```julia
function obj_val(x, y)
    return sin(x) * sin(y) + x / 7 * exp(-x^2 - y^2 / 50)
end

function update_velocity_position(x, v, pbest, gbest, w, phi1, phi2)
    r1, r2 = rand(), rand()
    v .= w .* v .+ phi1 .* r1 .* (pbest .- x) .+ phi2 .* r2 .* (gbest .- x)
    x .+= v
    return x, v
end

function PSO()
    num_particles, num_iterations = 4, 41
    w, phi1, phi2 = 0.95, 0.15, 0.08
    pbest_positions = [(4.11, -1.12), (4.73, -1.36), (1.87, 1.44), (4.68, -0.86)]
    positions = [(3.16, -1.73), (4.73, -1.36), (1.89, 1.35), (3.44, -0.88)]
    velocities = [(0.0, 0.0), (0.0, 0.3), (0.3, 0.0), (0.0, 0.0)]
    gbest_position = pbest_positions[argmax(obj_val.(pbest_positions))]

    for iteration in 1:num_iterations
        println("Iteration $iteration")
        for i in 1:num_particles
            positions[i], velocities[i] = update_velocity_position(positions[i],
                velocities[i], pbest_positions[i], gbest_position, w, phi1, phi2)
        end

        for i in 1:num_particles
            current_position = positions[i]
            current_fitness = obj_val(current_position...)
            if current_fitness > obj_val(pbest_positions[i]...)
                pbest_positions[i] = current_position
            end
        end

        gbest_position = pbest_positions[argmax(obj_val.(pbest_positions))]
        println("Particles: $(round.(positions, digits=2))")
        println("Best-known position: $(round.(gbest_position, digits=2))")
    end
end

PSO()
```

Listing 1: Particle Swarm Optimization in Julia