Approximating Kinetic Parameters Using Particle Swarm Optimization

Alen Lukic
Rice University

1 Introduction

In quantitative biological analysis, one way to model cellular networks is via a system of ordinary differential equations. Each equation represents a cellular process such as phosphorylation and dephosphorylation. The kinetic parameters of most such equation systems are largely unknown, and given that many models contain on the order of hundreds of equations, using the nave approach to determine these parameters is computationally infeasible, especially since the solution space is not discrete. Nonetheless, a good approximation of these parameters can still be useful. The mutual feedback system between executable models, such as Boolean network or Petri net representations of biological systems, and experimental data provides a way of refining these parameters given experimental data of cellular network behavior, and in turn refined parameters can be used as a hypothesis-generating mechanism for suggesting further experimentation. Particle swarm optimization (PSO) algorithms are a class of iterative improvement algorithms which search the solution space of a system [1]. Each particle is a representation of some sort of parameter of interest whose value is unknown; in this case, particles correspond to unknown kinetic parameters of cellular network ODE models. On any given iteration of the algorithm, a fitness metric scores each particle and the overall system based on particle positions (values) in the solution space, and then updates the velocity, or directional rate of value change, and position of each particle. The algorithm continues to iterate until a pre-defined termination condition is met: usually, either sufficient system fitness or a maximum number of iterations. The purpose of this project was to implement a PSO algorithm which makes use of different scoring metrics, investigate its ability to approximate kinetic parameters and estimate system behavior, and compare the results produced by PSO algorithms using different scoring metrics to experimental data. Each PSO implementation estimated the parameters of cellular systems whose behavior has been experimentally observed by simulating the system using the estimated parameters, comparing the simulated system behavior to the observed system behavior, and based on the score of the comparison, iteratively improving upon the parameter set.

2 Terminology and Algorithm Description

This section explains the terminology and gives a brief overview of the particle swarm optimization algorithm. The *position* of a particle is its current value within the search space. The search space itself is in this case a bounded domain. The *velocity* of a particle represents the direction its search is moving in and the speed of that movement; mathematically, it is expressed as a rate of change of position in dimensional space. A particle's *inertia* represents its tendency to keep searching the area of the solution space it is currently in. The algorithm adjusts this factor accordingly to prevent the particle from getting stuck in local optima. The algorithm takes several factors into account when determining how to update the position, velocity, and inertia of each particle at the end of each iteration. The velocity is updated based on the particles inertia, current velocity, the difference between the best position seen by the swarm and the particles current position, and the difference between the best position seen by the particle and the particles current position. The algorithm weighs the latter two factors with the coefficient of trust in the swarm and the coefficient of trust in self, respectively. These global constants represent how much confidence a particle has in its own observations versus how much confidence it has in the swarms observations. The position of the particle is then updated by adding the updated velocity to its old position. The inertia of the particle is updated according to how its score has changed recently; for example, if the particle is searching in a smaller, more local space and its score is not increasing, the inertia is increased to prompt a more global search; similarly, if it is searching in a larger space and its score is increasing consistently, the inertia is reduced to prompt search in a more local area. The generalized algorithm is as follows. The full-detail algorithm can be found in Appendix A.

- 1. Randomly initialize the position (within the solution space bounds) and velocity for each particle.
- 2. Find and store the score associated with the position of each particle using the scoring metric. Store the highest score among these scores as well; this is the global best score.
- 3. Use an update scheme to update the velocity and position of each particle, in that order. Recalculate the score of each particles new position and store it. Update the global best score if necessary.
- 4. Update the inertia of each particle if necessary.
- 5. Repeat steps 2 through 5 until the termination condition is met.

3 Scoring Metrics

The following scoring metrics were compared in this experiment. These metrics compared the transitory behavior of select reactants from the experimental data with the transitory behavior of these same reactants from the PSO-produced concentration curves.

3.1 Euclidean distance

This simple scoring metric compared the Euclidean distance between corresponding concentration curves in the experimental data and the simulated data; the N curves are each represented by a time series of T steps. The score was calculated by taking the inverse of the sum of the distances (that is, concentration differences) between corresponding time series points on the experimental and simulated curves. At time j, call the concentration $y(j)$. The formula for the score is

$$\frac{N}{\sum_{x=1}^{N}\sqrt{\sum_{j=1}^{T}|y_{true}(j) - y_{simulated}(j)|^2}}$$

3.2 Discrete Fourier Transform (DFT)

This scoring metric transformed the time-series data into the frequency domain; since the data is discrete, the DFT was chosen to perform this transformation. In signal processing, the DFT is commonly used to analyze the power of frequencies in a sampled signal, though it also has other applications. Applied to curves representing biological data, the idea is to analyze the transformation and derive some feature of the biological system. Each time-series curve was transformed into a set of Fourier coefficients, complex-number representations of the magnitude and phase of a sinusoidal time-series function. This is a key point; the DFT assumes that the time-series data being transformed is periodic, or oscillatory. Take the amount of time steps represented by the curve to be T. The formula for calculating the Fourier coefficients c_f, $f = 1, \ldots, T$ is

$$c_f = \frac{1}{\sqrt{N}}\sum_{x=0}^{N} f(x)e^{\frac{-2\pi i x}{N}}$$

where $i = \sqrt{-1}$. Theoretically, the Euclidean distance between the squares of the Fourier coefficients of systems with the same features should be zero. Hence, the score of this metric is calculated by summing this distance for each set of corresponding curves in the experimental data and the simulated data, and taking the inverse of this sum, as follows

$$\frac{N}{\sum_{x=1}^{N}\sqrt{\sum_{f=1}^{K}|c_{f true}(j) - c_f simulated(j)|^2}}$$

where $k = \frac{T}{4}$; that is, only the power of the lower frequencies of the curves was considered in the metric. Lower frequencies represent longer-term, rather than transitory, processes in the system and as such may be of interest in determining which processes act on the system in a sustained fashion; this requires further investigation.

3.3 Polynomial fitting

This scoring metric used a built-in Python library (NumPy) to calculate a best-fit fourth-degree polynomial for both the experimental and simulated data curves. The score was then calculated by taking the difference between corresponding term coefficients of both polynomials and summing these differences for each curve, calculating the mean of these differences, and then taking the inverse of the mean. Let a_i represent the coefficient of the i^{th}- order term in the polynomial and let D represent the degree of the polynomial. Then the formula for the score is

$$\frac{N}{\sum_{x=1}^{N}(\sum_{i=0}^{D}|a_{itrue}-a_{isimulated}|)}$$

4 Methodology

The biological systems utilized for comparison purposes in this experiment were the Janus-Kinase Signal Transducer and Activator of Transcription (JAK-STAT) signaling pathway [2], whose transitory reactions quickly achieve steady-state according to experimental data, and the mitogen-activated protein kinase (MAPK) cascade with a negative feedback loop [3], which cause sustained oscillatory behavior in MAPK phosphorylation. The MAPK system in particular has been studied extensively due to its role in a wide variety of important cellular processes. Our hypothesis was that the Euclidean distance metric would perform well for both systems, that the polynomial fitting metric would perform well for the JAK-STAT system, and that the DFT feature extraction metric would perform well for the oscillatory MAPK cascade. We first implemented the general particle swarm optimization algorithm in Python; the global parameters which we used and their justification can be found in Appendix B. We ran five instances of the algorithm for each scoring metric and for both of our reference biological systems. Our goal was convergence of the simulated system behavior with the experimental data; however, one of our termination conditions was a lack of change in cumulative score for 2,000 consecutive iterations of the algorithm. The scoring mechanism was implemented by simulating the biological system with the current set of kinetic parameters using COPASI, an open-source software for simulating biological networks. The time-series data returned by this simulation was then compared to the experimentally obtained time-series data, and a score was determined based on the particular scoring metric being used. After all instances of the algorithm finished computing, we selected the instances which returned the best overall score and the worst overall score. We used the respective kinetic parameter sets of these instances to simulate the biological system in question in COPASI. We plotted and saved the transient concentrations of the systems reactants of interest for both the best and worst-scored runs of the algorithm and compared them to experimental results.

5 Results

Oscillatory MAPK system

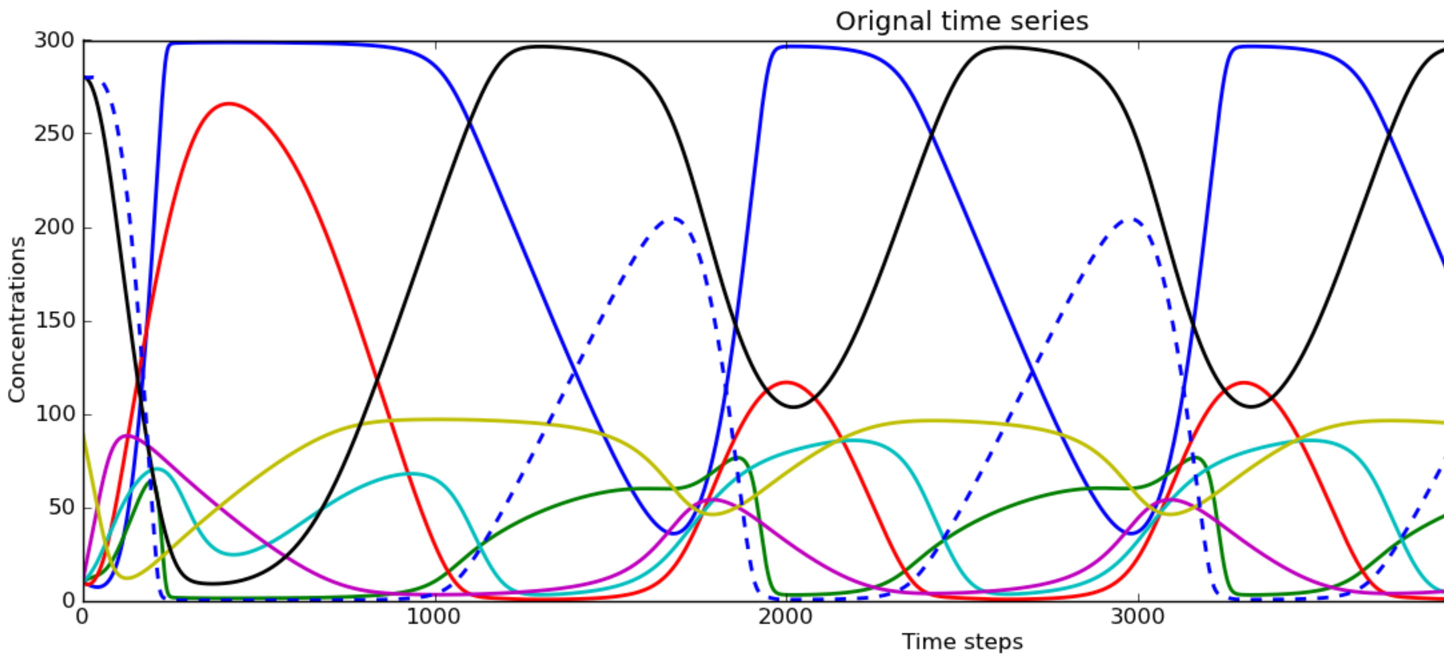

Figure 1: Experimental data of transitory concentrations of select reactants in oscillatory MAPK cascade system.

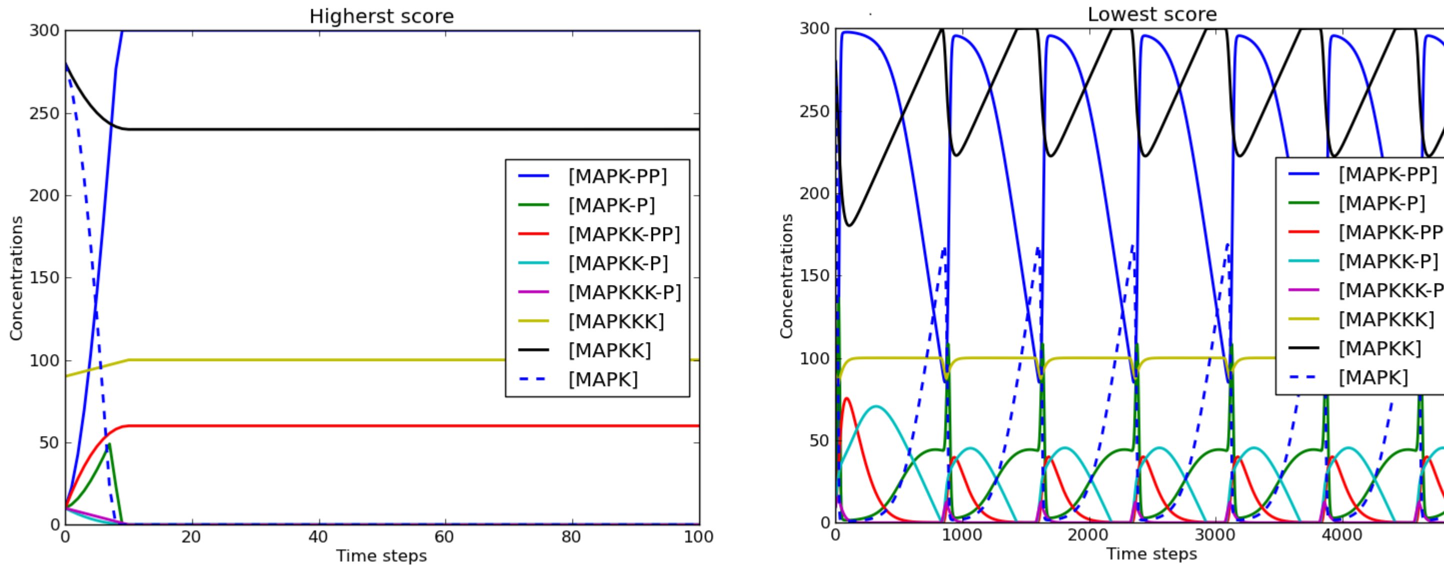

Figure 2: Best and worst-scored parameter set oscillatory MAPK system behavior Euclidean distance scoring metric.

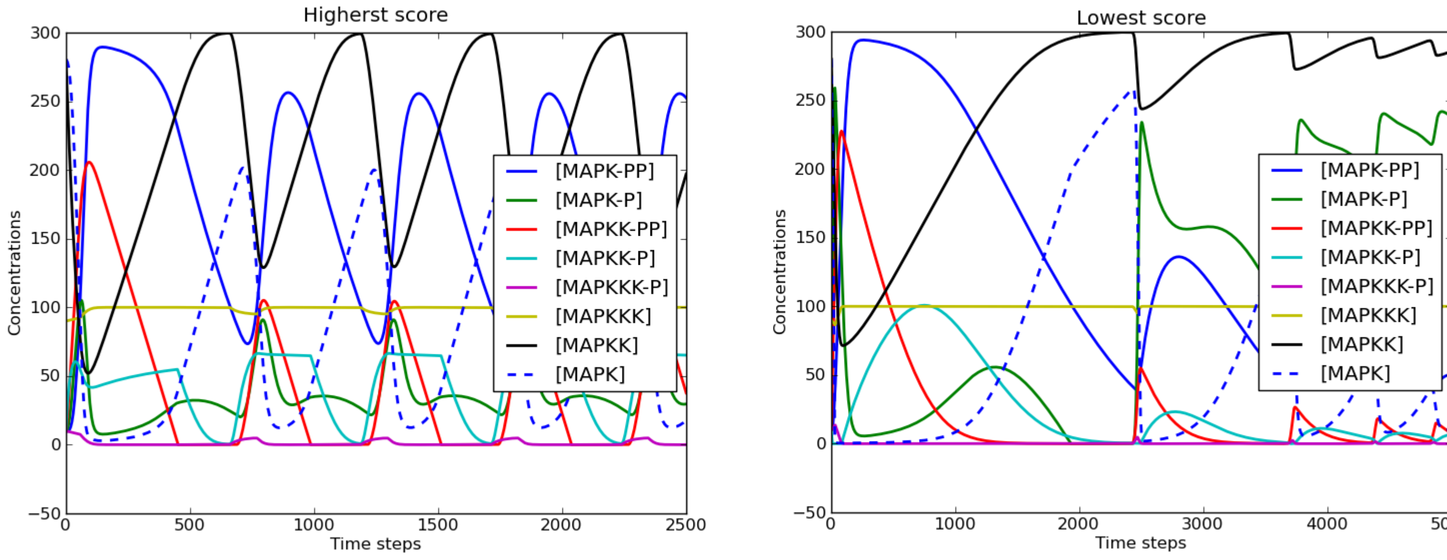

Figure 3: Best and worst-scored parameter set oscillatory MAPK system behavior DFT scoring metric.

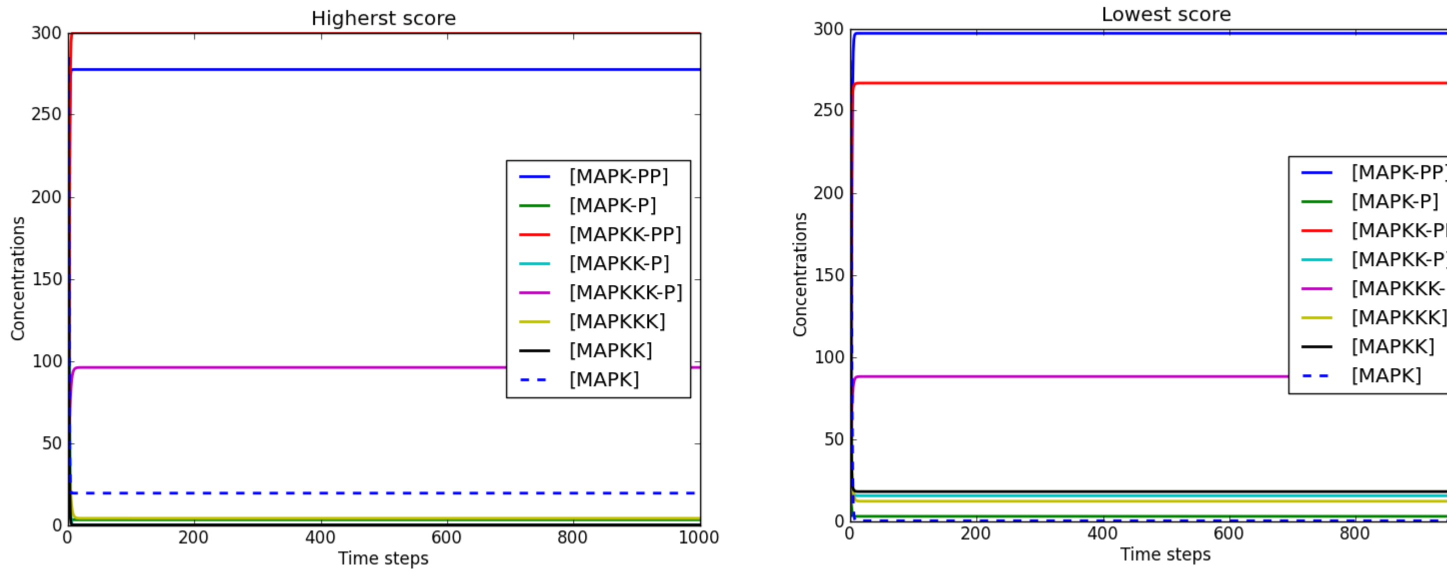

Figure 4: Best and worst-scored parameter set oscillatory MAPK system behavior polynomial fitting scoring metric.

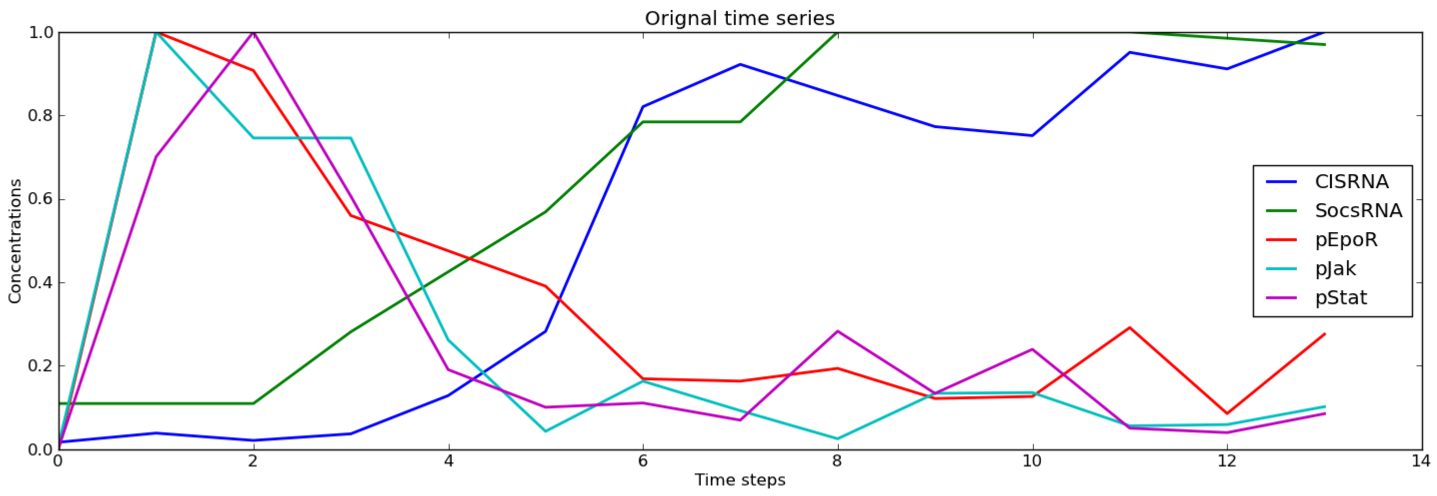

Figure 5: Experimental data of transitory concentrations of select reactants in JAK-STAT system.

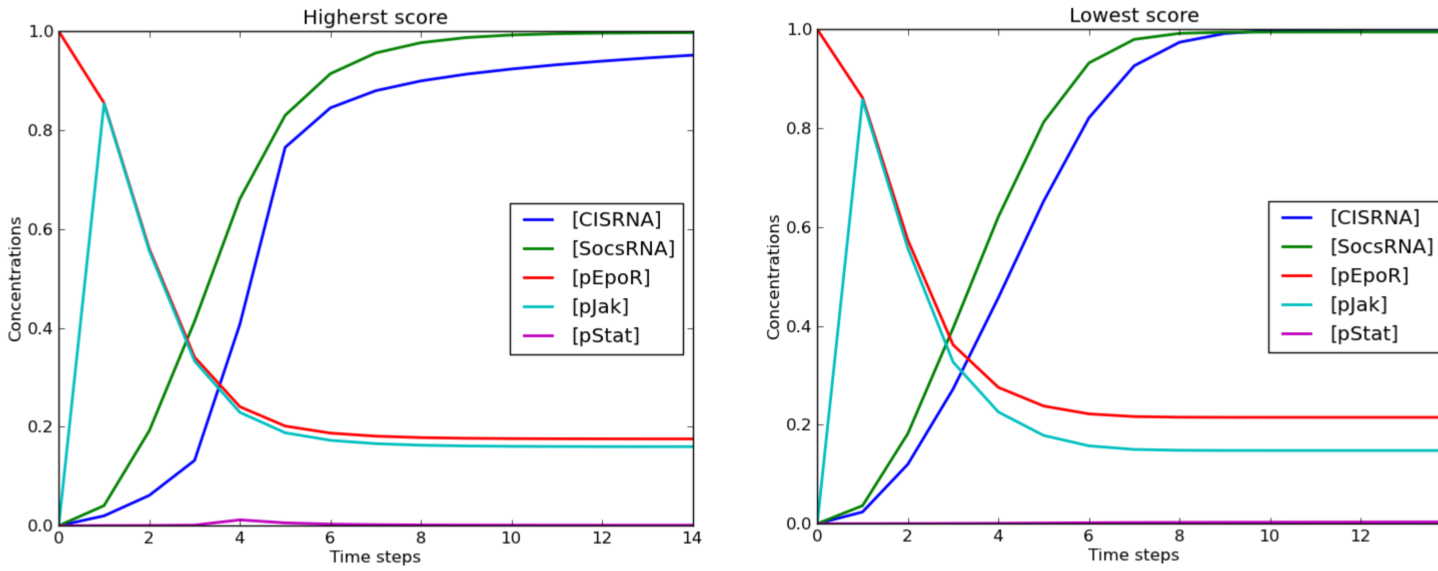

Figure 6: Best and worst-scored parameter set JAK-STAT system behavior Euclidean distance scoring metric.

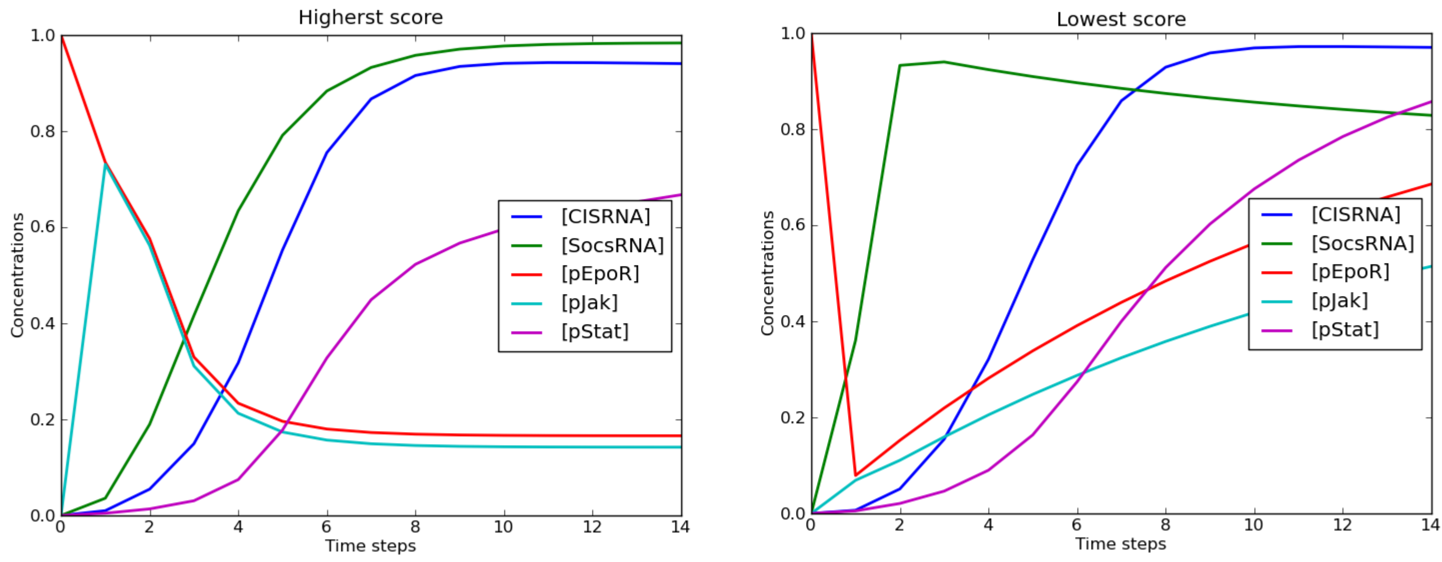

Figure 7: Best and worst-scored parameter set JAK-STAT system behavior DFT scoring metric.

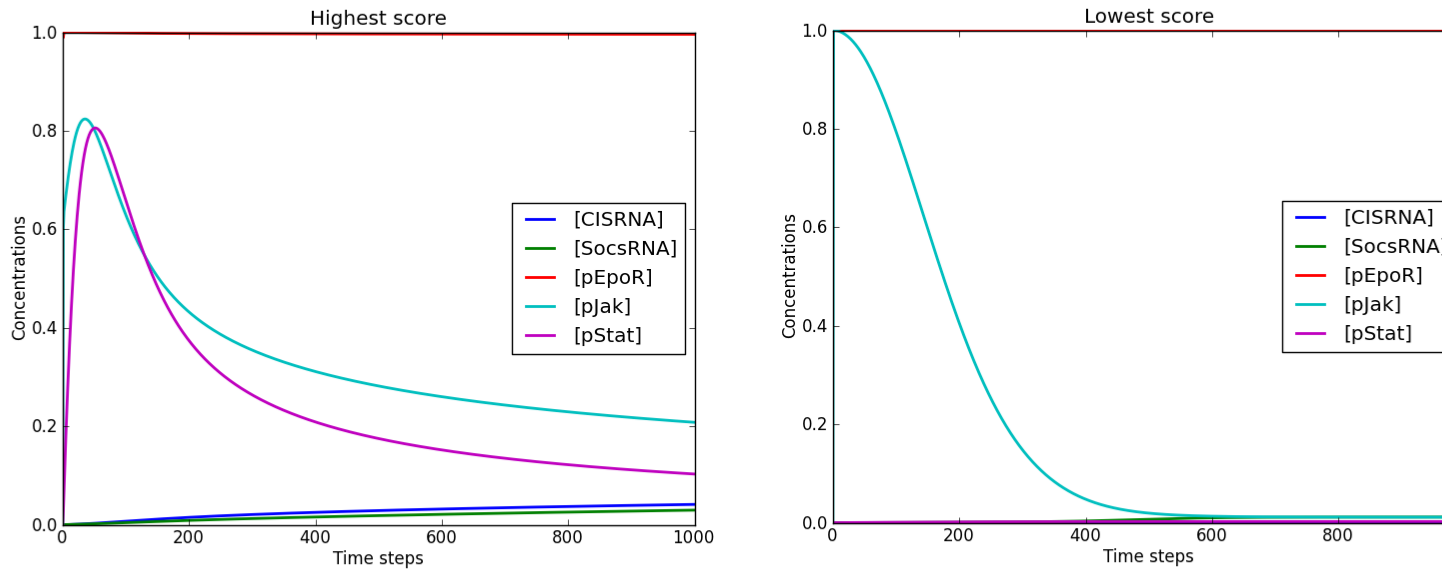

Figure 8: Best and worst-scored parameter set JAK-STAT system behavior polynomial fitting metric.

5.1 Analysis and Conclusions

Of the three scoring metrics, the DFT feature extraction metric seems to have the best performance for the oscillatory MAPK system, and performs about as well as the Euclidean distance metric for the JAK-STAT system. Again, it is not surprising that a particle swarm optimization algorithm using a DFT scoring metric would perform well on oscillatory time-series data, as the DFT assumes that the input data which it is transforming is oscillatory. The Euclidean distance metric performed well for JAK-STAT but not MAPK; interestingly, the worst-scored parameter set produced by the Euclidean- based PSO yielded system behavior closer to the experimental data tan the best-scored parameter set for the MAPK system. This may be an indication that a Euclidean distance-based scoring metric is not appropriate for oscillatory systems. The polynomial fitting metric-based PSO instance did not perform well for either system. On the one hand, this is not surprising in the case of the oscillatory MAPK system; it does not make sense to fit a polynomial to a sinusoid, as they are fundamentally different elementary functions. Similarly, a transient reaction which tends toward a steady-state value is likely also poorly described by a polynomial, which does not exhibit such behavior. Thus, of the conjectures in our hypothesis, only the performance of the DFT-based PSO on the oscillatory MAPK system was confirmed. The Euclidean-based PSO only did well for JAK-STAT and not MAPK, while the polynomial-based PSO did not do well for either system. An interesting feature of the results is that there is not a substantial difference between the results produced by the highest-scored parameter sets and those produced by the lowest-scored parameter sets. It is important to keep in mind that PSO is an approximation algorithm which does not guarantee to return an optimal value, and that there is not a one-to-one mapping between score and value. It is not sufficient to evaluate the goodness of the parameter set produced by a particular instance of a PSO execution by using only the score of that set. The accuracy of the DFT-based PSO for both systems and the Euclidean distance-based PSO for JAK-STAT in light the fact that each PSO instance was initialized with random particle positions and velocities and used no pre-existing knowledge of the biological systems or their parameters is quite remarkable. Particle swarm optimization algorithms merit further investigation in the context of cellular network model parameter estimation. Given the success of the DFT scoring metric, one possible area of investigation is other function transformation-based metrics, such as Legendre transformations.

5.2 Acknowledgements and References

This work was done in conjunction with Natalie Berestovsky, a graduate student in the Rice University computer science department, and Riya Fukui, an undergraduate student in the Rice computer science department.

1 Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. Particle swarm optimization. AIAA, 1235, 2002.

2 Julie Bachmann, Andreas Raue, Marcel Schilling, Martin E. Bohm, Clemens Kreutz, Daniel Kaschek, Hauke Basch, Norbert Gretz, Wold D Lehmann, Jens Timmer, and Ursula Klingmuller. Division of labor by dual feedback regulator controls jak2/stat5 signaling over broad ligand

range. Molecular Systems Biology, 7(516), 2011.

3 Chi-Ying Huang and James E. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. Proc. Natl. Acad. Sci, 93:10078-10083, September 1996.

4 Sergio Iadevaia, Yiling Lu, Fabiana C. Morales, et. al. Identification of Optimal Drug Combinations Targeting Cellular Networks: Integrating Phospho-Proteomics and Computational Network Analysis. Cancer Res 2010, 70:6704-6714, July 2010.

APPENDIX A

Algorithm 1 General Particle Swarm Optimization

Input Set of $P = \{1, \ldots, m\}$ particles; dimensionality n; coefficients of trust in self and swarm c_1 and c_2; local and global search thresholds s_l and s_g; lower and upper inertia bounds w_l and w_h; lower and upper search space bounds *min* and *max*; maximum number of iterations $iter_{max}$; a scoring function $f(p)$; and a sufficient fitness score $score_{goal}$.

Output Set of $S = \{1, \ldots, m\}$ parameters.

```
$score_{cur}$, $score_{best}$, $pos_{best}$, $iter_{cur} \leftarrow 0$
for all $p \in P$ do
    $p_p \leftarrow rand(min, max)$
    $p_w \leftarrow rand(w_l, w_h)$
    $p_v \leftarrow p_p - p_w$
    $p_c \leftarrow 0$
    $p_{pbest} \leftarrow p_p$
    $p_{sbest} \leftarrow f(p_p)$
    $S \leftarrow S \cup p$
end for        ▷ Initialize initial position, velocity, and inertia to random values within their appropriate lower and upper bounds. Initialize particle's best position to its initial random position and its score using the scoring function. Initialize a counter for each particle to 0.
while ($score_{cur} < score_{goal}$) or ($iter_{cur} < iter_{max}$) do
    return PSO Update Scheme($S$)
    $iter_{cur} \leftarrow iter_{cur} + 1$
end while
return $S$
```

Algorithm 2 PSO Update Scheme

Input Set of $S = \{1, \dots, m\}$ particles.

Output Set of $S = \{1, \dots, m\}$ particles with updated parameters.

```
for all $p \in S$ do
    $r1$, $r2 \leftarrow rand(0, 1)$
    $p_v \leftarrow (p_w * p_v) + (c_1 * r_1)(p_{pbest} - p_p) + (c_2 * r_2)(pos_{best} - p_p)$
    $p_p \leftarrow p_w + v_p$
    if $(p_i > max)$ or $(p_i < min)$ then
        $p_v \leftarrow (c_1 * r_1)(p_{pbest} - p_p) + (c_2 * r_2)(pos_{best} - p_p)$
        $p_p \leftarrow p_v + p_p$
    end if
    $s_{new} \leftarrow f(p_p)$
    if $s_{new} > p_{psbest}$ then
        $p_{pbest} \leftarrow p_p$
        $p_{sbest} \leftarrow s_{new}$
        $p_c \leftarrow p_c + 1$
    else
        $p_c \leftarrow p_c - 1$
    end if
    if $s_{new} > score_{best}$ then
        $score_{best} \leftarrow s_{new}$
        $pos_{best} \leftarrow p_p$
    end if
    if $p_c > s_g$ then
        $p_w \leftarrow \frac{p_w}{2}$
        if $p_w < w_l$ then
            $p_w \leftarrow w_l$
        end if
    else if $p_c < s_l$ then
        $p_w \leftarrow 2 * p_w$
        if $p_w > w_h$ then
            $p_w \leftarrow w_h$
        end if
    end if
end for
```

APPENDIX B

We used the following global constants in our implementation of PSO:

$min = 0.001$, $max = 1000$

$c_1 = c_2 = 1.49$

$w_l = 0.1$, $w_h = 1.1$

$s_l = 3$, $s_h = 5$

Termination conditions: convergence, or no change in score for 2,000 consecutive iterations.

These values were suggested by Iadevaia, et. al., and worked well with our PSO implementation.