SOFTWARE

JuPOETs: A Constrained Multiobjective Optimization Approach to Estimate Biochemical Model Ensembles in the Julia Programming Language

David Bassen[2], Michael Vilkhovoy[1], Mason Minot[1], Jonathan T Butcher[2] and Jeffrey D Varner[1*]

*Correspondence:
jdv27@cornell.edu
[1]Department of Chemical and Biomolecular Engineering, Cornell University, 14853 Ithaca NY, USA
Full list of author information is available at the end of the article

Abstract

Ensemble modeling is a well established approach for dealing with parameter uncertainty in large-scale deterministic models. In this study, we present an open-source implementation of the Pareto Optimal Ensemble Technique in the Julia programming language (JuPOETs). JuPOETs integrates simulated annealing with Pareto optimality to estimate parameter sets on or near the optimal tradeoff surface between competing training objectives. JuPOETs can be installed using the Julia package manager from the JuPOETs GitHub repository at https://github.com/varnerlab/POETs.jl.

Second part title: Text for this section.

Keywords: Ensemble modeling; Multiobjective optimization; Julia

Background

Ensemble modeling is a well established approach for dealing with parameter uncertainty in large-scale deterministic models. It is often not possible to uniquely identify biochemical model parameters from noisy experimental measurements, even when given extensive training data [1]. Thus, despite significant advances in standardizing biochemical model identification [2], the problem of estimating model parameters from experimental data remains challenging. Ensemble approaches have been used to address parameter uncertainty in systems biology and other fields like weather prediction [3–6]. In an ensemble approach, a family of parameter sets is identified instead of a single best-fit parameter set. Parameter families can be selected based upon simulation error, along with other criterion such as diversity. Simulations using parameter ensembles can estimate confidence intervals on model variables, and robustly constrain model predictions, despite having many poorly constrained parameters [7, 8]. There are several techniques to generate parameter ensembles. Battogtokh et al., Brown et al., and later Tasseff et al. generated experimentally constrained parameter ensembles using a Metropolis-type random walk through parameter space [3, 5, 9, 10]. Other strategies could also be adapted to generate parameter ensembles [11]. However, the unifying component of all of these previous strategies was the minimization of a single objective function.

Identification of large biochemical models with hundreds or even thousands of states and parameters may not be tractable as a single objective formulation. Large models require significant training data perhaps taken from diverse sources, for example different laboratories or cell-lines. These diverse training data are likely heterogenous, and contain intrinsic conflicts which complicate parameter estimation. On the other hand, large models can also be constructed from many smaller modules, each of which is independently identified. For example, Diamond and coworkers developed a model of calcium signaling in human platelets composed of many separate submodels identified using different types of training data [12]. Parameter ensemble estimation techniques which optimally balance tradeoffs between submodels or conflicts in training data lead to robust model performance. One class of such techniques is multiobjective optimization. Previously, we developed the Pareto Optimal Ensemble Technique (POETs) algorithm to address the challenge of competing or conflicting objectives. POETs, which integrates simulated annealing (SA) and multiobjective optimization through the notion of Pareto rank, finds a family of parameter estimates which optimally trade-off between competing (and potentially conflicting) experimental objectives [13]. POETs has been used in several biochemical and signal transduction studies to estimate parameter ensembles [14, 15]. However, the previous implementation of POETs, in the Octave programming language [16], suffered from poor performance, and did not offer user definable functions to accommodate custom cooling schedules, parameter constraints, or custom search logic. Octave-POETs was also not well integrated into a modern package or source code management (SCM) system, thus upgrades containing new features, or bug fixes were not centrally managed.

Implementation

In this study, we present an open-source implementation of the Pareto optimal ensemble technique in the Julia programming language (JuPOETs). JuPOETs offers many advantages and improvements when compared to Octave-POETs. First, JuPOETs takes advantage of the unique performance features of the Julia programming language. Julia, which has performance comparable to C but with syntax similar to MATLAB/Octave and Python, is a cross-platform, high-performance programming language for technical computing [17]. Julia offers a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library. Additionally, Julia offers a built-in package manager which is directly integrated with GitHub, a popular web-based Git repository hosting service which offers distributed revision control and source code management. Next, because Julia can natively call other languages such as Python or C, JuPOETs can be used with models implemented in a variety of languages on many platforms. Lastly, the architecture of JuPOETs takes advantage of the first-class function type in Julia allowing user definable behavior for all key aspects of the algorithm, including objective functions, custom search logic, linear/non-linear parameter constraints (and parameter bounds constraints) as well as custom cooling schedule functions. Thus, JuPOETs can easily be adapted to solve many problem types, including mixed binary and continuous variable types, without the need to change the base algorithm (which was not true of the previous POETs implementation).

JuPOETs optimization problem formulation.

JuPOETs solves the $\mathcal{K}-$dimensional constrained multiobjective optimization problem:

$$\min_{\mathbf{p}} \begin{cases} O_1\left(\mathbf{x}(t,\mathbf{p}),\mathbf{p}\right) \\ \vdots \\ O_{\mathcal{K}}\left(\mathbf{x}(t,\mathbf{p}),\mathbf{p}\right) \end{cases} \tag{1}$$

subject to:

$$\begin{aligned} \mathbf{f}(t,\mathbf{x}(t,\mathbf{p}),\dot{\mathbf{x}}(t,\mathbf{p}),\mathbf{u}(t),\mathbf{p}) &= \mathbf{0} \\ g_1\left(t,\mathbf{x}(t,\mathbf{p}),\mathbf{u}(t),\mathbf{p}\right) &\geq 0 \\ &\vdots \\ g_{\mathcal{C}}\left(t,\mathbf{x}(t,\mathbf{p}),\mathbf{u}(t),\mathbf{p}\right) &\geq 0 \end{aligned}$$

and parameter bound constraints:

$$\mathcal{L} \leq \mathbf{p} \leq \mathcal{U}$$

using a modified simulated annealing approach. The quantity t denotes time, $\mathbf{x}\left(t,\mathbf{p}\right)$ denotes the model state (with an initial state $\mathbf{x}_0$), and $\mathbf{u}(t)$ denotes an input vector. The terms $\mathbf{f}(t,\mathbf{x}(t,\mathbf{p}),\dot{\mathbf{x}}(t,\mathbf{p}),\mathbf{u}(t),\mathbf{p})$ denote the system of model equations (e.g., differential equations, differential algebraic equations or linear/non-linear algebraic equations) where $\mathbf{p}$ denotes the unknown parameter vector ($\mathcal{D}\times\ 1$). The parameter search can be subject to parameter bound constraints, where $\mathcal{L}$ and $\mathcal{U}$ denote the lower and upper parameter bounds, respectively as well as $\mathcal{C}$ problem specific constraints $g_i\left(t,\mathbf{x}(t,\mathbf{p}),\mathbf{u}(t),\mathbf{p}\right), i = 1,\dots,\mathcal{C}$.

JuPOETs integrates simulated annealing with Pareto optimality to estimate parameter sets on or near the optimal tradeoff surface between competing training objectives (Fig. 1 and Algorithm 1). The central idea of POETs is a mapping between the value of the objective vector evaluated at $\mathbf{p}_{i+1}$ (parameter guess at iteration $i+1$) and Pareto rank. JuPOETs calculates the performance of a candidate parameter set $\mathbf{p}_{i+1}$ by calling the user defined `objective` function; `objective` takes a parameter set as an input and returns a $\mathcal{K}\times 1$ objective vector. Candidate parameter sets are generated by the user supplied `neighbor` function. The error vector associated with $\mathbf{p}_{i+1}$ is ranked using the builtin Pareto `rank` function, by comparing the current error at iteration i+1 to the error archive $\mathcal{O}_i$. Pareto rank is a measure of distance from the trade-off surface; parameter sets on or near the optimal trade-off surface between the objectives have a rank equal to 0 (no other current parameter sets are better). Sets with increasing non-zero rank are progressively further away from the optimal trade-off surface. Thus, a parameter set with a rank$=0$ is *better* in a trade-off sense than rank>0. We implemented the Fonseca and Fleming ranking scheme [18] in the builtin `rank` function:

$$\texttt{rank}\left(\mathbf{p}_{i+1} \mid \mathcal{O}_i\right) = r \tag{2}$$

where rank r is the number of parameter sets that dominate (are better than) parameter set $\mathbf{p}_{i+1}$. We used the Pareto rank to inform the SA calculation. The parameter set $\mathbf{p}_{i+1}$ was accepted or rejected by the SA, by calculating an `acceptance` probability $\mathcal{P}\left(\mathbf{p}_{i+1}\right)$:

$$\mathcal{P}(\mathbf{p}_{i+1}) \equiv \exp\left\{-\texttt{rank}\left(\mathbf{p}_{i+1} \mid \mathcal{S}_i\right)/T\right\} \tag{3}$$

where T is the computational annealing temperature. As $\texttt{rank}\left(\mathbf{p}_{i+1} \mid \mathcal{O}_i\right) \rightarrow 0$, the acceptance probability moves toward one, ensuring that we explore parameter sets along the Pareto surface. Occasionally (depending upon T) a parameter set with a high Pareto rank was accepted by the SA allowing a more diverse search of the parameter space. However, as T is reduced, the probability of accepting a high-rank set occurring decreases. Parameter sets could also be accepted by the SA but *not* permanently archived in $\mathcal{S}_i$. Only parameter sets with rank less than or equal to threshold (rank$\leq$4 by default) were included in $\mathcal{S}_i$, where the archive was re-ranked and filtered after every new parameter set was accepted. Parameter bounds were implemented in the `neighbor` function as box constraints, while problem specific constraints were implemented in `objective` using a penalty method:

$$O_i + \lambda\sum_{j=1}^{\mathcal{C}}\min\Big\{0, g_j\left(t, \mathbf{x}(t, \mathbf{p}), \mathbf{u}(t), \mathbf{p}\right)\Big\} \qquad i = 1, \ldots, \mathcal{K} \tag{4}$$

where λ denotes the penalty parameter ($\lambda = 100$ by default). However, because both the neighbor and objective functions are user defined, different constraint implementations are easily defined. JuPOETs can be installed using the Julia package manager from the JuPOETs repository at https://github.com/varnerlab/POETs.jl.

input : User specified neighbor, objective, acceptance and cooling functions. Initial parameter guess ($\mathcal{P} \times 1$)
Output: Rank archive $\mathcal{R}$, solution archive $\mathcal{S}$ and objective archive $\mathcal{O}$

initialize: $\mathcal{R}$, $\mathcal{S}$ and $\mathcal{O}$ using initial guess;
initialize: T ←1.0;
initialize: T_{min} ←1/10000;
initialize: Maximum number of steps per temperature $\mathcal{I}$;

while $T > T_{min}$ **do**
  i ← 1;
  while $i < \mathcal{I}$ **do**
    // Generate a new parameter solution using user neighbor function
    $\mathbf{p}_{i+1} \leftarrow$ user-function::`neighbor`($\mathbf{p}^*$);
    // Evaluate $\mathbf{p}_{i+1}$ using user objective function
    $\mathbf{o}_{i+1} \leftarrow$ user-function::`objective`($\mathbf{p}_{i+1}$);
    Add $\mathbf{p}_{i+1}$ to solution archive $\mathcal{S}$;
    Add $\mathbf{o}_{i+1}$ to objective archive $\mathcal{O}$;
    // Calculate Pareto rank of solutions in $\mathcal{O}$ using builtin rank function
    $\mathcal{R} \leftarrow$ builtin-function::`rank`($\mathcal{O}$);
    // Accept $\mathbf{p}_{i+1}$ into the archive with user defined probability
    $\mathcal{P} \leftarrow$ user-function::`acceptance`($\mathcal{R}$,T);
    if $\mathcal{P} > rand$ **then**
      // Update the best solution with $\mathbf{p}_{i+1}$
      $\mathbf{p}^* \leftarrow \mathbf{p}_{i+1}$;
      prune $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{O}$ of all solutions above a rank threshold;
    else
      Remove $\mathbf{p}_{i+1}$ from solution archive $\mathcal{S}$;
      Remove $\mathbf{o}_{i+1}$ from error archive $\mathcal{O}$;
    end
    i ← i + 1;
  end
  // Update T using the user cooling function
  $T \leftarrow$ user-function::cooling(T);
end

Algorithm 1: Pseudo-code for the main run-loop of JuPOETs. The user specifies the `neighbor`, `acceptance`, `cooling` and `objective` functions along with an initial parameter guess. The rank archive $\mathcal{R}$, solution archive $\mathcal{S}$ and objective archive $\mathcal{O}$ are initialized from the initial guess. The initial guess is perturbed in the `neighbor` function, which generates a new solution whose performance is evaluated using the user supplied `objective` function. The new solution and objective values are then added to the respective archives and ranked using the builtin `rank` function. If the new solution is accepted (based upon a probability calculated with the user supplied `acceptance` function) it is added to the solution and objective archive. This solution is then perturbed during the next iteration of the algorithm. However, if the solution is not accepted, it is removed from the archive and discarded. The computational temperature is adjusted using the user supplied `cooling` function after each $\mathcal{I}$ iterations.

Results and Discussion

JuPOETs identified optimal or nearly optimal solutions significantly faster than Octave-POETs for a suite of test problems (Table 1). The wall-clock time for JuPOETs and Octave-POETs was measured for 10 independent trials for each of the test problems. The same `cooling`, `neighbor`, `acceptance`, and `objective` logic was employed between the implementations, and all other parameters were held constant. For each test function, the search domain was partitioned into 10 segments, where an initial parameter guess was drawn from each partition. The number of search steps for each temperate was $\mathcal{I} = 10$ for all cases, and the cooling parameter was $\alpha = 0.9$. On average, JuPOETs identified optimal or near optimal solutions for the suite of test problems six-fold faster (60s versus 400s) than Octave-POETs (Fig. 2). JuPOETs produced the characteristic tradeoff curves for each test problem, given both parameter bound and problem constraints (Fig. 3). Thus, JuPOETs estimated an ensemble of solutions to constrained multiobjective optimization test problems significantly faster than the current Octave implementation. Next, we tested JuPOETs on a proof-of-concept biochemical model identification problem.

JuPOETs estimated an ensemble of biochemical models that was consistent with the mean of synthetic training data (Fig. 4). Four synthetic training data sets were generated from a prototypical biochemical network consisting of 6 metabolites and 7 reactions (Fig. 4, inset right). We considered a common case in which the same measurements were made on four hypothetical cell types, each having the same biological connectivity but different performance. Network dynamics were modeled using the hybrid cybernetic model with elementary modes (HCM-EM) approach of Ramkrishna and coworkers [19]. In the HCM-EM approach, metabolic networks are first decomposed into a set of elementary modes (EMs) (chemically balanced steady-state pathways, see [20]). Dynamic combinations of elementary modes are then used to characterize network behavior. Each elementary mode is catalyzed by a pseudo enzyme; thus, each mode has both kinetic and enzyme synthesis parameters. The proof of concept network generated 6 EMs, resulting in 13 model parameters. The synthetic data was generated by randomly varying these parameters. JuPOETs produced an ensemble of models that captured the mean of the measured data sets for extracellular metabolites and cellmass (Fig. 4A and B). The 95% confidence estimate produced by the ensemble was consistent with mean of the measured data, despite having significant uncertainty in the training data. JuPOETs produced a consensus estimate of the synthetic data by calculating optimal trade-offs between the training data sets (Fig. 4C). Thus, JuPOETs produced an ensemble of parameter sets that gave the mean of the training data for conflicting data sets.

JuPOETs is a significant advancement over Octave-POETs, both in terms of performance and capability. However, there are several areas that could be explored further. First, JuPOETs should be compared with other multiobjective evolutionary algorithms (MOEAs) to determine its relative performance on test and real world problems. Many evolutionary approaches e.g., the nondominated sorting genetic algorithm (NSGA) family of algorithms, have been adapted to solve multiobjective optimization problems [REF]. It is unclear if JuPOETs will perform as well as these other approaches; one potential advantage that JuPOETs may have is the local refinement step which temporarily reduces the problem to a single objective formulation. Previously, this hybrid approach led to better convergence on a proof-of-concept signal transduction model [REF]. Next, JuPOETs should take advantage of the native parallel execution capabilities of Julia to accelerate the evaluation of the objective functions. For many real world parameter estimation problems, the bulk of the execution time is spent evaluating the objective functions. One strategy to improve performance could be to optimize surrogates [REF], while another would be parallel execution of the objective functions. Currently, JuPOETs serially evaluate the components of the objective function vector. However, because of the flexible function pointer architecture of JuPOETs, a shift to parallel evaluation of the objective functions requires changes to only the user defined objective function and not the main run loop. Thus, parallel evaluation of objective functions could be easily implemented using a variety of techniques without changing to JuPOETs.

Conclusions

Competing interests
The authors declare that they have no competing interests.

Funding
This study was supported by an award from the National Science Foundation (NSF CBET-0955172) and the National Institutes of Health (NIH HL110328) to J.B, and by a National Science Foundation Graduate Research Fellowship (NSGRF) to D.B.

Author's contributions
J.V developed the software presented in this study. M.M and M.V developed the proof-of-concept biochemical model. The manuscript was prepared and edited for publication by D.B, J.B. and J.V.

Author details
[1]Department of Chemical and Biomolecular Engineering, Cornell University, 14853 Ithaca NY, USA. [2]Department of Biomedical Engineering, Cornell University, 14853 Ithaca NY, USA.

References
1. Gadkar, K.G., Varner, J., Doyle, F.J.: Model identification of signal transduction networks from data using a state regulator problem. Syst Biol (Stevenage) **2**(1), 17–30 (2005)
2. Gennemark, P., Wedelin, D.: Benchmarks for identification of ordinary differential equations from time series data. Bioinformatics **25**(6), 780–6 (2009). doi:10.1093/bioinformatics/btp050
3. D Battogtokh, M.E.C.J.A.H.-B.s. D K Asch: An ensemble method for identifying regulatory circuits with special reference to the qa gene cluster of neurospora crassa. P Natl Acad Sci USA **99**(26), 16904–16909 (2002)
4. L. Kuepfer, U.S.J.S. M. Peter: Ensemble modeling for analysis of cell signaling dynamics. Nat Biotech **25**(9), 1001–1006 (2007)
5. K S Brown, J.P.S.: Statistical mechanial approaches to models with many poorly known parameters. Phys Rev E **68**, 021904–19 (2003)
6. T. N. Palmer, R.H.F.J.D.-R.T.J.M.L. G. J. Shutts: Representing model uncertainty in weather and climate prediction. Ann Rev Earth and Planetary Sci **33**, 163–193 (2005)
7. Gutenkunst, R.N., Waterfall, J.J., Casey, F.P., Brown, K.S., Myers, C.R., Sethna, J.P.: Universally sloppy parameter sensitivities in systems biology models. PLoS Comput Biol **3**(10), 1871–1878 (2007). doi:10.1371/journal.pcbi.0030189
8. Song, S., Varner, J.: Modeling and Analysis of the Molecular Basis of Pain in Sensory Neurons. PLoS ONE **4**, 6758–6772 (2009)
9. Tasseff, R., Nayak, S., Salim, S., Kaushik, P., Rizvi, N., Varner, J.D.: Analysis of the molecular networks in androgen dependent and independent prostate cancer revealed fragile and robust subsystems. PLoS One **5**(1), 8864 (2010). doi:10.1371/journal.pone.0008864
10. Tasseff, R., Nayak, S., Song, S.O., Yen, A., Varner, J.D.: Modeling and analysis of retinoic acid induced differentiation of uncommitted precursor cells. Integr Biol (Camb) **3**(5), 578–91 (2011). doi:10.1039/c0ib00141d
11. C. G. Moles, J.R.B. P. Mendes: Parameter estimation in biochemical pathways: a comparison of global optimization methods. Genome Res **13**, 2467–2474 (2003)
12. J. E. Purvis, L.F.B.S.L.D. M. S. Chatterjee: A molecular signaling model of platelet phosphoinositide and calcium regulation during homeostasis and p2y 1 activation. Blood **112**(10), 4069–4079 (2008)
13. Song, S.O., Chakrabarti, A., Varner, J.D.: Ensembles of signal transduction models using pareto optimal ensemble techniques (poets). Biotechnol J **5**(7), 768–80 (2010). doi:10.1002/biot.201000059
14. Song, S.O., Varner, J.: Modeling and analysis of the molecular basis of pain in sensory neurons. PLoS One **4**(9), 6758 (2009). doi:10.1371/journal.pone.0006758

15. Lequieu, J., Chakrabarti, A., Nayak, S., Varner, J.D.: Computational modeling and analysis of insulin induced eukaryotic translation initiation. PLoS Comput Biol **7**(11), 1002263 (2011). doi:10.1371/journal.pcbi.1002263
16. `https://www.gnu.org/software/octave/` Accessed May 11, 2016
17. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing (2015). 1411.1607
18. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416–423 (1993)
19. Kim, J., Varner, J., Ramkrishna, D.: A hybrid model of anaerobic e. coli gjt001: Combination of elementary flux modes and cybernetic variables. Biotechnol. Prog. **24**(5), 993–1006 (2008). doi:10.1002/btpr.73
20. Schuster, S., Fell, D.A., Dandekar, T.: A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. Nat Biotechnol **18**(3), 326–32 (2000). doi:10.1038/73786

Figures

Figure 1: Schematic of multiobjective parameter mapping. The performance of any given parameter set is mapped into an objective space using a ranking function which quantifies the quality of the parameters. The distance away from the optimal tradeoff surface is quantified using the Pareto ranking scheme of Fonseca and Fleming in JuPOETs.

Figure 2: The performance of JuPOETs on the multi-objective test suite. The execution time (wall-clock) for JuPOETs and POETs implemented in Octave was measured for 10 independent trials for the suite of test problems. The number of steps per temperature $\mathcal{I} = 10$, and the cooling parameter $\alpha = 0.9$ for all cases. The problem domain was partitioned into 10 equal segments, an initial guess was drawn from each segment. For each of the test functions, JuPOETs estimated solutions on (rank zero solutions, black) or near (gray) the optimal tradeoff surface, subject to bounds and problem constraints.

Figure 3: Representative JuPOETs solutions for problems in the multi-objective test suite. The number of steps per temperature $\mathcal{I} = 10$, and the cooling parameter $\alpha = 0.9$ for all cases. The problem domain was partitioned into 10 equal segments, an initial guess was drawn from each segment. For each of the test functions, JuPOETs estimated solutions on (rank zero solutions, black) or near (gray) the optimal tradeoff surface, subject to bounds and problem constraints.

Figure 4: Proof of concept biochemical network study. Inset: Prototypical biochemical network with six metabolites and seven reactions modeled using the hybrid cybernetic approach (HCM). Intracellular cellmass precursors A, B, and C are balanced (no accumulation) while the extracellular metabolites A_e, B_e, and C_e are dynamic. The oval denotes the cell boundary, q_j is the jth flux across the boundary, and v_k denotes the kth intracellular flux. Four data sets (each with A_e, B_e,C_e and cellmass measurements) were generated by varying the kinetic constant for each biochemical modes. Each data set was a single objective. A: Ensemble simulation of extracellular substrate A_e and cellmass versus time. B: Ensemble simulation of extracellular substrate B_e and C_e versus time. The gray region denotes the 95% confidence estimate of the mean ensemble simulation. The data points denote mean synthetic measurements, while the error bars denote the 95% confidence estimate of the measurement computed over the four training data sets. C: Trade-off plots between the four training objectives. The quantity O_j denotes the jth training objective. Each point represents a member of the parameter ensemble, where gray denotes rank 0 sets, while black denotes rank 1 sets.

Table 1: Multi-objective optimization test problems. We tested the JuPOETs implementation on three two-dimensional test problems, with one-, two- and three-dimensional parameter vectors. Each problem had parameter bounds constraints, however, on the Binh and Korn function had additional non-linear problem constraints. For the Fonesca and Fleming problem, N = 3.

Tables