Minimizing a Weighted Distance Function in a Triangle

Problem Statement, Analytical Solution, and Numerical Verification

Vamshi Jandhyala

13 December 2025

Contents

§1 Introduction

This paper investigates two related optimization problems on triangles: minimizing weighted squared distances and the generalized Fermat-Torricelli problem. Both problems have elegant analytical solutions with deep connections to classical triangle geometry.

We first solve a weighted squared-distance minimization problem and show that the solution is the incenter. We then explore the generalized Fermat-Torricelli problem, presenting key results from Uteshev's 2014 work [1], and prove a remarkable special case: when weights equal side lengths, the minimum equals $4 \cdot \mathrm{Area}$.

§2 Weighted Squared-Distance Minimization

§2.1 Problem Statement

Given a triangle with vertices A, B, and C, and side lengths:

- $a = |BC|$ (side opposite to vertex A)
- $b = |CA|$ (side opposite to vertex B)
- $c = |AB|$ (side opposite to vertex C)

Find the point P inside the triangle that minimizes the function:

$$f(P) = a \cdot |AP|^2 + b \cdot |BP|^2 + c \cdot |CP|^2$$

where $|AP|$ denotes the Euclidean distance between points A and P.

§2.2 Analytical Solution

§2.2.1 Step 1: Expand the Function

Let $P = (x, y)$, $A = (A_x, A_y)$, $B = (B_x, B_y)$, $C = (C_x, C_y)$. Then:

$$\begin{aligned} f(P) = a \cdot \left[(x - A_x)^2 + (y - A_y)^2\right] \\ + b \cdot \left[(x - B_x)^2 + (y - B_y)^2\right] \\ + c \cdot \left[(x - C_x)^2 + (y - C_y)^2\right] \end{aligned}$$

Expanding:

$$\begin{aligned} f(P) = a\left(x^2 - 2xA_x + A_x^2 + y^2 - 2yA_y + A_y^2\right) \\ + b\left(x^2 - 2xB_x + B_x^2 + y^2 - 2yB_y + B_y^2\right) \\ + c\left(x^2 - 2xC_x + C_x^2 + y^2 - 2yC_y + C_y^2\right) \end{aligned}$$

Collecting terms:

$$\begin{aligned} f(P) = (a + b + c)\left(x^2 + y^2\right) - 2x(aA_x + bB_x + cC_x) \\ - 2y\left(aA_y + bB_y + cC_y\right) + K \end{aligned}$$

where K is a constant (independent of x and y).

§2.2.2 Step 2: Find Critical Points

To minimize $f(P)$, we compute the partial derivatives and set them to zero:

$$\frac{\partial f}{\partial x} = 2(a+b+c)x - 2(aA_x + bB_x + cC_x) = 0$$

$$\frac{\partial f}{\partial y} = 2(a+b+c)y - 2\big(aA_y + bB_y + cC_y\big) = 0$$

Solving for x and y:

$$x = \frac{aA_x + bB_x + cC_x}{a+b+c}$$

$$y = \frac{aA_y + bB_y + cC_y}{a+b+c}$$

Therefore, the critical point is:

$$\boxed{P_{min} = \frac{a \cdot A + b \cdot B + c \cdot C}{a+b+c}}$$

§2.2.3 Step 3: Identify the Critical Point

The formula $P_{min} = \frac{a \cdot A + b \cdot B + c \cdot C}{a+b+c}$ is the definition of the **incenter** of the triangle!

The incenter is the center of the inscribed circle (incircle) and is the weighted average of the vertices, where the weights are the opposite side lengths.

§2.2.4 Step 4: Verify it is a Minimum

The Hessian matrix of f is:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2(a+b+c) & 0 \\ 0 & 2(a+b+c) \end{pmatrix}$$

Both eigenvalues equal $2(a+b+c) > 0$, so H is positive definite. This confirms that P_{min} is indeed a **minimum** (not a maximum or saddle point).

Since this is the only critical point and $f(P) \to \infty$ as $|P| \to \infty$, this is the **global minimum**.

§2.2.5 The Minimum Value

At the incenter P_{min}, the minimum value is:

$$f_{min} = \frac{ab \cdot c^2 + bc \cdot a^2 + ca \cdot b^2}{a+b+c}$$

This can be derived by substituting P_{min} back into the original function.

§2.3 Special Case: Equilateral Triangle

For an equilateral triangle with side length s (i.e., $a = b = c = s$):

The incenter coincides with the centroid, and:

$$P_{min} = \frac{A + B + C}{3}$$

$$f_{min} = s^3$$

§2.4 Summary of Weighted Squared-Distance Problem

Key Results:

1. The function $f(P) = a \cdot |AP|^2 + b \cdot |BP|^2 + c \cdot |CP|^2$ is minimized at the **incenter** of the triangle:
$$P_{min} = \frac{a \cdot A + b \cdot B + c \cdot C}{a + b + c}$$
2. The minimum value is:
$$f_{min} = \frac{ab \cdot c^2 + bc \cdot a^2 + ca \cdot b^2}{a + b + c}$$
3. For equilateral triangles only:
$$f_{min} = s^3$$
where s is the side length.
4. The incenter is always inside the triangle (for any valid triangle), confirming that the unconstrained minimum lies within the domain.

§3 Generalized Fermat-Torricelli Problem

The classical **Fermat-Torricelli problem** seeks a point P in the plane that minimizes the sum of distances to three given points. The **generalized Fermat-Torricelli problem** extends this by introducing positive weights.

§3.1 Problem Formulation

Given a triangle with vertices A_1, A_2, and A_3, and positive weights m_1, m_2, and m_3, find the point $P = (x, y)$ that minimizes:

$$F(x, y) = m_1 \, |PA_1| + m_2 \, |PA_2| + m_3 \, |PA_3|$$

where $|PA_i|$ denotes the Euclidean distance from P to A_i.

This is a fundamental problem in location theory with applications in facility location, network design, and computational geometry.

§3.2 Uteshev's Analytical Solution

In 2014, Alexei Yu. Uteshev [1] provided an explicit analytical solution to the generalized Fermat-Torricelli problem. His key contribution was expressing the coordinates of the optimal point directly in terms of the vertex coordinates and weights.

§3.2.1 Theorem 1.1 (Existence and Uniqueness)

Let P_1, P_2, P_3 be three non-collinear points in $\mathbb{R}^2$ with positive weights m_1, m_2, $m_3 > 0$. Denote by α_1, α_2, α_3 the corner angles of the triangle $P_1P_2P_3$.

If the conditions

$$\begin{cases} m_1^2 < m_2^2 + m_3^2 + 2m_2m_3 \cos\alpha_1, \\ m_2^2 < m_1^2 + m_3^2 + 2m_1m_3 \cos\alpha_2, \\ m_3^2 < m_1^2 + m_2^2 + 2m_1m_2 \cos\alpha_3 \end{cases}$$

are fulfilled, then there exists a unique solution $P^* = (x^*, y^*) \in \mathbb{R}^2$ for the generalized Fermat-Torricelli problem lying **inside** the triangle $P_1P_2P_3$. This point is a stationary point for the function $F(x, y)$, i.e., a real solution of the system:

$$\sum_{j=1}^{3} \frac{m_j(x - x_j)}{\sqrt{(x - x_j)^2 + (y - y_j)^2}} = 0, \quad \sum_{j=1}^{3} \frac{m_j(y - y_j)}{\sqrt{(x - x_j)^2 + (y - y_j)^2}} = 0$$

If any of the conditions is violated, then $F(x, y)$ attains its minimum value at the corresponding vertex of the triangle.

§3.2.2 Theorem 2.1 (Analytical Formula for the Coordinates)

Under the conditions of Theorem 1.1, the coordinates of the stationary point (x^*, y^*) of the function $F(x, y)$ are as follows:

$$x^* = \frac{K_1K_2K_3}{4S\sigma d}\left(\frac{x_1}{K_1} + \frac{x_2}{K_2} + \frac{x_3}{K_3}\right), \quad y^* = \frac{K_1K_2K_3}{4S\sigma d}\left(\frac{y_1}{K_1} + \frac{y_2}{K_2} + \frac{y_3}{K_3}\right)$$

with

$$F(x^*, y^*) = \min_{(x,y)} F(x, y) = \sqrt{d}$$

Here

$$d = \frac{1}{2\sigma}\left(m_1^2 K_1 + m_2^2 K_2 + m_3^2 K_3\right)$$

and

$$r_{j\ell} = \sqrt{\left(x_j - x_\ell\right)^2 + \left(y_j - y_\ell\right)^2} = |P_j P_\ell| \quad \text{for } \{j, \ell\} \in \{1, 2, 3\}$$

$$S = |x_1 y_2 + x_2 y_3 + x_3 y_1 - x_1 y_3 - x_3 y_2 - x_2 y_1|$$

$$\sigma = \frac{1}{2}\sqrt{-m_1^4 - m_2^4 - m_3^4 + 2m_1^2 m_2^2 + 2m_1^2 m_3^2 + 2m_2^2 m_3^2}$$

and

$$\begin{cases} K_1 = \left(r_{12}^2 + r_{13}^2 - r_{23}^2\right)\sigma + \left(m_2^2 + m_3^2 - m_1^2\right)S, \\ K_2 = \left(r_{23}^2 + r_{12}^2 - r_{13}^2\right)\sigma + \left(m_1^2 + m_3^2 - m_2^2\right)S, \\ K_3 = \left(r_{13}^2 + r_{23}^2 - r_{12}^2\right)\sigma + \left(m_1^2 + m_2^2 - m_3^2\right)S \end{cases}$$

§3.2.3 Geometric Interpretation

The constant S equals the doubled area of triangle $P_1P_2P_3$:

$$S = \left|\det\begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix}\right|$$

The constant σ can be treated as Heron's formula for the doubled area of a triangle formed by the triple of weights m_1, m_2, m_3. Factorization of the radicand gives:

$$\sigma = 2\sqrt{\frac{m_1 + m_2 + m_3}{2}\left(\frac{m_1 + m_2 + m_3}{2} - m_1\right)\left(\frac{m_1 + m_2 + m_3}{2} - m_2\right)\left(\frac{m_1 + m_2 + m_3}{2} - m_3\right)}$$

Under the restrictions from Theorem 1.1, such a triangle with sides m_1, m_2, m_3 exists. The coefficients K_j can be interpreted using the law of cosines: if we denote the angles of this weight-triangle as β_1, β_2, β_3, then:

$$K_1 = 2\sigma S(\cot\alpha_1 + \cot\beta_1)$$

and similarly for K_2 and K_3. This shows that under the restrictions of Theorem 1.1, all values K_1, K_2, K_3 are positive.

§3.3 Special Case: Minimum Value with Side-Length Weights

Theorem 3.1. *For an acute-angled triangle $P_1P_2P_3$ with side lengths a, b, c and area A, when the weights in the Fermat-Torricelli problem are set equal to the side lengths ($m_1 = a$, $m_2 = b$, $m_3 = c$), we have:*

$$d = 16A^2$$

and therefore the minimum value equals:

$$F_{min} = \sqrt{d} = 4A$$

§3.3.1 Derivation

When $m_1 = a$, $m_2 = b$, $m_3 = c$ are the side lengths of the triangle, we substitute into the formula for σ from Theorem 2.1:

$$\sigma = \frac{1}{2}\sqrt{-a^4 - b^4 - c^4 + 2a^2b^2 + 2a^2c^2 + 2b^2c^2}$$

The radicand can be factored as:

$$-a^4 - b^4 - c^4 + 2a^2b^2 + 2a^2c^2 + 2b^2c^2 = -\left(a^2 + b^2 - c^2\right)^2 + 4a^2b^2$$

$$= (2ab)^2 - \left(a^2 + b^2 - c^2\right)^2 = \left(2ab - a^2 - b^2 + c^2\right)\left(2ab + a^2 + b^2 - c^2\right)$$

$$= \left(c^2 - (a-b)^2\right)\left((a+b)^2 - c^2\right) = (c - a + b)(c + a - b)(a + b - c)(a + b + c)$$

Let $s = \frac{a+b+c}{2}$ be the semi-perimeter. Then:

$$a + b - c = 2(s - c), \quad b + c - a = 2(s - a), \quad c + a - b = 2(s - b), \quad a + b + c = 2s$$

Therefore:

$$\sigma = \frac{1}{2}\sqrt{2(s-a)\cdot 2(s-b)\cdot 2(s-c)\cdot 2s} = \frac{1}{2}\cdot 4\sqrt{s(s-a)(s-b)(s-c)} = 2A$$

by Heron's formula.

Now we compute d using equation (2.3) from Theorem 2.1. Since $S = 2A$ (doubled area), we have:

$$d = 2S\sigma + \frac{1}{2}\left[a^2\left(r_{12}^2 + r_{13}^2 - r_{23}^2\right) + b^2\left(r_{23}^2 + r_{12}^2 - r_{13}^2\right) + c^2\left(r_{13}^2 + r_{23}^2 - r_{12}^2\right)\right]$$

With $m_1 = a$, $m_2 = b$, $m_3 = c$ and using the fact that $r_{12} = c$, $r_{23} = a$, $r_{13} = b$:

$$d = 2(2A)(2A) + \frac{1}{2}\left[a^2\left(c^2 + b^2 - a^2\right) + b^2\left(a^2 + c^2 - b^2\right) + c^2\left(b^2 + a^2 - c^2\right)\right]$$

The second term simplifies:

$$\frac{1}{2}\left[a^2c^2 + a^2b^2 - a^4 + b^2a^2 + b^2c^2 - b^4 + c^2b^2 + c^2a^2 - c^4\right]$$

$$= \frac{1}{2}\left[2a^2b^2 + 2b^2c^2 + 2c^2a^2 - a^4 - b^4 - c^4\right]$$

But from our earlier factorization, we know:

$$2a^2b^2 + 2b^2c^2 + 2c^2a^2 - a^4 - b^4 - c^4 = 16\frac{A^2}{4} = (2\sigma)^2 = 16A^2$$

Therefore:

$$d = 8A^2 + \frac{1}{2} \cdot 16A^2 = 8A^2 + 8A^2 = 16A^2$$

Thus:

$$F_{min} = \sqrt{d} = \sqrt{16A^2} = 4A$$

Key Result: When $m_1 = a, m_2 = b, m_3 = c$, we have $d = 16A^2$ and $F_{min} = 4A$

Remark 3.2. This elegant result shows that for a triangle with weights equal to its side lengths, the minimum weighted sum of distances is exactly four times the area. This holds for acute-angled triangles where Theorem 1.1′s conditions are satisfied.

§4 Python Code for Numerical Verification

§4.1 Verification of weighted squared distances problem

Below is Python code that numerically finds the minimum and verifies it matches the analytical solution (the incenter).

```python
import numpy as np

def triangle_area(a, b, c):
    """Calculate triangle area using Heron's formula."""
    s = (a + b + c) / 2
    return np.sqrt(s * (s - a) * (s - b) * (s - c))

def create_triangle_vertices(a, b, c):
    """
    Create triangle vertices from side lengths.

    Places:
    - A at origin (0, 0)
    - B at (c, 0)
    - C calculated using law of cosines
    """
    A = np.array([0.0, 0.0])
    B = np.array([c, 0.0])

    # Law of cosines: cos(angle_A) = (b² + c² - a²) / (2bc)
    cos_angle_A = (b**2 + c**2 - a**2) / (2 * b * c)
    sin_angle_A = np.sqrt(1 - cos_angle_A**2)

    C = np.array([b * cos_angle_A, b * sin_angle_A])

    return A, B, C

def compute_function(P, A, B, C, a, b, c):
    """Compute f(P) = a·|AP|² + b·|BP|² + c·|CP|²"""
    AP_squared = np.sum((P - A)**2)
    BP_squared = np.sum((P - B)**2)
    CP_squared = np.sum((P - C)**2)
    return a * AP_squared + b * BP_squared + c * CP_squared

def compute_incenter(A, B, C, a, b, c):
    """Compute the incenter analytically."""
    return (a * A + b * B + c * C) / (a + b + c)

def compute_minimum_value(A, B, C, a, b, c):
    """Compute the minimum value using the formula."""
    return (a*b*c**2 + b*c*a**2 + c*a*b**2) / (a + b + c)

```

```
def brute_force_minimize(a, b, c, n_grid=1000):
    """Find minimum using brute force grid search."""
    A, B, C = create_triangle_vertices(a, b, c)

    min_value = float('inf')
    min_point = None

    # Grid search using barycentric coordinates
    for i in range(n_grid + 1):
        for j in range(n_grid + 1 - i):
            u = i / n_grid
            v = j / n_grid
            w = 1.0 - u - v

            # Convert barycentric to Cartesian
            P = w * A + u * B + v * C

            # Evaluate function
            f_value = compute_function(P, A, B, C, a, b, c)

            if f_value < min_value:
                min_value = f_value
                min_point = P

    return min_point, min_value

# Test triangles
test_triangles = [
    (5, 5, 5, "Equilateral"),
    (6, 7, 8, "Acute Scalene"),
    (3, 4, 5, "Right Triangle"),
]

print("=" * 80)
print("NUMERICAL VERIFICATION OF ANALYTICAL SOLUTION")
print("=" * 80)
print()

for a, b, c, name in test_triangles:
    print(f"{name} - sides: a={a}, b={b}, c={c}")
    print("-" * 80)

    # Create triangle
    A, B, C = create_triangle_vertices(a, b, c)

    # Analytical solution (incenter)
    incenter = compute_incenter(A, B, C, a, b, c)
    f_analytical = compute_function(incenter, A, B, C, a, b, c)
```

```
    f_formula = compute_minimum_value(A, B, C, a, b, c)

    # Numerical solution (brute force)
    P_numerical, f_numerical = brute_force_minimize(a, b, c, n_grid=1000)

    # Compare
    distance = np.linalg.norm(P_numerical - incenter)
    value_diff = abs(f_numerical - f_analytical)

    print(f"  Incenter (analytical):  ({incenter[0]:.6f}, {incenter[1]:.6f})")
    print(f"  Numerical minimum:      ({P_numerical[0]:.6f}, {P_numerical[1]:.6f})")
    print(f"  Distance between them:  {distance:.8f}")
    print()
    print(f"  f(incenter) analytical: {f_analytical:.6f}")
    print(f"  f(incenter) formula:    {f_formula:.6f}")
    print(f"  f(P) numerical:         {f_numerical:.6f}")
    print(f"  Difference:             {value_diff:.8f}")
    print()

print("=" * 80)
print("CONCLUSION")
print("=" * 80)
print()
print("✓ Numerical optimization finds the same point as the analytical")
print("  solution (the incenter)")
print()
print("✓ The minimum value formula is verified")
```

§4.2 Verification of Generalized Fermat-Torricelli Problem

The Python script verify_fermat_torricelli.py verifies the result $F_{min} = 4A$ for acute-angled triangles with weights equal to side lengths. Key features:

```python
# Excerpt from verify_fermat_torricelli.py
import numpy as np
from scipy.optimize import minimize

def fermat_torricelli_objective(P, A, B, C, m1, m2, m3):
    """Compute F(P) = m1*|PA| + m2*|PB| + m3*|PC|"""
    P = np.array(P)
    dist_A = np.linalg.norm(P - A)
    dist_B = np.linalg.norm(P - B)
    dist_C = np.linalg.norm(P - C)
    return m1 * dist_A + m2 * dist_B + m3 * dist_C

def verify_triangle(a, b, c, name):
    """Verify F_min = 4*Area for triangle with sides a, b, c"""
    # Calculate area using Heron's formula
    s = (a + b + c) / 2
```

```
    area = math.sqrt(s * (s - a) * (s - b) * (s - c))
    predicted_min = 4 * area

    # Set weights equal to side lengths
    m1, m2, m3 = a, b, c

    # Numerical optimization using L-BFGS-B
    result = minimize(
        fermat_torricelli_objective,
        x0=centroid,
        args=(A, B, C, m1, m2, m3),
        method='L-BFGS-B'
    )

    F_numerical = result.fun
    error = abs(F_numerical - predicted_min)

    print(f"Triangle {a}-{b}-{c}:")
    print(f"  4*Area = {predicted_min:.8f}")
    print(f"  F_min  = {F_numerical:.8f}")
    print(f"  Error  = {error:.2e}")

    return error < 1e-4  # Verification passes

# Test acute-angled triangles
test_triangles = [
    (5, 6, 7),
    (6, 7, 8),
    (7, 8, 9),
    (10, 11, 12),
]

for a, b, c in test_triangles:
    verify_triangle(a, b, c, f"Triangle {a}-{b}-{c}")
```

The complete script tests 8 acute-angled triangles and verifies that the numerical optimization result matches the theoretical prediction $F_{min} = 4A$ with errors less than 10^{-13}, confirming our analytical derivation.

Sample output:

```
Triangle 5-6-7:
  4*Area = 58.78775383
  F_min  = 58.78775383
  Error  = 7.11e-15
  [PASS] VERIFICATION PASSED
```

§5 References

[1] A. Y. Uteshev, "Analytical Solution for the Generalized Fermat–Torricelli Problem," *The American Mathematical Monthly*, vol. 121, no. 4, pp. 318–331, Apr. 2014, doi: 10.4169/amer.math.monthly.121.04.318.