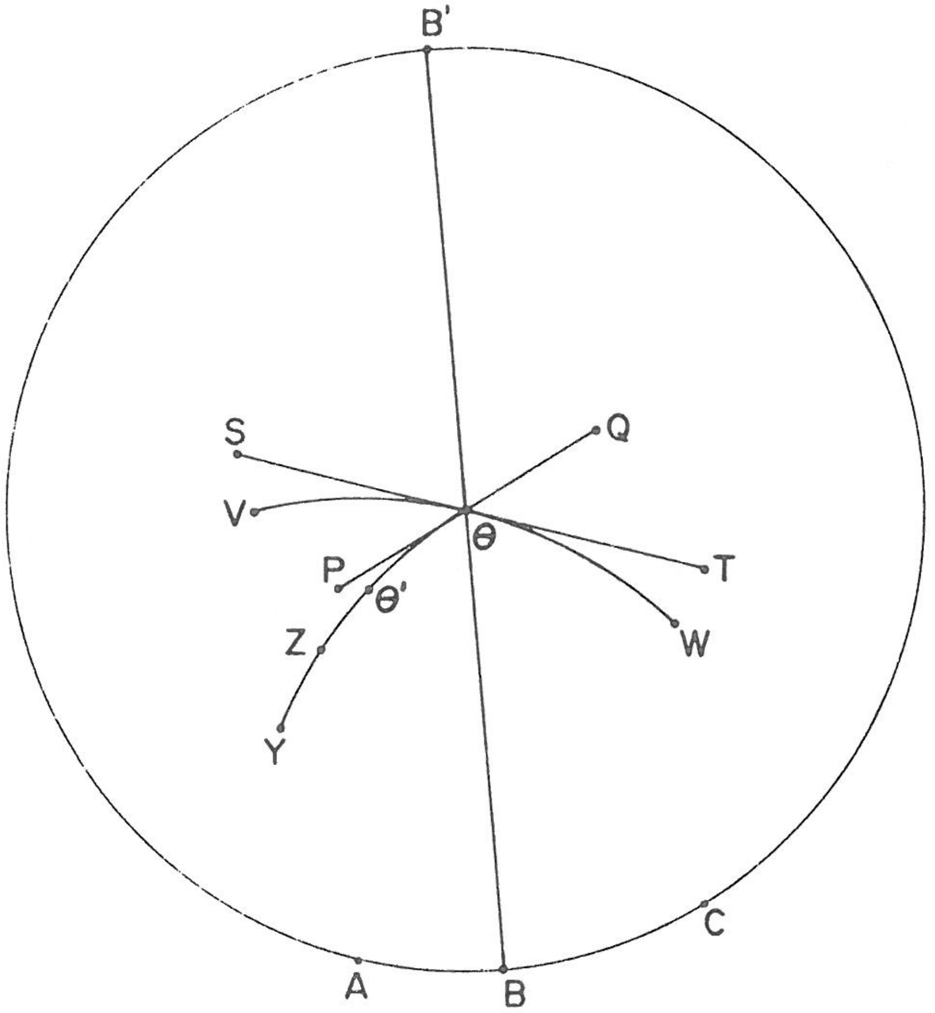

Figure 1. Θ and R are the center and radius of the circumcircle of ABC. YZΘ is the arc of radius R with center C. VΘW is the arc of radius R with center A. The tangents to these arcs are PΘQ and SΘT respectively.

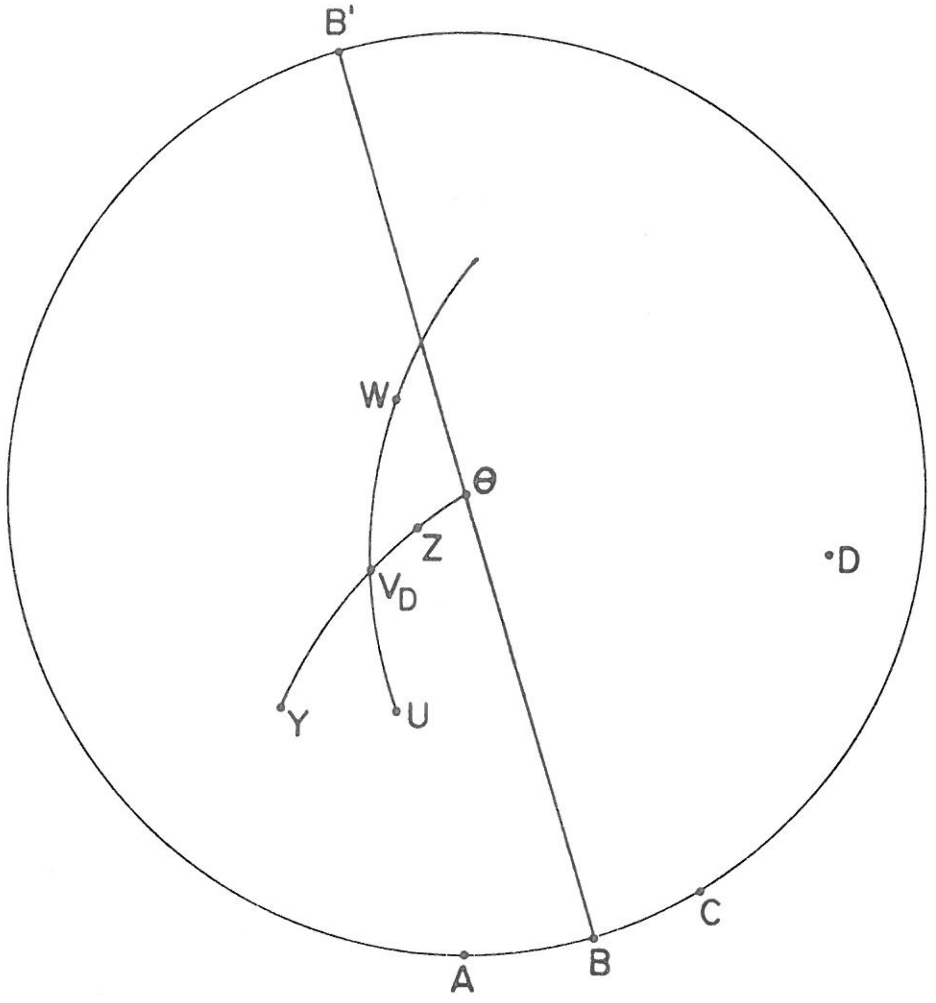

Figure II. Θ and R are the center and radius of the circumcircle of ABC. $YV_DZ\Theta$ is the arc of radius R from C. UV_DW is the arc of radius R from D. The point D in the semicircle BCB' which minimizes $|V_D\Theta|$ is the vertex of minimum perturbation with respect to C.

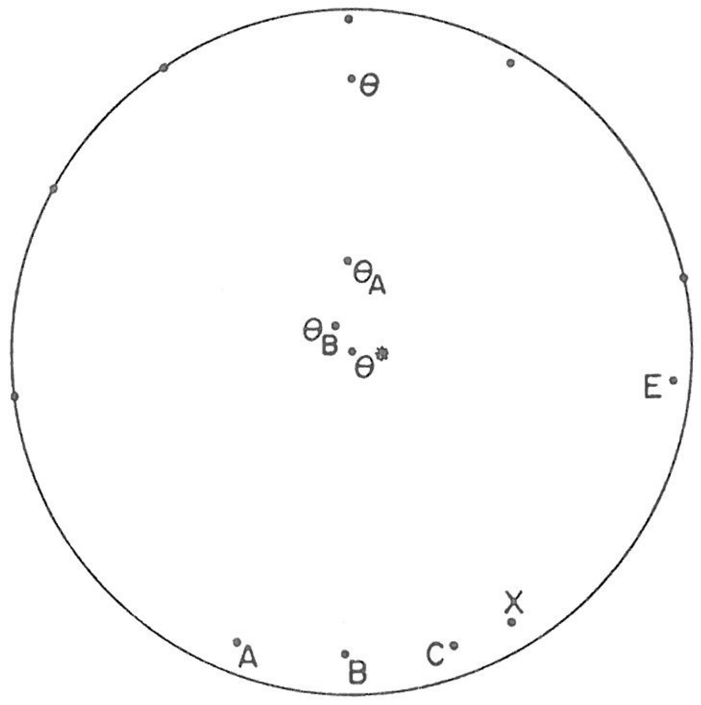

Figure III. θ is the center of the circumcircle of ABC. The center of the circle after the next iteration of the Chrystal-Pierce Algorithm is θ_A. The center of the circle after the next iteration of Algorithm MSC is θ_B. In this case X is the vertex of minimum perturbation and the approximation of the MSC passes through C, X and E. Note that θ^* is the center of the MSC.

CARLETON UNIVERSITY

School of Computer Science

BIBLIOGRAPHY OF SCS TECHNICAL REPORTS

SCS-TR-1	THE DESIGN OF CP-6 PASCAL Jim des Rivieres and Wilf R. LaLonde, June 1982.
SCS-TR-2	SINGLE PRODUCTION ELIMINATION IN LR(1) PARSERS: A SYNTHESIS Wilf R. LaLonde, June 1982.
SCS-TR-3	A FLEXIBLE COMPILER STRUCTURE THAT ALLOWS DYNAMIC PHASE ORDERING Wilf R. LaLonde and Jim des Rivieres, June 1982.
SCS-TR-4	A PRACTICAL LONGEST COMMON SUBSEQUENCE ALGORITHM FOR TEXT COLLATION Jim des Rivieres, June 1982.
SCS-TR-5	A SCHOOL BUS ROUTING AND SCHEDULING PROBLEM Wolfgang Lindenberg, Frantisek Fiala, July 1982.
SCS-TR-6	ROUTING WITHOUT ROUTING TABLES Nicola Santoro, Ramez Khatib, July 1982.
SCS-TR-7	CONCURRENCY CONTROL IN LARGE COMPUTER NETWORKS Nicola Santoro, Hasan Ural, July 1982.
SCS-TR-8 Out of Print	ORDER STATISTICS ON DISTRIBUTED SETS Nicola Santoro, Jeffrey B. Sidney, July 1982.
SCS-TR-9	OLIGARCHICAL CONTROL OF DISTRIBUTED PROCESSING SYSTEMS Moshe Krieger, Nicola Santoro, August 1982.
SCS-TR-10	COMMUNICATION BOUNDS FOR SELECTION IN DISTRIBUTED SETS Nicola Santoro, Jeffrey B. Sidney, September 1982.
SCS-TR-11	SIMPLE TECHNIQUE FOR CONVERTING FROM A PASCAL SHOP TO A C SHOP Wilf R. LaLonde, John R. Pugh, November 1982.
SCS-TR-12	EFFICIENT ABSTRACT IMPLEMENTATIONS FOR RELATIONAL DATA STRUCTURES Nicola Santoro, December 1982.
SCS-TR-13	ON THE MESSAGE COMPLEXITY OF DISTRIBUTED PROBLEMS Nicola Santoro, December 1982.

SCS-TR-14	A COMMON BASIS FOR SIMILARITY MEASURES INVOLVING TWO STRINGS R. L. Kashyap and B. J. Oommen, January 1983.
SCS-TR-15	SIMILARITY MEASURES FOR SETS OF STRINGS R. L. Kashyap and B. J. Oommen, January 1983.
SCS-TR-16	THE NOISY SUBSTRING MATCHING PROBLEM R. L. Kashyap and B. J. Oommen, January 1983.
SCS-TR-17	DISTRIBUTED ELECTION IN A CIRCLE WITHOUT A GLOBAL SENSE OF ORIENTATION E. Korach, D. Rotem, N. Santoro, January 1983.
SCS-TR-18	A GEOMETRICAL APPROACH TO POLYGONAL DISSIMILARITY AND THE CLASSIFICATION OF CLOSED BOUNDARIES R. L. Kashyap and B. J. Oommen, January 1983.
SCS-TR-19	SCALE PRESERVING SMOOTHING OF POLYGONS R. L. Kashyap and B. J. Oommen, January 1983.
SCS-TR-20	NOT-QUITE-LINEAR RANDOM ACCESS MEMORIES Jim des Rivieres, Wilf LaLonde and Mike Dixon, August 1982, Revised March 1, 1983.
SCS-TR-21	SHOUT ECHO SELECTION IN DISTRIBUTED FILES D. Rotem, N. Santoro, J. B. Sidney, March 1983.
SCS-TR-22	DISTRIBUTED RANKING E. Korach, D. Rotem, N. Santoro, March 1983.
SCS-TR-23	A REDUCTION TECHNIQUE FOR SELECTION IN DISTRIBUTED FILES : I N. Santoro, J. B. Sidney, April 1983.
SCS-TR-24	LEARNING AUTOMATA POSSESSING ERGODICITY OF THE MEAN : THE TWO ACTION CASE M. A. L. Thathachar and B. J. Oommen, May 1983.
SCS-TR-25	ACTORS - THE STAGE IS SET John R. Pugh, June 1983.
SCS-TR-26	ON THE ESSENTIAL EQUIVALENCE OF TWO FAMILIES OF LEARNING AUTOMATA M. A. L. Thathachar and B. J. Oommen, May 1983.
SCS-TR-27	GENERALIZED KRYLOV AUTOMATA AND THEIR APPLICABILITY TO LEARNING IN NONSTATIONARY ENVIROMENTS B. J. Oommen, June 1983.
SCS-TR-28	ACTOR SYSTEMS: SELECTED FEATURES Wilf R. LaLonde, July 1983.

SCS-TR-29	ANOTHER ADDENDUM TO KRONECKER'S THEORY OF PENCILS M. D. Atkinson, August 1983.
SCS-TR-30	SOME TECHNIQUES FOR GROUP CHARACTER REDUCTION M. D. Atkinson and R. A. Hassan, August 1983.
SCS-TR-31	AN OPTIMAL ALGORITHM FOR GEOMETRICAL CONGRUENCE M. D. Atkinson, August 1983.
SCS-TR-32	MULTI-ACTION LEARNING AUTOMATA POSSESSING ERGODICITY OF THE MEAN B. J. Oommen and M. A. L. Thathachar, August 1983.
SCS-TR-33	FIBONACCI GRAPHS, CYCLIC PERMUTATIONS AND EXTREMAL POINTS N. Santoro and J. Urrutia, December 1983
SCS-TR-34	DISTRIBUTED SORTING D. Rotem, N. Santoro, and J. B. Sidney, December 1983
SCS-TR-35	A REDUCTION TECHNIQUE FOR SELECTION IN DISTRIBUTED FILES: II N. Santoro, M. Scheutzow, and J. B. Sidney, December 1983
SCS-TR-36	THE ASYMPTOTIC OPTIMALITY OF DISCRETIZED LINEAR REWARD-INACTION LEARNING AUTOMATA B. J. Oommen and Eldon Hansen, January 1984
SCS-TR-37	GEOMETRIC CONTAINMENT IS NOT REDUCIBLE TO PARETO DOMINANCE N. Santoro, J. B. Sidney, S. J. Sidney, and J. Urrutia, January 1984
SCS-TR-38	AN IMPROVED AGLORITHM FOR BOOLEAN MATRIX MULTIPLICATION N. Santoro and J. Urrutia, January 1984
SCS-TR-39	CONTAINMENT OF ELEMENTARY GEOMETRIC OBJECTS J. Sack, N. Santoro and J. Urrutia, February 1984
SCS-TR-40	SADE: A PROGRAMMING ENVIRONMENT FOR DESIGNING AND TESTING SYSTOLIC ALGORITHMS J. P. Corriveau and N. Santoro, February 1984
SCS-TR-41	INTERSECTION GRAPHS, {B }-ORIENTABLE GRAPHS AND PROPER CIRCULAR ARC GRAPHS Jorge Urrutia, February 1984
SCS-TR-42	MINIMUM DECOMPOSITIONS OF POLYGONAL OBJECTS J. Mark Keil and Jorg-R. Sack, March 1984
SCS-TR-43	AN ALGORITHM FOR MERGING HEAPS Jorg-R. Sack and Thomas Strothotte, March 1984

SCS-TR-44	A DIGITAL HASHING SCHEME FOR DYNAMIC MULTIATTRIBUTE FILES E. J. Otoo, March 1984
SCS-TR-45	SYMMETRIC INDEX MAINTENANCE USING MULTIDIMENSIONAL LINEAR HASHING E. J. Otoo, March 1984
SCS-TR-46	A MAPPING FUNCTION FOR THE DIRECTORY OF A MULTIDIMENSIONAL EXTENDIBLE HASHING E. J. Otoo, March 1984
SCS-TR-47	TRANSLATING POLYGONS IN THE PLANE Jorg-R. Sack, March 1984
SCS-TR-48	CONSTRAINED STRING EDITING J. Oommen, May 1984
SCS-TR-49	O(N) ELECTION ALGORITHMS IN COMPLETE NETWORKS WITH GLOBAL SENSE OF ORIENTATION Jorg Sack, Nicola Santoro, Jorge Urrutia, May 1984
SCS-TR-50	THE DESIGN OF A PROGRAM EDITOR BASED ON CONSTRAINTS Christopher A. Carter and Wilf R. LaLonde, May 1984
SCS-TR-51	DISCRETIZED LINEAR INACTION-PENALTY LEARNING AUTOMATA B. J. Oommen and Eldon Hansen, May 1984
SCS-TR-52	SENSE OF DIRECTION, TOPOLOGICAL AWARENESS AND COMMUNICATION COMPLEXITY Nicola Santoro, May 1984
SCS-TR-53	OPTIMAL LIST ORGANIZING STRATEGY WHICH USES STOCHASTIC MOVE-TO-FRONT OPERATIONS B. J. Oommen, June 1984
SCS-TR-54	RECTINLINEAR COMPUTATIONAL GEOMETRY J. Sack, June 1984
SCS-TR-55	AN EFFICIENT, IMPLICIT DOUBLE-ENDED PRIORITY QUEUE M. D. Atkinson, Jorg Sack, Nicola Santoro, T. Strothotte, July 1984
SCS-TR-56	DYNAMIC MULTIPAGING: A MULTIDIMENSIONAL STRUCTURE FOR FAST ASSOCIATIVE SEARCHING E. Otoo, T. H. Merrett
SCS-TR-57	SPECIALIZATION, GENERALIZATION AND INHERITANCE Wilf R. LaLonde, John R. Pugh, August 1984

SCS-TR-58	COMPUTER ACCESS METHODS FOR EXTENDIBLE ARRAYS OF VARYING DIMENSIONS E. Otoo, August 1984.
SCS-TR-59	AREA-EFFICIENT EMBEDDINGS OF TREES J. P. Corriveau, Nicola Santoro, August 1984.
SCS-TR-60	UNIQUELY COLOURABLE m-DICHROMATIC ORIENTED GRAPHS V. Neumann-Lara, N. Santoro, J. Urrutia, August 1984.
SCS-TR-61	ANALYSIS OF A DISTRIBUTED ALGORITHMS FOR EXTREMA FINDING IN A RING D. Rotem, E. Korach and N. Santoro, August 1984.
SCS-TR-62	ON ZIGZAG PERMUTATIONS AND COMPARISONS OF ADJACENT ELEMENTS M. D. Atkinson, October 1984
SCS-TR-63	SETS OF INTEGERS WITH DISTINCT DIFFERENCES M. D. Atkinson, A. Hassenklover, October 1984.
SCS-TR-64	TEACHING FIFTH GENERATION COMPUTING: THE IMPORTANCE OF SMALL TALK Wilf R. LaLonde, Dave A. Thomas, John R. Pugh, October 1984.
SCS-TR-65	AN EXTREMELY FAST MINIMUM SPANNING CIRCLE ALGORITHM B. J. Oommen, October, 1984.
SCS-TR-66	ON THE FUTILITY OF ARBITRARILY INCREASING MEMORY CAPABILITIES OF STOCHASTIC LEARNING AUTOMATA B. J. Oommen, October, 1984.