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

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

School of Chemical and Biomolecular Engineering and

[1]School of Biomedical Engineering, Cornell University, Ithaca NY 14853

Running Title: Multiobjective model ensemble generation

To be submitted: *BMC Systems Biology*

*Corresponding author:

Jeffrey D. Varner,

Professor, School of Chemical and Biomolecular Engineering,

244 Olin Hall, Cornell University, Ithaca NY, 14853

Email: jdv27@cornell.edu

Phone: (607) 255 - 4258

Fax: (607) 255 - 9166

Abstract

Ensemble modeling is a well established approach for obtaining robust predictions and course grained population behavior in large deterministic mathematical models. In this study, we present the Pareto Optimal Ensemble Technique in the Julia programming language (JuPOETs). JuPOETs integrates simulated annealing with Pareto optimality to estimate an ensemble of parameter sets on or near the optimal tradeoff surface between competing training objectives. We demonstrated JuPOETs on a suite of multiobjective problems, including test functions with parameter bounds and system constraints as well as identification of a proof-of-concept biochemical model. JuPOETs identified optimal or near optimal solutions approximately six-fold faster than a corresponding implementation in Octave. JuPOETs can be installed using the Julia package manager from the JuPOETs GitHub repository at https://github.com/varnerlab/POETs.jl.

Background

Ensemble modeling is a well established approach for obtaining robust predictions and course grained population behavior in large deterministic mathematical models. It is often 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 criterion such as diversity or steady-state peformance. 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 that 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, to robustly simulate a wide variety of signal transduction processes [9, 10, 17–19], neutrophil trafficking in sepsis [20], patient specific coagulation behavior [21] and to capture cell to cell variation [22]. Thus, ensemble approaches are widely used to robustly simulate a variety of biochemical systems.

Identification of biochemical models with hundreds or even thousands of states and parameters may not be tractable as a single objective optimization problem. Further, large models require significant training data perhaps taken from diverse sources, for example different laboratories or cell-lines. These data are often heterogenous, and contain intrinsic conflicts that complicate parameter estimation. Parameter ensembles which optimally balance tradeoffs between submodels and conflicts in training data can lead to robust model performance. Multiobjective optimization is an ensemble generation technique that naturally balances conflicting training data. 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, estimates parameter ensembles which optimally trade-off between competing (and potentially conflicting) experimental objectives [23]. However, the previous implementation of POETs, in the Octave programming language [24], 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 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 compared to Octave-POETs. JuPOETs takes advantage of the unique features of Julia. 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 [25]. 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:

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

and parameter bound constraints:

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

using a modified simulated annealing approach. The quantity t denotes time, $\mathbf{x}(t,\mathbf{p})$ 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 $g_i(t, \mathbf{x}(t,\mathbf{p}), \mathbf{u}(t), \mathbf{p}), i = 1, \ldots, \mathcal{C}$.

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 $\mathcal{O}_i$ (all error vectors up to iteration i-1 that meeting a ranking criteria). 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 [26] 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 $\texttt{rank}\left(\mathbf{p}_{i+1} \mid \mathcal{O}_i\right) \to 0$, 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 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 (λ = 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. Sample code is included in the `sample/biochemical` subdirectory of the JuPOETs repository to help users get started using JuPOETs in their projects.

Results and Discussion

JuPOETs identified optimal or nearly optimal solutions significantly faster than Octave-POETs for a suite of multiobjective test problems (Table **??**). 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 α = 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 [27]. In the HCM-EM approach, metabolic networks are first decomposed into a set of elementary modes (EMs) (chemically balanced steady-state pathways, see [28]). 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 approximately $\dim \mathcal{S} \simeq$ 13,000 parameters 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 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

JuPOETs is a significant advance over the previous POETs implementation. It offers improved performance and is highly adaptable to different problem types. We demonstrated JuPOETs on a suite of test problems, and a proof-of-concept biochemical model. However, there are several areas that could be explored further to improve JuPOETs. 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 algo-

rithms, have been adapted to solve multiobjective optimization problems [29, 30]. 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 [23]. 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 [31], 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 could be easily implemented using a variety of techniques without changing the main run loop of JuPOETs. Because of the flexible function pointer architecture of JuPOETs, the only changes required are in the user defined objective function. Taken together, JuPOETs has 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 Julia.

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.

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.

References

1. Gadkar KG, Varner J, Doyle FJ (2005) Model identification of signal transduction networks from data using a state regulator problem. Syst Biol (Stevenage) 2: 17–30.
2. Gennemark P, Wedelin D (2009) Benchmarks for identification of ordinary differential equations from time series data. Bioinformatics 25: 780-6.
3. Battogtokh D, Asch D, Case M, Arnold J, Shüttler H (2002) 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: 16904-16909.
4. L Kuepfer USJS M Peter (2007) Ensemble modeling for analysis of cell signaling dynamics. Nat Biotech 25: 1001-1006.
5. K S Brown JPS (2003) Statistical mechanial approaches to models with many poorly known parameters. Phys Rev E 68: 021904:1-9.
6. Palmer T, Shutts G, Hagedorn R, Doblas-Reyes F, Jung T, et al. (2005) Representing

model uncertainty in weather and climate prediction. Ann Rev Earth and Planetary Sci 33: 163-193.

7. Gutenkunst RN, Waterfall JJ, Casey FP, Brown KS, Myers CR, et al. (2007) Universally sloppy parameter sensitivities in systems biology models. PLoS Comput Biol 3: 1871–1878.
8. Song S, Varner J (2009) Modeling and Analysis of the Molecular Basis of Pain in Sensory Neurons. PLoS ONE 4: e6758 - e6772.
9. Tasseff R, Nayak S, Salim S, Kaushik P, Rizvi N, et al. (2010) Analysis of the molecular networks in androgen dependent and independent prostate cancer revealed fragile and robust subsystems. PLoS One 5: e8864.
10. Tasseff R, Nayak S, Song SO, Yen A, Varner JD (2011) Modeling and analysis of retinoic acid induced differentiation of uncommitted precursor cells. Integr Biol (Camb) 3: 578-91.
11. Tran LM, Rizk ML, Liao JC (2008) Ensemble modeling of metabolic networks. Biophys J 95: 5606-17.
12. Tan Y, Rivera JGL, Contador CA, Asenjo JA, Liao JC (2011) Reducing the allowable kinetic space by constructing ensemble of dynamic models with the same steady-state flux. Metab Eng 13: 60-75.
13. Contador CA, Rizk ML, Asenjo JA, Liao JC (2009) Ensemble modeling for strain development of l-lysine-producing escherichia coli. Metabolic Engineering 11: 221 - 233.
14. Tan Y, Liao JC (2012) Metabolic ensemble modeling for strain engineers. Biotechnol J 7: 343-53.
15. Lee Y, Lafontaine Rivera JG, Liao JC (2014) Ensemble modeling for robustness analysis in engineering non-native metabolic pathways. Metab Eng 25: 63-71.
16. Khodayari A, Zomorrodi AR, Liao JC, Maranas CD (2014) A kinetic model of es-

cherichia coli core metabolism satisfying multiple sets of mutant flux data. Metab Eng 25: 50-62.

17. Luan D, Zai M, Varner JD (2007) Computationally derived points of fragility of a human cascade are consistent with current therapeutic strategies. PLoS Comput Biol 3: e142.
18. Song SO, Varner J (2009) Modeling and analysis of the molecular basis of pain in sensory neurons. PLoS One 4: e6758.
19. Nayak S, Siddiqui JK, Varner JD (2011) Modelling and analysis of an ensemble of eukaryotic translation initiation models. IET Syst Biol 5: 2.
20. Song SO, Song SOK, Hogg J, Peng ZY, Parker R, et al. (2012) Ensemble models of neutrophil trafficking in severe sepsis. PLoS Comput Biol 8: e1002422.
21. Luan D, Szlam F, Tanaka KA, Barie PS, Varner JD (2010) Ensembles of uncertain mathematical models can identify network response to therapeutic interventions. Mol Biosyst 6: 2272-86.
22. Lequieu J, Chakrabarti A, Nayak S, Varner JD (2011) Computational modeling and analysis of insulin induced eukaryotic translation initiation. PLoS Comput Biol 7: e1002263.
23. Song SO, Chakrabarti A, Varner JD (2010) Ensembles of signal transduction models using pareto optimal ensemble techniques (poets). Biotechnol J 5: 768-80.
24. Eaton JW, Bateman D, Hauberg S (2009) GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations. North Charleston, SC, USA: CreateSpace Independent Publishing Platform.
25. Bezanson J, Edelman A, Karpinski S, Shah VB (2015) Julia: A fresh approach to numerical computing .
26. Fonseca C, Fleming PJ (1993) Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Proceedings of the 5th International

Conference on Genetic Algorithms. pp. 416 - 423.

27. Kim J, Varner J, Ramkrishna D (2008) A hybrid model of anaerobic e. coli gjt001: Combination of elementary flux modes and cybernetic variables. Biotechnol Prog 24: 993–1006.
28. Schuster S, Fell DA, Dandekar T (2000) A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. Nat Biotechnol 18: 326-32.
29. Kalyanmoy D, Pratap A, Agarwal S, Meyarivan T (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans Evol Comp 6: 182 - 197.
30. Huband S, Hingston P, Barone L, While L (2006) A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Trans Evol Comp 10: 477 - 506.
31. Booker A, Dennis J, Frank P, Serafini D, Torczon V, et al. (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct Optim 17: 1 - 13.

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

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

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` $\mathbf{p}_{i+1}$ `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 $\leftarrow$ 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.

Figures

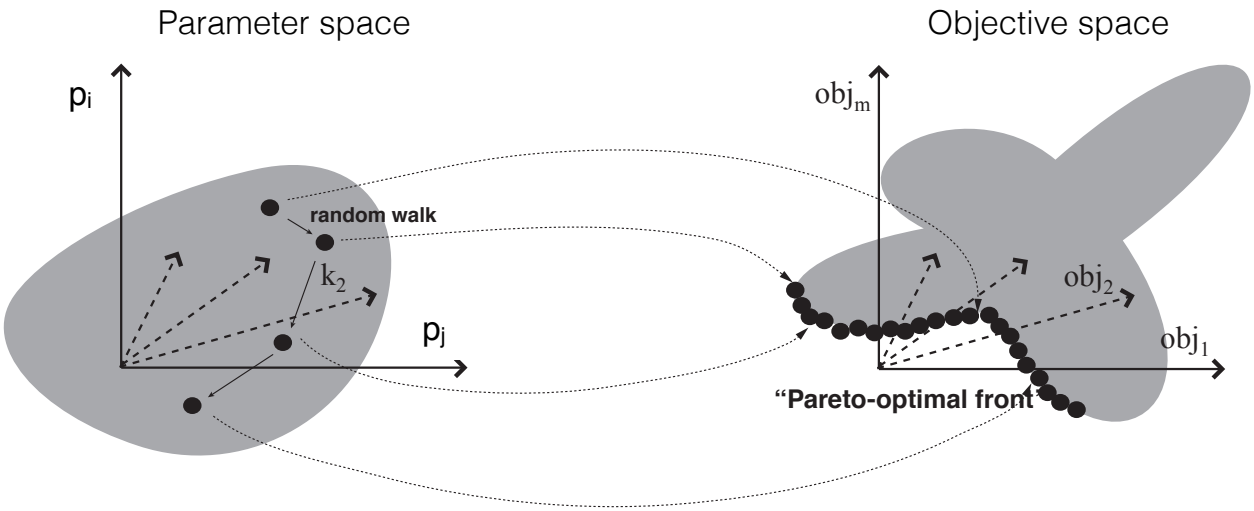

Fig. 1: Schematic of multiobjective parameter mapping. The performance of any given parameter set is mapped into an objective space using a ranking function which quantifies the quality of the parameters. The distance away from the optimal tradeoff surface is quantified using the Pareto ranking scheme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

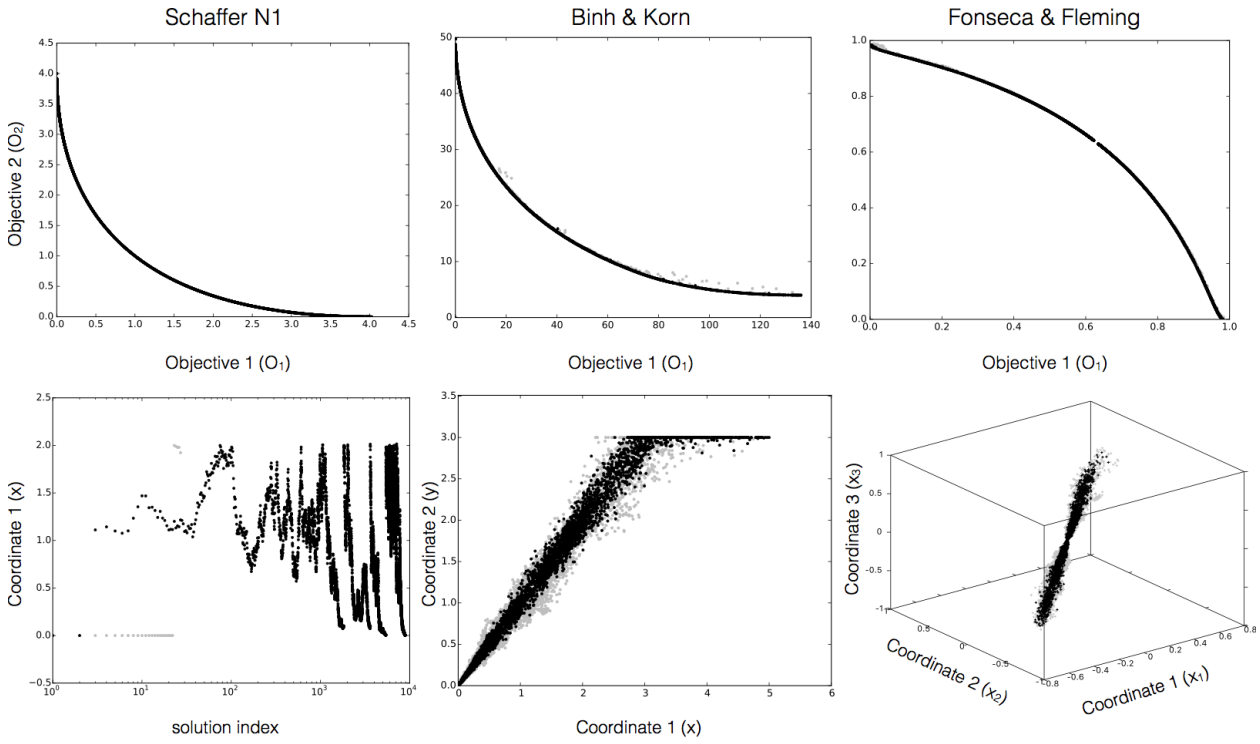

Fig. 2: The performance of JuPOETs on the multi-objective test suite. The execution time (wall-clock) for JuPOETs and POETs implemented in Octave was measured for 10 independent trials for the suite of test problems. The number of steps per temperature $\mathcal{I} = 10$, and the cooling parameter $\alpha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

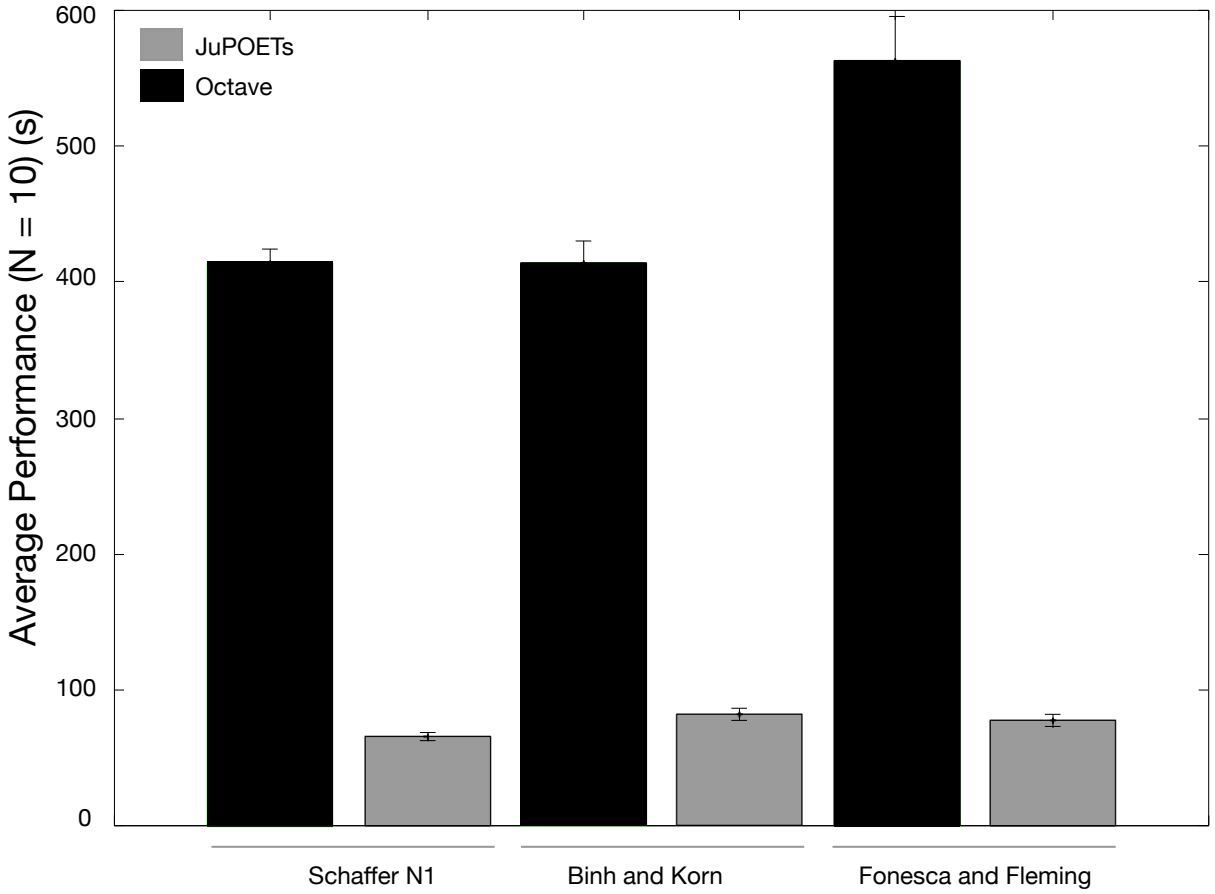

Fig. 3: Representative JuPOETs solutions for problems in the multi-objective test suite. The number of steps per temperature $\mathcal{I}$ = 10, and the cooling parameter α = 0.9 for all cases. The problem domain was partitioned into 10 equal segments, an initial guess was drawn from each segment. For 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.

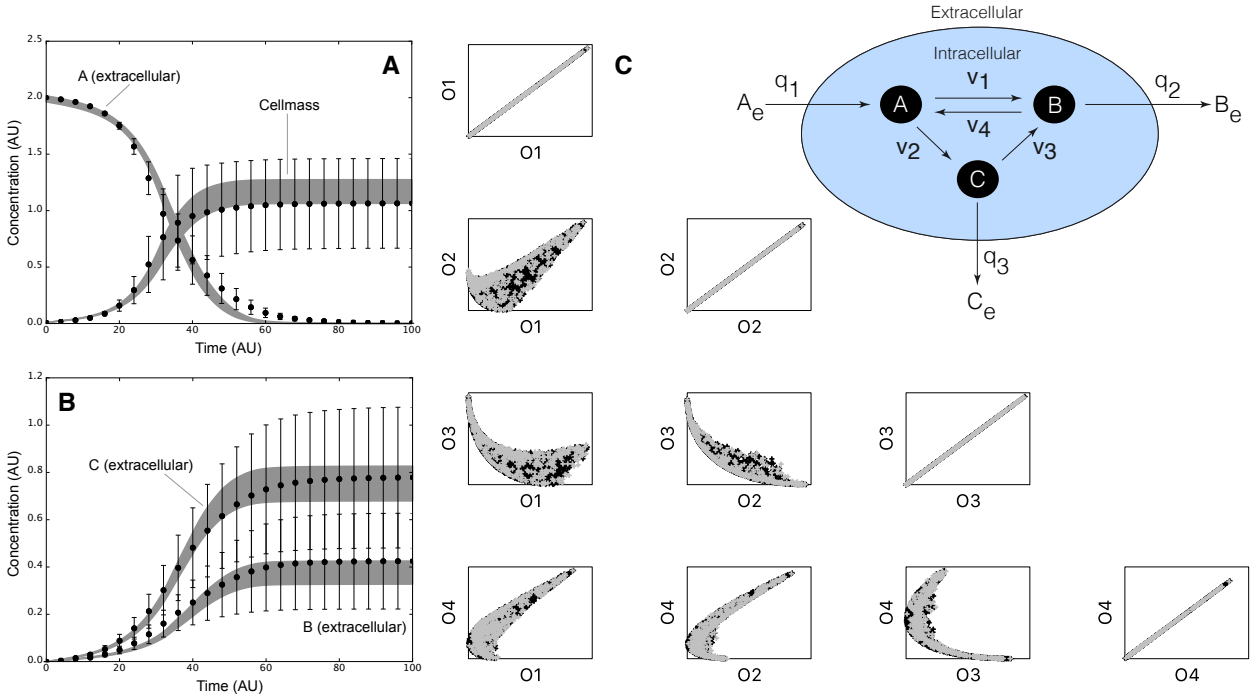

Fig. 4: Proof of concept biochemical network study. Inset right: Prototypical biochemical network with six metabolites and seven reactions modeled using the hybrid cybernetic approach (HCM). Intracellular cellmass precursors A, B, and C are balanced (no accumulation) while the extracellular metabolites 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.

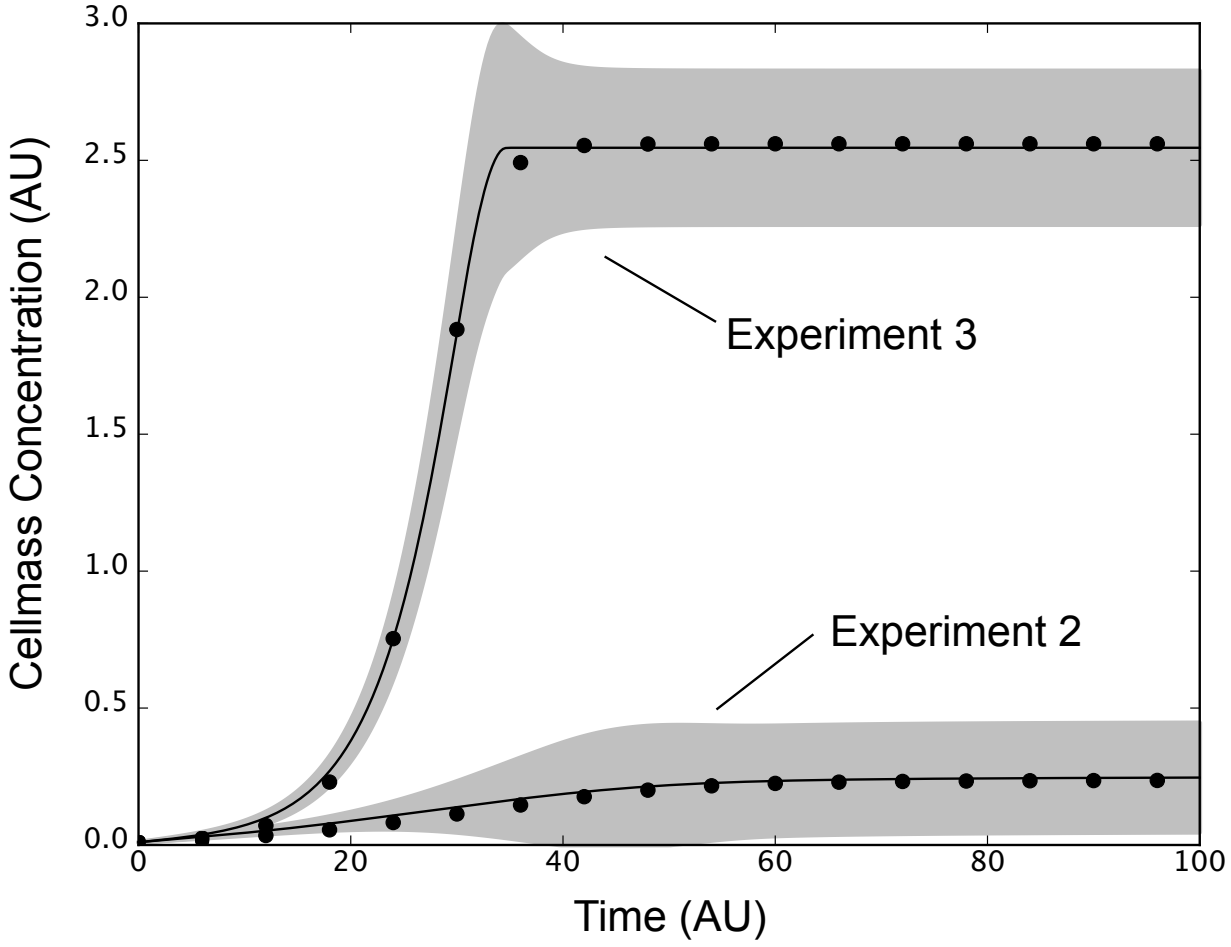

Fig. 5: Experiment to experiment variation captured by the ensemble. Cellmass measurements (points) versus time for experiment 2 and 3 were compared with ensemble simulations. The full ensemble was sorted by simultaneously selecting the top 25% of solutions for each objective with rank ≤ 1. The best 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.