`optimize generalized.jl` implements an optimization algorithm to find periodic orbital configurations for multi-planet systems. The central goal is to identify initial conditions where planets return to the same relative positions after a specific integration time—creating a truly periodic system that would remain stable over long timescales.

1. Parameterization. I've chosen to parameterize the system with 5 key parameter types:

- **Inner planet period** (1 parameter): A scaling factor for the entire system
- **Eccentricities** (n_{planets} parameters): Orbital shapes for each planet
- **Arguments of perihelion** ($n_{\text{planets}} - 1$ parameters): Orbital orientations (first one fixed at 0)
- **Period ratio deviations** ($n_{\text{planets}} - 1$ parameters): How much each period ratio deviates from my nominal ratio (κ)
- **Mean anomalies** ($n_{\text{planets}} - 1$ parameters): Initial orbital phases (first one fixed at 0)

This gives a total of $1 + n_{\text{planets}} + (n_{\text{planets}} - 1) + (n_{\text{planets}} - 1) + (n_{\text{planets}} - 1)$ parameters.

2. Objective Function.

- Extracts the optimization parameters
- Sets up the initial planetary system
- Integrates it forward for a specific time (scaling with planet count as $2^{(n_{\text{planets}}-1)} \times$ actual period)
- Compares the initial and final states
- Returns a weighted sum of squared differences

The differences I measure are:

- Eccentricity changes
- Relative orientation (ω) changes
- Period ratio changes
- Mean anomaly changes

A truly periodic system should have minimal differences between initial and final states.

3. Weighting Strategy. I apply different weights to prioritize certain parameters:

```
weights = ones(length(differences))
rel__indices = (n_planets+1):(2*n_planets-1)
weights[rel__indices] .= 3.0
period_ratio_indices = (2*n_planets):(3*n_planets-2)
weights[period_ratio_indices] .= 2.0
mean_anomaly_indices = (3*n_planets-1):(4*n_planets-3)
weights[mean_anomaly_indices] .= 5.0
```

This reveals my priorities:

- Highest weight (5.0): Mean anomaly preservation
- High weight (3.0): Relative orientation (orbital alignment)
- Medium weight (2.0): Period ratio preservation
- Base weight (1.0): Eccentricity preservation

This weighting scheme makes sense—the most important aspect of periodicity is maintaining the mean anomalies (orbital phases), followed by the relative orientation of orbits.

4. Integration Time Scaling. A critical feature is how I scale the integration time with the number of planets:

```
integration_time = 2^(n_planets-1) * actual_period
```

This means:

- 2 planets: $2^1 = 2$ inner periods or 4π mean anomally
- 3 planets: $2^2 = 4$ inner periods or 8π mean anomally
- 4 planets: $2^3 = 8$ inner periods or 16π mean anomally

This is extremely important for proper verification of periodicity in systems with more planets, as the natural timescale for the system grows exponentially with the number of planets for resonant chains.

Implementation Details

I have some initial condiotion defaults but my function also lets the user override them with their own values. I use the Fminbox algorithm with NelderMead as its inner optimizer that:

- Doesn't require derivatives
- Works well with bounded parameters

My code implements a search for "fixed point solutions" to get the best parameters

Verification. After optimization, I verify the results by:

- Creating a fresh system with the optimized parameters
- Integrating it for the same time
- Checking differences between initial and final states
- Reporting these differences clearly

This allows me to confirm the true periodicity of the solution.