Minimum Enclosing Circle

Mohammed Alshammasi
Mohammed.alshammasi@kaust.edu.sa

Shahad Qathan
shahad.qathan@kaust.edu.sa

Faisal Mushayt
faisal.mushayt@kaust.edu.sa

Shariq F. Bhat
shariq.bhat@kaust.edu.sa

Aseel Albeshri
aseel.beshri@kaust.edu.sa

1 Introduction

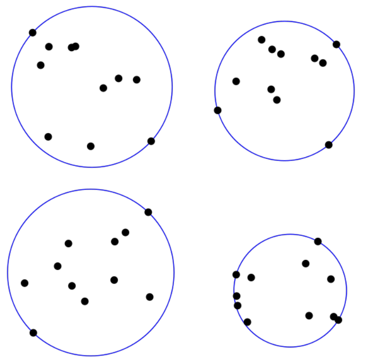


Figure 1: Minimum Enclosing Circle problem[1]

Computational Geometry lies at the intersection of Computer Science and Geometry, and is considered as one of the most aesthetic combination of Mathematics and Computer Science. Geometry gives us insights regarding the properties of shapes, size, and position of abstract and real-world objects. These insights lead us to finding solutions to some important practical problems. One interesting problem that lies in the domain of Computational Geometry is the Minimal Enclosing Circle (MEC), introduced by Sylvester in 1857. It states that given a set of points, find the circle that has the smallest radius that covers all the points in the set (A point could lie on the boundary of the circle) [1].

There are a variety of applications to this problem in many different domains, such as networking and health. In the networking domain, the paper [2] proposed an application that uses MEC, which considers base stations that provide wireless connectivity. In the event of failure of any terrestrial

[1]Adapted from Claudio Rocchini `https://commons.wikimedia.org/wiki/File:Smallest_circle_problem.svg`

base station, an Unmanned Aerial Vehicle Base Station (UAV-BS) could quickly support the demand for the wireless service. However, finding locations for the UAV-BSs that maximizes the number of covered users and minimizes transmission power is a difficult problem. The transmission power needed increases with the coverage radius. In that paper, they used the MEC problem to optimize for the transmission power, where the center of a circle that encloses users with the smallest radius is optimal to place a UAV-BS. Another application resides in the facility construction domain, particularly Emergency Health Service, which is finding the right place to build an emergency station that holds ambulance vehicles in a city. As one can notice, the customer houses can be interpreted to be the set of points, and finding the center of the MEC that covers all the houses would be a perfect place to build the facility. The MEC problem has been a problem of interest for more than a century, and many algorithms were proposed, each improving the time complexity of its predecessor. However, it took the researchers a whole century to find a simple, yet efficient algorithm to this problem. The algorithm in discussion is a randomized incremental algorithm developed by Welzl[1] which takes expected $O(n)$ time, where n is the number of points. Although there existed a linear time algorithm prior to this algorithm [3], it is more complicated and the involved constants in $O(n)$ are large [4]. Our goal is to explore two algorithms that solve the MEC problem by analyzing and comparing their time complexity, both theoretically and experimentally.

2 Problem Formulation

Let the given set of n points be denoted by $P := \{p_1, p_2, \ldots, p_n\}$, where each $p_i \in \mathbb{R}^2$. Let $C(P)$ denote the minimal enclosing circle of the set P. As a circle can be defined by two parameters: the radius r and the centre p_o, the minimal enclosing circle $C(P)$ of a set of points P is

$$C(P) := argmin_{r,p_o}\, r^2 \quad s.t. \quad \|p_i - p_o\|^2 \leq r^2 \quad \forall i \in \{1, 2, \ldots, n\} \tag{1}$$

3 Algorithms

3.1 Brute Force

This problem may seem difficult to brute force at first due to the continuous nature of the search space, but a few observations from computational geometry make such a method feasible. The first thing to note is any MEC will always intersect at least two points on its boundary. Given a set of points in a plane enclosed by any circle, the circle can be shrunk at least until it hits one of the points, α . Then, α is a fixed point, and the circle can be further shrunk towards that point until it intersects at least one more point, β. The second thing to note is, if the line $\alpha\beta$ forms the diameter of this circle, we have found our MEC. Otherwise, the circle can be further shifted towards the midpoint of this line until it hits a third point, γ. Now γ is a fixed point and and the circle is again shrunk until it intersects α and β again. This forms the MEC.

Knowing this, a circle is uniquely defined by two or three points. A brute force approach to the MEC problem is done by examining all possible pairs and triples of points to get the circle defined by those points, then checking if the corresponding circle encloses all the other points. We keep track of the smallest circle recorded and return the points that form it.

We would like to understand some theoretical bounds on such an approach. We can note that there is an order of $O(n^2)$ pairs we must try, as well as $O(n^3)$ triplets. Then we have to check whether each other point lies within the circle or not which takes $O(n)$ comparisons. The time complexity for this naive approach amounts to $O(n^3 + n^4)$, or $O(n^4)$. Note this bound is purely theoretical and does not take into account any implementation overhead.

3.2 Welzl's Randomized Algorithm

Next, we would like to explore a more widely used approach to solve this problem, Welzl's Randomized Algorithm. Welzl used the facts discussed in the last section to develop an algorithm that can quickly compute such a circle. Essentially, Welzl's algorithm grows C(P) incrementally. Given the set of points P, we recursively compute the MEC for the set $P - p_i$ for some randomly chosen p_i . The beauty of this lies in its randomness. There is a chance point p_i is already enclosed by the circle, and thus we only need to return the circle already found. If this is not the case then we know the point lies on the boundary of the MEC for this subproblem. So we can again recurse on P, this time indicating p_i is on the boundary. To do this, we define a Support Set to maintain Support Points, points that lie on the perimeter of the circle.

It is good to note a base case is reached if P is empty, meaning all the points have been looked at, or the support set has a size equal to or less than 3, then the MEC can be computed directly from the support points. This leads to the MEC for the entire set of points being constructed in a bottom-up manner. The circle is constantly shifted and expanded until it encloses every point.

Such a method seems promising as Welzl proved in [1] that the algorithm computes the smallest enclosing circle in expected O(n) time. He proposes the idea of removing points randomly to minimize the chance they are arranged in such a way that would lead to the worst case time complexity.

3.3 Move-to-first Heuristic (MTF)

Welzl [1] showed that it suffices to choose one random permutation of $1...n$ in the beginning of the computation and at each call remove the point at the last index for the boundary test described above. This further simplifies the algorithm and leads to a heuristic which significantly speeds up the algorithm in practice. We can pose the question: what would be a good permutation to begin with? Of course, if the first elements are those which determine the solution, then we have the optimal situation, and the algorithm will not make more than n + O(1) boundary tests [1]. Since we cannot just compute such an optimal permutation beforehand, we would at-least like to have points early in the sequence which determine a circle with few points outside. Although we are not given such a sequence, we can gradually update the sequence during the computation by moving points to the front of the sequence which we consider important. Intuitively, these are the points which are outside the current circle. The 'important' points would therefore get processed early in subsequent recursive calls [5]. Here we assume that the point set is stored in a global sequence, preferably in a linked list which enables us to move points to the front in constant time.

The effect of the Move-to-first heuristic is that the circle maintained by the algorithm gets large fast. Thus the algorithm processes much lesser number of points and converges to the solution much quicker than the vanilla version.

4 Implementation Overview

We have implemented both algorithms using Python, along with the MTF variant of the Welzl. In order to ensure the correctness of the algorithms we carried out comparison tests between the results of all implementations. Theoretically, all implementations should find the same MEC since it is uniquely defined. That being said, we had some preliminary expectations for the performance of the two main algorithms. We were confident the runtime of the brute force approach will be quite long when compared to that of Welzl's. So, we conducted an experiment where we tracked the rate of growth of each algorithm by measuring the time taken for various input sizes.

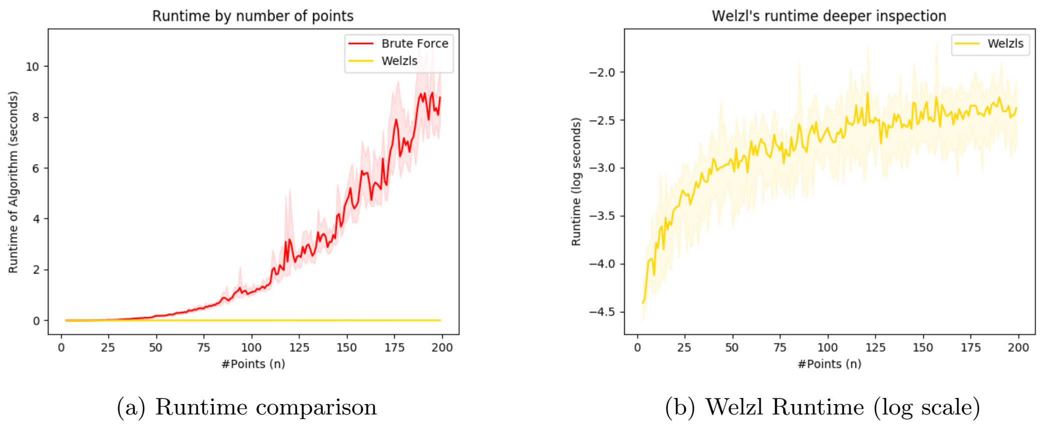
Furthermore, we expected the distribution of the points to play a significant role in the time it takes Welzl's to find the solution. Although Welzl proved the average case time complexity of his algorithm remains linear no matter what, in practice, we believed the location of the points on the plane will affect the number of times the algorithm must reconstruct the circle. Intuitively, it is more likely to randomly choose points that will form a circle enclosing most of the points if they are all clustered together than if multiple clusters exist. We wanted to test this idea. There was a chance randomness accounted for it, but if not, we also wanted to test whether MTF could solve this issue. Therefore, we also conducted an experiment to observe how different distributions of points can impact the speed of Welzl's along with its MTF variant.

5 Experimental Results

In this section we will briefly elaborate on the implementation details of each experiment as well as provide our main results.

5.1 Correctness

To ensure the correctness of the algorithms, we first defined a few test cases which include point sets whose MEC is easy to calculate by hand. We then ensured the correctness on a few random point sets by visual inspection, particularly for the Brute Force implementation. To further confirm the correctness, since the MEC for a given point set is unique, we executed all the three algorithms (Brute Force, Welzl-Vanilla, Welzl-MTF) on 1000 randomly generated point sets of varying cardinalities (six to thirty) and verified whether all the three algorithms resulted in identical center points and radii for a given point set (while taking floating-point precision into consideration). Furthermore, we have written a script to generate gif's showing the construction of MEC by each of the three algorithms.

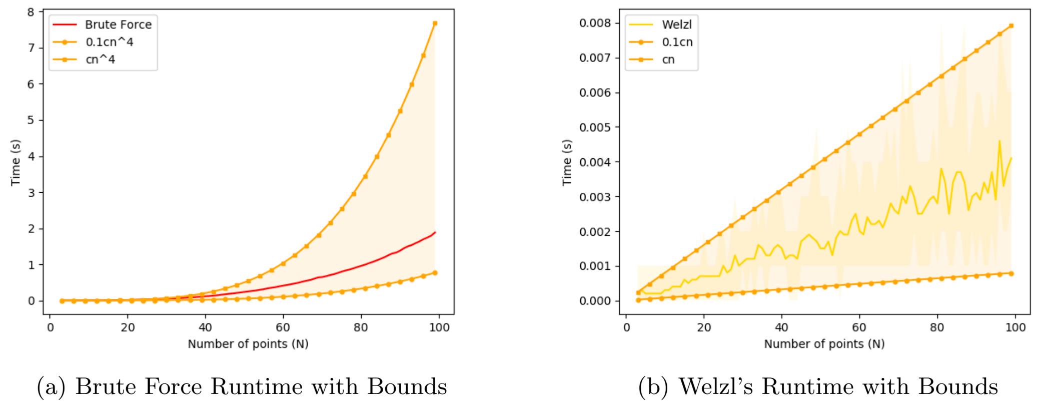


(a) Runtime comparison (b) Welzl Runtime (log scale)

Figure 2: Algorithms Quantitative Comparison

5.2 Runtime

After the confirmation of correctness of our algorithms, that is the uniqueness of the MEC, we conduct the runtime analysis for both of the described main algorithms. We record the runtimes of the algorithm for various cardinalities of the input point set. For each n (number of points) ranging from 3 up to 200, we perform 10 experiments where we generate points from a uniform distribution and solve the MEC problem using both brute force and Welzl, keeping track of the runtimes for each. We then plot average runtimes versus n, shown in Fig. 2.



(a) Brute Force Runtime with Bounds (b) Welzl's Runtime with Bounds

Figure 3: Runtime Comparison with Bounds

Fig. 2a displays the runtimes of both approaches. The shaded region around the plots shows maximum and minimum observed runtimes for each experiment with n points. However Welzl's curve is unreadable since its runtimes are fractions of seconds. Down at this level, runtimes cannot really be measured accurately due to system overhead (memory access, CPU clock inaccuracy, etc.) So, instead, in Fig. 2b we show Welzl's runtime on a log scale to observe its growth trend. Fig. 4 shows the growth rate of the Brute Force runtime relative to that of Welzls. This shows us the volatility of the brute force approach. At 200 points, Brute Force takes up to around 3000 times as long as Welzl. Observe the spikes in the green hues however. We can see that in the worse case scenario, even with only around 175 points, Brute Force might take up to around 7000 times as long as Welzl. The growth rate is massive.

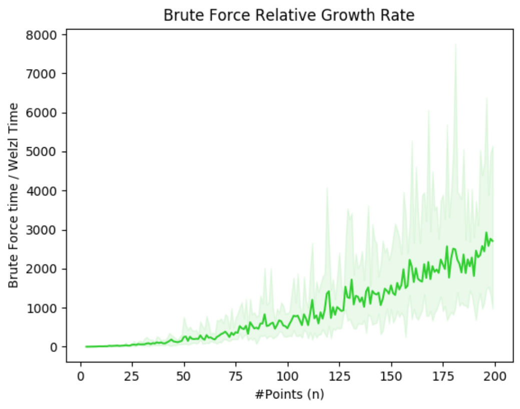


Figure 4: Relative Growth Rate of Brute Force Runtime

Furthermore, to test theoretical bounds on each algorithm. Using the same experimental scheme, we performed bound tests where we fit curves to bound the observed spread of results . Fig. 3a shows the observed Brute Force runtimes bounded from above and below by polynomials of the 4th order. Likewise, Fig. 3b shows the observed Welzl runtimes bounded from above and below by a linear function.

5.3 Distribution Effect on Welzl

Finally, we compare Welzl's randomized algorithm and its MTF variant. The goal of the experiment is to observe the speedup gained using the Move-To-First heuristic for various distributions of points. We tested the algorithms on Uniform, Gaussian, Beta, and Multi-modal (with varying cluster numbers k) distributions of points. For each distribution we executed experiments for n up to 500, running 5 trials for each configuration and recording the runtimes. The results of the tests on the different distributions can be seen in Fig. 5. We can see that both algorithms follow the same trend regardless of the distribution of the points. They both take longer as n increases but MTF decisively outperforms vanilla Welzl. Both algorithms are mostly unaffected by the distribution.

6 Discussion

For the most part, the algorithms performed as expected. Welzl's algorithm is able to solve the MEC problem on a much larger number of points than a Brute Force approach in a much smaller time frame. We've also shown experimentally how both studied approaches obey theoretical bounds. The most unexpected result perhaps is Welzl's algorithm's efficiency in providing a correct MEC

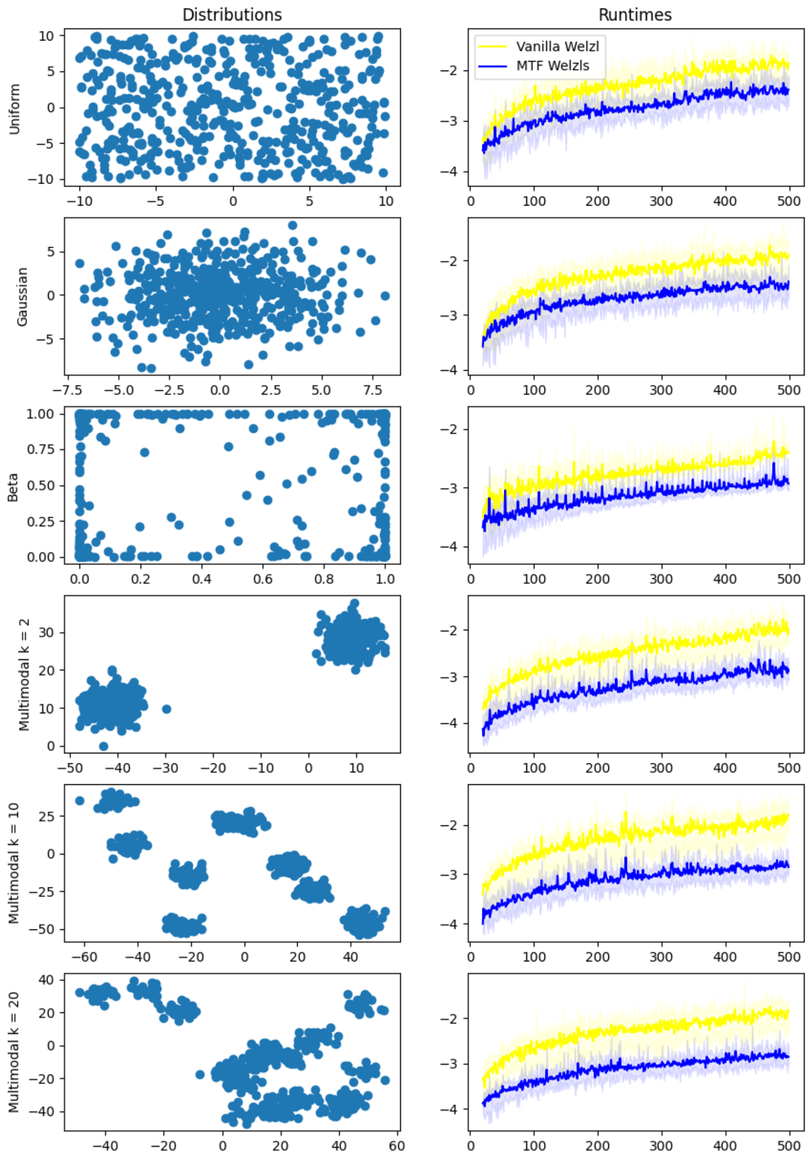


Figure 5: Distributions and Welzl's Results

regardless of the shape or distribution of the set P. The simple MTF heuristic speeds up Welzl significantly but is not needed to deal with varying distributions like we had initially thought. It is good to note one limiting factor in our implementation which is the maximum recursion depth in Python. Unless manually changed, Welzl cannot be executed on large sets of points (approximately 800).

For future work, it would be quite interesting to see whether Welzl can be used to develop an approximate algorithm that detects outliers, or perhaps classifies whether a point belongs to a given set or not. Furthermore, it might be useful to compare and analyse an implementation of Megiddo's algorithm described in [3] with Welzl's algorithm and its MTF variant, since both algorithms are linear. Although both algorithms would produce the same solution due to the uniqueness of the MEC, such experimental analysis of the two algorithms could lead to interesting computational time and processing resources results due to the different nature of the two algorithms.

7 Conclusion

In this paper, we have described two algorithms to solve the MEC problem. Both algorithms rely on a property of the problem, that is, there exists a unique MEC (solution) to the problem defined by points lying on the boundary of the MEC. Both algorithms successfully demonstrated their efficacy to solve the problem. However, there were clear observed limitations of Brute Force approaches. It can be seen from experimental results that such an approach does not scale very well with the size of the input. On the other hand, Welzl's algorithm demonstrated the power of randomized algorithms, a simple algorithm that runs efficiently in expectation. The MTF heuristic showed that simple ideas based on analysing features of the problem can lead to better results in practice.

References

[1] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)", in *New Results and New Trends in Computer Science. LNCS, vol. 555.* Springer, 1991, pp. 359–370.

[2] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage", *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 434–437, 2017.

[3] N. Megiddo, "Linear-time algorithms for linear programming in Rˆ3 and related problems", *SIAM Journal on Computing*, vol. 12, no. 4, pp. 759–776, 1983.

[4] S. Skyum, "A simple algorithm for computing the smallest enclosing circle", *Information Processing Letters*, vol. 37, no. 3, pp. 121–125, 1991.

[5] B. Gärtner, "Fast and robust smallest enclosing balls", in *European symposium on algorithms*, Springer, 1999, pp. 325–338.