SOFTWARE

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

David M 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

Background: Ensemble modeling is a promising approach for obtaining robust predictions and coarse grained population behavior in deterministic mathematical models. Ensemble approaches address model uncertainty by using parameter or model families instead of single best-fit parameters or fixed model structures. Parameter ensembles can be selected based upon simulation error, along with other criteria such as diversity or steady-state performance. Simulations using parameter ensembles can estimate confidence intervals on model variables, and robustly constrain model predictions, despite having many poorly constrained parameters.

Results: In this software note, we present a multiobjective based technique to estimate parameter or models ensembles, the Pareto Optimal Ensemble Technique in the Julia programming language (JuPOETs). JuPOETs integrates simulated annealing with Pareto optimality to estimate ensembles on or near the optimal tradeoff surface between competing training objectives. We demonstrate JuPOETs on a suite of multiobjective problems, including test functions with parameter bounds and system constraints as well as for the identification of a proof-of-concept biochemical model with four conflicting training objectives. JuPOETs identified optimal or near optimal solutions approximately six-fold faster than a corresponding implementation in Octave for the suite of test functions. For the proof-of-concept biochemical model, JuPOETs produced an ensemble of parameters that gave both the mean of the training data for conflicting data sets, while simultaneously estimating parameter sets that performed well on each of the individual objective functions.

Conclusions: JuPOETs is a promising approach for the estimation of parameter and model ensembles using multiobjective optimization. JuPOETs can be adapted to solve many problem types, including mixed binary and continuous variable types, bilevel optimization problems and constrained problems without altering the base algorithm. JuPOETs is open source, available under an MIT license, and can be installed using the Julia package manager from the JuPOETs GitHub repository

Keywords: Ensemble modeling; Multiobjective optimization; Julia

Background

Ensemble modeling is a promising approach for obtaining robust predictions and coarse grained population behavior in deterministic mathematical models. It is of-

ten not possible to uniquely identify all the parameters in biochemical models, 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 address parameter uncertainty in systems biology and other fields like weather prediction [3–6] by using parameter families instead of single best-fit parameter sets. Parameter families can be selected based upon simulation error, along with other criteria such as diversity or steady-state performance. 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 many 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 [3, 5, 9, 10]. Liao and coworkers developed methods to generate ensembles that all approach the same steady-state, for example one determined by fluxomics measurements [11]. They have used this approach for model reduction [12], strain engineering [13, 14] and to study the robustness of non-native pathways and network failure [15]. Maranas and coworkers have also applied this method to develop a comprehensive kinetic model of bacterial central carbon metabolism, including mutant data [16]. We and others have used ensemble approaches, generated using both sampling and optimization techniques, that have robustly simulated a wide variety of signal transduction processes [9, 10, 17–19], neutrophil trafficking in sepsis [20], patient specific coagulation behavior [21], uncertainty quantification in metabolic kinetic models [22] and to capture cell to cell variation [23]. Further, ensemble approaches have been used in synthetic biology to sample possible biocircuit configurations [24]. Thus, ensemble approaches are widely used to robustly simulate a variety of biochemical systems.

Identification of biochemical models requires significant training data perhaps taken from diverse sources. These real-world data sets often contain intrinsic conflicts resulting from, for example, the use of different cell lines, different measurement technologies, different reagent vendors or lots, uncontrollable experimental artifacts or general cross laboratory variability. Parameter ensembles that optimally balance these inherent conflicts lead to more robust model performance. Multiobjective optimization is an ensemble generation technique that naturally balances conflicts in noisy training data [25]. Multiobjective optimization has been used to identify signal transduction models [18, 23], for the design of synthetic circuits [24], to design the folding behaviors of novel RNAs [26], to design bioprocesses [27], and to understand bacterial adaptation [28]. Thus, it is a widely used approach for a variety of biochemical applications. Previously, we developed the Pareto Optimal Ensemble Technique (POETs) algorithm to address the challenge of competing or conflicting training objectives. POETs, which integrates simulated annealing (SA) and multiobjective optimization through the notion of Pareto rank, estimates parameter ensembles which optimally trade-off between competing (and potentially conflicting) experimental objectives [29]. However, the previous implementation of POETs, in the Octave programming language [30], suffered from poor performance and was not configurable. For example, Octave-POETs does not accommodate user definable objective functions, bounds and problem constraints, cooling schedules,

different variable types e.g., a mixture of binary and continuous design variables or custom diversity generation routines. Octave-POETs was also not well integrated into a package or source code management (SCM) system. Thus, upgrades to the approach containing new features, or bug fixes were not centrally managed.

Implementation

In this software note, we present an open-source implementation of the Pareto optimal ensemble technique in the Julia programming language (JuPOETs). JuPOETs takes advantage of the unique features of Julia to address many of the shortcomings of the previous implementation. Julia is a cross-platform, high-performance programming language for technical computing that has performance comparable to C but with syntax similar to MATLAB/Octave and Python [31]. Julia also offers a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive function library. Further, 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 diversity generation logic, linear/non-linear parameter constraints (and parameter bounds constraints) as well as custom cooling schedules. Julia's ability to naturally call other languages such as Python or C also allows JuPOETs to be used with models implemented in a variety of languages across many platforms. Additionally, Julia offers a built-in package manager which is directly integrated with GitHub, a popular web-based Git repository hosting service offering distributed revision control and source code management. Thus, JuPOETs can be adapted to many problem types, including mixed binary and continuous variable types, bilevel problems and constrained problems without altering the base algorithm, as was required in 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 the model equations and constraints:

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

and parameter bound constraints:

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

The quantity O_j denotes the j^{th} objective function ($j = 1, 2, \ldots, \mathcal{K}$), typically the sum of squared errors for the j^{th} data set for biochemical modeling applications. 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 decision variable vector e.g., unknown model parameters ($\mathcal{D} \times \ 1$). In typical biochemical modeling applications, the model equations $\mathbf{f}(\cdot)$ are a system of continuous real-valued non-linear differential equations that comprise a kinetic model, but other types of models e.g., stoichiometric models are also common. The quantity t denotes time, $\mathbf{x}(t, \mathbf{p})$ denotes the model state (with an initial state $\mathbf{x}_0$), and $\mathbf{u}(t)$ denotes an input vector. The decision variables (e.g., kinetic parameters) can be subject to bounds constraints, where $\mathcal{L}$ and $\mathcal{U}$ denote the lower and upper bounds, respectively as well as $\mathcal{C}$ problem specific constraints $g_i(t, \mathbf{x}(t, \mathbf{p}), \mathbf{u}(t), \mathbf{p}), i = 1, \ldots, \mathcal{C}$. The decision variables $\mathbf{p}$ are typically real-valued kinetic constants, or metabolic fluxes in the case of stoichiometric models. However, other variables types e.g., binary or categorical decision variables can also be accommodated.

JuPOETs integrates simulated annealing (SA) [32] with Pareto ranking to estimate parameter sets on or near the optimal tradeoff surface between competing objectives (Fig. 1 and Algorithm 1). A tradeoff surface defines the best possible performance for every conflicting objective, such that an increase in the performance of one objective does not decrease the performance of at least one other objective. Pareto rank is a measure of distance away from the optimal tradeoff surface. Thus, the central idea underlying 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 (Fig. 1). A parameter set $\mathbf{p}_{i+1}$ lies along the optimal tradeoff surface if no other parameter guess leads to decreased error for every objective. 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, evaluates the model equations, and using this solution, returns the $\mathcal{K} \times 1$ objective vector. Candidate parameter sets are generated by the user supplied `neighbor` function; the default implementation of `neighbor` is a random perturbation, however other perturbation logic can be implemented by the user. The error vector associated with $\mathbf{p}_{i+1}$ is ranked using the builtin Pareto `rank` function, by comparing the error at iteration $i + 1$ to the error archive $\mathcal{O}_i$ (all error vectors up to iteration i meeting a ranking criterion). 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). These rank zero parameter sets define the Pareto optimal group for the ensemble, wherein Pareto optimality is defined as a parameter set not being dominated by any other sets within the ensemble. 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 in the builtin `rank` function [33]:

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

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

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

where T is the computational annealing temperature that provides control over how strictly Pareto rank is enforced. As $\texttt{rank}\left(\mathcal{O}_{i+1}(\mathbf{p}_{i+1}) \mid \mathcal{O}_i\right) \to 0$, 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 is 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 decreases. Parameter sets could also be accepted by the SA but *not* permanently archived in $\mathcal{S}_i$, where $\mathcal{S}_i$ is the solution archive. Only parameter sets with rank less than or equal to a threshold (rank $\leq$4 by default) are included in $\mathcal{S}_i$, where the archive is re-ranked and filtered after accepting every new parameter set. 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.

To use JuPOETs, the user specifies the `neighbor`, `acceptance`, `cooling` and `objective` functions along with an initial decision variable guess. If the user is operating JuPOETs in hybrid mode, then a `refinement` function pointer must also be specified; the default `refinement` implementation is simply a pass through function. Default implementations of the `neighbor`, `acceptance` and `cooling` functions can be used directly, or they can be overridden by user defined logic. However, the user must provide an implementation of the `objective` function and provide an initial decision variable guess. In addition, there are several user configurable parameters that can be adjusted to control the performance of JuPOETs: `maximum_number_of_iterations` controls the number of iterations per temperature (default 20); `rank_cutoff` controls the upper rank bound on the solution archive (default 5); `temperature_min` controls the minimum temperature after which JuPOETs returns the error and solution archives (default 0.001); `show_trace` controls the level of output shown to the user (default true). After the completion of the run, JuPOETs returns the parameter solution archive $\mathcal{S}$, objective archive $\mathcal{O}$ and rank archive $\mathcal{R}$. The parameter solution archive $\mathcal{S}$ contains is an $\mathcal{D} \times \mathcal{A}$ array, where $\mathcal{A}$ denotes the number of solutions in the archive when JuPOETs terminated. On the other hand, the objective archive $\mathcal{O}$ is an $\mathcal{K} \times \mathcal{A}$ array containing the performance values for each objective corresponding the columns of

$\mathcal{S}$. Lastly, JuPOETs returns the rank archive $\mathcal{R}$ which is an $\mathcal{A} \times 1$ array of Pareto ranks corresponding to the columns of $\mathcal{S}$.

Availability of data and materials

JuPOETs is open source, available under an MIT software license. The JuPOETs source code is freely available from the JuPOETs GitHub repository at https://github.com/varnerlab/POETs.jl. All samples used in this study are included in the `sample/biochemical` and `sample/test_functions` subdirectories of the JuPOETs GitHub repository.

input : User specified objective function, and initial guess ($\mathcal{D} \times 1$). User can also specify custom neighbor, acceptance. cooling and refinement functions or use the default functions provided.

Output: Rank archive $\mathcal{R}$ ($\mathcal{A} \times 1$), parameter solution archive $\mathcal{S}$ ($\mathcal{D} \times \mathcal{A}$) and objective archive $\mathcal{O}$ ($\mathcal{K} \times \mathcal{A}$), where $\mathcal{A}$ denotes the number of accepted solutions

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

```
// Call to local refinement function (single objective problem)
```
$\mathbf{p}_o \leftarrow$ user-function:`refinement`$(\mathbf{p}_o)$;

while $T > T_{min}$ **do**
  i $\leftarrow$ 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 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 O using builtin rank function
```
    $\mathcal{R} \leftarrow$ builtin-function::`rank`$(\mathcal{O})$;

```
// Accept 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 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 $\leftarrow$ i + 1;
  end

```
// Update T using the user cooling function
```
  $T \leftarrow$ user-function::cooling(T);
end

Algorithm 1: Pseudo-code for the JuPOETs run-loop. The user must specify the `objective` function and an initial parameter guess. The user can optionally specify the `neighbor`, `acceptance`, `cooling` and `refinement` functions (or use the default implementations). The rank archive $\mathcal{R}$, solution archive $\mathcal{S}$ and objective archive $\mathcal{O}$ are initialized from the initial guess. The initial guess (potentially following a single objective local refinement step) 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 temperature is adjusted using the user supplied `cooling` function after each $\mathcal{I}$ iterations. When JuPOETs terminates, the parameter solution archive $\mathcal{S}$, objective archive $\mathcal{O}$ and rank archive $\mathcal{R}$ are retuned to the caller.

Results and Discussion

JuPOETs identified optimal or nearly optimal solutions significantly faster than Octave-POETs for a suite of multiobjective algebraic test problems (Table 1). The algebraic test problems were constrained non-linear functions with bound constraints and additional non-linear constraints on the decision variables in one case. The problems had up to three-dimensional continuous real-valued decision vectors, and each case had two objective functions. 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 decision variable bound and problem constraints (Fig. 3). Thus, JuPOETs estimated an ensemble of solutions to constrained multiobjective algebraic 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 model parameters that were 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 extracellular measurements of A_e, B_e, C_e and cellmass 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) approach of Ramkrishna and coworkers [34]. In the HCM approach, metabolic networks are first decomposed into a set of elementary modes (EMs) (chemically balanced steady-state pathways, see [35]). 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 (continuos real-valued decision variables). The synthetic training data was generated by randomly varying these parameters.

The general form of the biochemical test problem was given by:

$$\min_{\mathbf{p}} (O_1, \ldots, O_{\mathcal{K}}) \tag{5}$$

subject to model and bounds constraints. We considered four training data sets ($\mathcal{K} = 4$), each of which contained time-series measurements of A_e, B_e, C_e and cellmass. Each objective O_j, $j = 1, \ldots, \mathcal{K}$ quantified the squared difference between the simulated (x_i) and measured extracellular species abundance (y_i) in the j^{th} data set:

$$O_j = \sum_i \sum_\tau (x_i(\tau) - y_i(\tau))^2 \qquad j = 1, \ldots, \mathcal{K} \tag{6}$$

where, i denotes the species index and τ denotes the time index. The abundance of extracellular species i (x_i), the pseudo enzyme e_l (catalyzes flux through mode l), and cellmass were governed by the model equations:

$$\begin{aligned}
\frac{dx_i}{dt} &= \sum_{j=1}^{\mathcal{R}} \sum_{l=1}^{\mathcal{L}} \sigma_{ij} z_{jl} q_l(\mathbf{e}, \mathbf{p}, \mathbf{x})\, c \qquad i = 1, \ldots, \mathcal{M} \\
\frac{de_l}{dt} &= \alpha_l + r_{El}(\mathbf{p}, \mathbf{x})\, u_l - (\beta_l + r_G)\, e_l \qquad l = 1, \ldots, \mathcal{L} \\
\frac{dc}{dt} &= r_G c
\end{aligned}$$

where $\mathcal{R}$ and $\mathcal{M}$ denote the number of reactions and extracellular species in the model and $\mathcal{L}$ denotes the number of elementary modes. The quantity σ_{ij} denotes the stoichiometric coefficient for species i in reaction j and z_{jl} denotes the normalized flux for reaction j in mode l. If $\sigma_{ij} > 0$, species i is produced by reaction j; if $\sigma_{ij} < 0$, species i is consumed by reaction j; if $\sigma_{ij} = 0$, species i is not connected with reaction j. Extracellular species, cellmass and pseudo-enzyme were subject to the initial conditions $\mathbf{x}(t_o) = \mathbf{x}_o$, $c(t_o) = c_o$ and $e_l = 0.5$, respectively. The term $q_l(\mathbf{e}, \mathbf{p}, \mathbf{x})$ denotes the specific uptake/secretion rate for mode l where $\mathbf{e}$ denotes the pseudo enzyme vector, $\mathbf{p}$ denotes the unknown kinetic parameter vector (decision variables), $\mathbf{x}$ denotes the extracellular species vector, and c denotes the cell mass; $q_l(\mathbf{e}, \mathbf{p}, \mathbf{x})$ is the product of a kinetic term ($\bar{q}_l$) and a control variable governing enzyme activity. Flux through each mode was catalyzed by a pseudo enzyme e_l, synthesized at the regulated specific rate $r_{E,l}(\mathbf{p}, \mathbf{x})$, and constitutively at the rate α_l. The term u_l denotes the cybernetic variable controlling the synthesis of enzyme l. The term β_l denotes the rate constant governing non-specific enzyme degradation, and r_G denotes the specific growth rate through all modes. The specific uptake/secretion rates and the specific rate of enzyme synthesis were modeled using saturation kinetics. The specific growth rate was given by:

$$r_G = \sum_{l=1}^{\mathcal{L}} z_{\mu l} q_l(\mathbf{e}, \mathbf{p}, \mathbf{x})$$

where $z_{\mu l}$ denotes the growth flux μ through mode l. The control variables u_l and v_l, which control the synthesis and activity of each enzyme respectively, were given by:

$$u_l = \frac{z_{sl}\bar{q}_l}{\sum_{l=1}^{\mathcal{L}} z_{sl}\bar{q}_l} \tag{7}$$

and

$$v_l = \frac{z_{sl}\bar{q}_l}{\max_{l=1,\ldots,\mathcal{L}} z_{sl}\bar{q}_l} \tag{8}$$

where z_{sl} denotes the uptake flux of substrate s through mode l. Each unknown kinetic parameter was continuous and real-valued, and subject to bounds constraints: $\mathcal{L} \leq \mathbf{p} \leq \mathcal{U}$.

JuPOETs produced an ensemble of approximately $\dim \mathcal{S} \simeq 13{,}000$ parameter sets that captured the mean of the measured data sets for extracellular metabolites and cellmass (Fig. 4A and B). JuPOETs minimized the difference between the simulated and measured values for extracellular metabolites A_e, B_e, C_e and cellmass, where the residual for each data set was treated as a single objective (leading to four objectives). The 95% confidence estimate produced by the ensemble was consistent with the 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). Multiple trade-off fronts were visible in the objective plots, for example between data set 3 (O_3) and data set 2 (O_2). Thus, without a multiobjective approach, it would be challenging to capture these data sets as fitting one leads to decreased performance on the other. However, the ensemble contained parameter sets that described each data set independently (Fig. 5). Thus, JuPOETs produced an ensemble of parameters that gave the mean of the training data for conflicting data sets, while simultaneously estimating parameter sets that performed well on each individual objective function.

Conclusions

In this software note, we presented JuPOETs, a multiobjective technique to estimate parameter ensembles in the Julia programming language. JuPOETs is open source, and available for download under an MIT license from the JuPOETs GitHub repository at https://github.com/varnerlab/POETs.jl. We demonstrated JuPOETs on a suite of algebraic test problems, and a proof-of-concept ODE based biochemical model. While JuPOETs outperformed (and was significantly more flexible) than the previous Octave implementation, 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 non-dominated sorting genetic algorithm (NSGA) family of algorithms, have been adapted to solve multiobjective problems [36, 37]. However, since there is a lack of open source Julia implementations of these alternative approaches, we did not benchmark the relative performance of JuPOETs in this note. One advantage that JuPOETs may have when compared to a strictly evolutionary approaches, is the inclusion of a local refinement step (hybrid mode), which temporarily reduces the problem to a single objective formulation. Previously, POETs run in hybrid mode led to better convergence on a proof-of-concept signal transduction model compared to the same approach without the hybrid refinement step [29]. Other hybrid multiobjective methods have also been shown to be more efficient than evolutionary approaches alone, for a variety of biochemical optimization problems [24, 38]. Thus, there are several different algorithms that we can use to benchmark, and improve the performance of JuPOETs, after we implement them in Julia. Another strategy to improve the performance of JuPOETs is to reduce the number (or cost) of function evaluations that are required to obtain optimal or near optimal solutions. For example, in many real world parameter estimation problems, the bulk of the execution time is spent evaluating the objective functions. One strategy to improve JuPOETs performance could be to optimize surrogates [39], while another would be parallel execution of the objective functions.

Currently, JuPOETs serially evaluates the objective function vector. However, parallel evaluation of the objective functions e.g., using the `parallel` Julia macro or other techniques, could be implemented without significantly changing the JuPOETs run loop. Taken together, JuPOETs demonstrated improved flexibility, and performance over POETs in parameter identification and ensemble generation for multiple objectives. JuPOETs has the potential for widespread use due to the flexibility of the implementation, and the high level syntax and distribution tools native to the Julia programming language.

Ethics approval and consent to participate
Not applicable

Consent to publish
Not applicable

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 (DGE-1144153) to D.B. Lastly, J.V was supported by an award from the US Army and Systems Biology of Trauma Induced Coagulopathy (W911NF-10-1-0376).

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.

Acknowledgements
We gratefully acknowledge Ani Chakrabarti, Russell Gould and Kathy Rogers for their input and suggestions regarding new features to include into JuPOETs. We also gratefully acknowledge the suggestions from the anonymous reviewers to improve this manuscript and JuPOETs.

Availability of data and materials
JuPOETs is open source, available under an MIT software license. The JuPOETs source code is freely available from the JuPOETs GitHub repository at https://github.com/varnerlab/POETs.jl. All samples used in this study are included in the `sample/biochemical` and `sample/test_functions` subdirectories of the JuPOETs GitHub repository.

List of abbreviations
Not applicable

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. Battogtokh, D., Asch, D.K., Case, M.E., Arnold, J., Shüttler, H.B.: An ensemble method for identifying regulatory circuits with special reference to the qa gene cluster of Neurospora crassa. Proc Natl Acad Sci U S A **99**(26), 16904–16909 (2002)
4. Kuepfer, L., Peter, M., Sauer, U., Stelling, J.: Ensemble modeling for analysis of cell signaling dynamics. Nat Biotechnol **25**(9), 1001–6 (2007). doi:10.1038/nbt1330
5. Brown, K.S., Sethna, J.P.: Statistical mechanial approaches to models with many poorly known parameters. Phys Rev E **68**, 021904–19 (2003)
6. Palmer, T.N., Shutts, G.J., Hagedorn, R., Doblas-Reyes, F.J., Jung, T., Leutbecher, M.: 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. Tran, L.M., Rizk, M.L., Liao, J.C.: Ensemble modeling of metabolic networks. Biophys J **95**(12), 5606–17 (2008). doi:10.1529/biophysj.108.135442
12. Tan, Y., Rivera, J.G.L., Contador, C.A., Asenjo, J.A., Liao, J.C.: Reducing the allowable kinetic space by constructing ensemble of dynamic models with the same steady-state flux. Metab Eng **13**(1), 60–75 (2011). doi:10.1016/j.ymben.2010.11.001
13. Contador, C.A., Rizk, M.L., Asenjo, J.A., Liao, J.C.: Ensemble modeling for strain development of l-lysine-producing escherichia coli. Metabolic Engineering **11**(4–5), 221–233 (2009). doi:10.1016/j.ymben.2009.04.002
14. Tan, Y., Liao, J.C.: Metabolic ensemble modeling for strain engineers. Biotechnol J **7**(3), 343–53 (2012). doi:10.1002/biot.201100186
15. Lee, Y., Lafontaine Rivera, J.G., Liao, J.C.: Ensemble modeling for robustness analysis in engineering non-native metabolic pathways. Metab Eng **25**, 63–71 (2014). doi:10.1016/j.ymben.2014.06.006
16. Khodayari, A., Zomorrodi, A.R., Liao, J.C., Maranas, C.D.: A kinetic model of escherichia coli core metabolism satisfying multiple sets of mutant flux data. Metab Eng **25**, 50–62 (2014). doi:10.1016/j.ymben.2014.05.014
17. Luan, D., Zai, M., Varner, J.D.: Computationally derived points of fragility of a human cascade are consistent with current therapeutic strategies. PLoS Comput Biol **3**(7), 142 (2007). doi:10.1371/journal.pcbi.0030142
18. 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
19. Nayak, S., Siddiqui, J.K., Varner, J.D.: Modelling and analysis of an ensemble of eukaryotic translation initiation models. IET Syst Biol **5**(1), 2 (2011). doi:10.1049/iet-syb.2009.0065
20. Song, S.O., Song, S.O.K., Hogg, J., Peng, Z.-Y., Parker, R., Kellum, J.A., Clermont, G.: Ensemble models of neutrophil trafficking in severe sepsis. PLoS Comput Biol **8**(3), 1002422 (2012). doi:10.1371/journal.pcbi.1002422
21. Luan, D., Szlam, F., Tanaka, K.A., Barie, P.S., Varner, J.D.: Ensembles of uncertain mathematical models can identify network response to therapeutic interventions. Mol Biosyst **6**(11), 2272–86 (2010). doi:10.1039/b920693k
22. Andreozzi, S., Miskovic, L., Hatzimanikatis, V.: iSCHRUNK–in silico approach to characterization and reduction of uncertainty in the kinetic models of genome-scale metabolic networks. Metab Eng **33**, 158–68 (2016). doi:10.1016/j.ymben.2015.10.002
23. 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
24. Otero-Muras, I., Banga, J.R.: Multicriteria global optimization for biocircuit design. BMC Syst Biol **8**, 113 (2014). doi:10.1186/s12918-014-0113-3
25. Handl, J., Kell, D.B., Knowles, J.: Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Trans Comput Biol Bioinform **4**(2), 279–92 (2007). doi:10.1109/TCBB.2007.070203
26. Taneda, A.: Multi-objective optimization for RNA design with multiple target secondary structures. BMC bioinformatics **16**(1), 280 (2015). doi:10.1186/s12859-015-0706-x
27. Sendın, J., Otero-Muras, I., Alonso, A.A., Banga, J.: Improved Optimization Methods for the Multiobjective Design of Bioprocesses. Ind. Eng. Chem. Res. **45**, 8594–8603 (2006)
28. Angione, C., Lió, P.: Predictive analytics of environmental adaptability in multi-omic network models. Sci Rep **5**, 15147 (2015). doi:10.1038/srep15147
29. 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
30. Eaton, J.W., Bateman, D., Hauberg, S.: GNU Octave Version 3.0.1 Manual: a High-level Interactive Language for Numerical Computations. CreateSpace Independent Publishing Platform, North Charleston, SC, USA (2009)
31. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. CoRR **abs/1411.1607** (2014)
32. Kirkpatrick, S., Gelatt, C.D. Jr, Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–80 (1983). doi:10.1126/science.220.4598.671
33. 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)
34. 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
35. 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
36. Kalyanmoy, D., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. Evol. Comp. **6**, 182–197 (2002)
37. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Trans. Evol. Comp. **10**, 477–506 (2006)
38. Sendin, J.O.H., Otero-Muras, I., Alonso, A.A., Banga, J.R.: Improved optimization methods for multiple objective design of bioprocesses. Industrial and Engineering Chemistry Research **45**, 8594–8603 (2006)
39. Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Struct Optim **17**, 1–13 (1999)

Figures

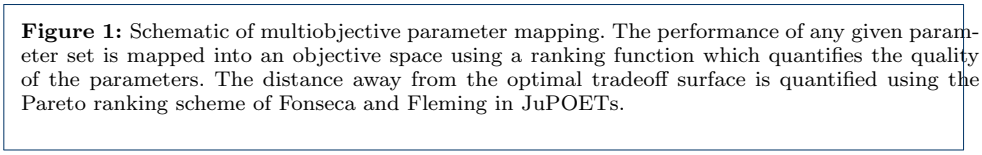

Figure 1: Schematic of multiobjective parameter mapping. The performance of any given parameter set is mapped into an objective space using a ranking function which quantifies the quality of the parameters. The distance away from the optimal tradeoff surface is quantified using the Pareto ranking scheme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

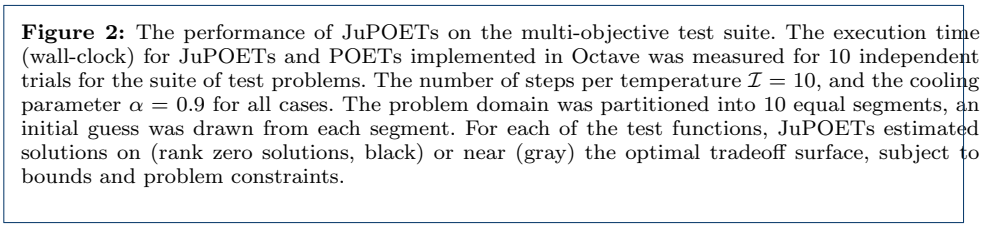

Figure 2: The performance of JuPOETs on the multi-objective test suite. The execution time (wall-clock) for JuPOETs and POETs implemented in Octave was measured for 10 independent trials for the suite of test problems. The number of steps per temperature $\mathcal{I} = 10$, and the cooling parameter $\alpha = 0.9$ 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.

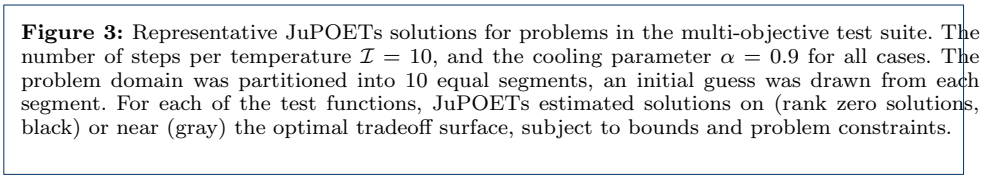

Figure 3: Representative JuPOETs solutions for problems in the multi-objective test suite. The number of steps per temperature $\mathcal{I} = 10$, and the cooling parameter $\alpha = 0.9$ for all cases. The problem domain was partitioned into 10 equal segments, an initial guess was drawn from each segment. 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 right: 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 constants for each biochemical mode. Each data set was a single objective in the JuPOETs procedure. 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. Ensembles were generated using POETs without employing local refinement.

Figure 5: Experiment to experiment variation captured by the ensemble. Cellmass measurements (points) versus time for experiment 2 and 3 were compared with ensemble simulations. The full ensemble was sorted by simultaneously selecting the top 25% of solutions for each objective with rank ≤ 1. The best fit solution for each objective (line) $\pm$ 1-standard deviation (gray region) for experiment 2 and 3 brackets the training data despite significant differences the training values between the two data 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