Geometric algorithms for sampling the flux space of metabolic networks

Apostolos Chalkis, Ioannis Z. Emiris, Vissarion Fisikopoulos, Elias Tsigaridas, Haris Zafeiropoulos

GEOMETRIC ALGORITHMS FOR SAMPLING THE FLUX SPACE OF METABOLIC NETWORKS*

Apostolos Chalkis,† *Ioannis Z. Emiris,*‡ *Vissarion Fisikopoulos,*§ *Elias Tsigaridas,*¶ *and Haris Zafeiropoulos*‖

Abstract. Metabolic networks and their reconstruction set a new era in the analysis of metabolic and growth functions in the various organisms. By modeling the reactions occurring inside an organism, metabolic networks provide the means to understand the underlying mechanisms that govern biological systems.

Constraint-based approaches have been widely used for the analysis of such models and led to intriguing geometry-oriented challenges. In this setting, sampling uniformly points from polytopes derived from metabolic models (flux sampling) provides a representation of the solution space of the model under various conditions. However, the polytopes that result from such models are of high dimension (in the order of thousands) and usually considerably skinny. Therefore, to sample uniformly at random from such polytopes shouts for a novel algorithmic and computational framework specially tailored for the properties of metabolic models.

We present a complete software framework to handle sampling in metabolic networks. Its backbone is a Multiphase Monte Carlo Sampling (MMCS) algorithm that unifies rounding and sampling in one pass, yielding both upon termination. It exploits an optimized variant of the Billiard Walk that enjoys faster arithmetic complexity per step than the original. We demonstrate the efficiency of our approach by performing extensive experiments on various metabolic networks. Notably, sampling on the most complicated human metabolic network accessible today, Recon3D, corresponding to a polytope of dimension 5 335, took less than 30 hours. To the best of our knowledge, that is out of reach for existing software.

*Submitted to the editors October 1, 2023. A preliminary version of the article appeared in [14].

†Department of Informatics & Telecommunications, National & Kapodistrian University of Athens and Athena Research Innovation Center, Greece and GeomScale org. (achalkis@di.uoa.gr, ORCID: 0000-0002-4628-1907)

‡Athena Research Innovation Center, Greece and Department of Informatics & Telecommunications, National & Kapodistrian University of Athens. (emiris@athenarc.gr) Partial support by "INSPIRED-The National Research Infrastructures on Integrated Structural Biology, Drug Screening Efforts & Drug Target Functional Characterization" (Grant MIS 5002550), implemented under the action "Reinforcement of the Research & Innovation Infrastructure", funded by the Operational programme "Competitiveness, Entrepreneurship & Innovation" (Grant NSRF 2014–2020), co-financed by Greece and the EU (European Regional Development Fund).

§Department of Informatics & Telecommunications, National & Kapodistrian University of Athens and GeomScale org. (vfisikop@di.uoa.gr, ORCID: 0000-0002-0780-666X)

¶Inria Paris and IMJ-PRG, Sorbonne Université and Paris Université and GeomScale org. (elias.tsigaridas@inria.fr) Partially supported by ANR JCJC GALOP (ANR-17-CE40-0009).

‖Department of Biology, University of Crete and Institute of Marine Biology, Biotechnology and Aquaculture, Hellenic Centre for Marine Research. (haris-zaf@hcmr.gr, ORCID: 0000-0002-4405-6802)

1 Introduction

1.1 Background

Genome-scale metabolic models (GEMs) incorporate the vast majority of the processes that occur in a cell or an organism in a mathematical format [20] . With respect to the chemical reactions that take place in the system, this representation has several advantages as it is based on the *stoichiometry* of the reactions involved. A *chemical equation*, that is the symbolic representation of a chemical reaction, has a fixed stoichiometry, meaning the quantitative relationships between the components of a reaction are always the same. The rate of turnover of molecules through a metabolic reaction is called *flux*. GEM reconstruction and their analysis is further discussed on Section 2.1 and 2.2.

Constraint-based methods are commonly used for the analysis of GEMs [40]. As all compounds are finite the concentration of each metabolite is bounded [54], meaning that the models derived from the metabolic networks have constraints. Likewise, as the laws of thermodynamics need to apply in such systems, the flux of each reaction is also bounded. Flux Balance Analysis (FBA) [52] is the most well known type of analysis in such models aiming at estimating the minimum or the maximum of a specific (linear) *objective function* over the constrained model, typically a convex polytope. However, FBA as well as Flux Variability Analysis (FVA) [24] that return a sole or a pair of values for each reaction flux are biased methods [40]. We target complementary *unbiased* methods such as *random flux sampling* that sample (typically uniformly at random) points from a convex polytope derived from the metabolic network, explore the whole solution space of fluxes and provide a more detailed biological insight. In other words, sampling enables a thorough overview of all the potential steady states; the latter are states where the production rate of each metabolite equals its consumption rate [8]. Using uniformly distributed steady states one could estimate the probability distribution for the flux of any reaction [30] that in turn can lead to a deep understanding and statistical analysis of the metabolic network.

Flux sampling is omnipresent and fundamental computation in computational biology and thus there is a wide range of algorithms and implementations [26, 32]; we refer to [19] for a thorough list. Nevertheless, sampling metabolic networks with thousands of chemical reactions is challenging from the computational point of view [27, 60] and undoubtedly demands more computational resources than solving a linear optimization problem as is the case in FBA. Interestingly, up to now, there is no efficient method for handling models in more than typically a thousand of dimensions; to our understanding, this is so because of a number of computational geometry oriented challenges that we have to overcome, see Section 2.2. As consequence, to the best of our knowledge, up to now we were not able to sample accurately, that is under various statistical guarantees, from the flux space of the latest human metabolic network, a network with 13 543 reactions.

1.2 Our contribution

We introduce a Multi-phase Monte Carlo Sampling (MMCS) algorithm (Section 4 and Algorithm 2) to sample from a polytope P. In particular, we split the sampling procedure

in phases where, starting from P, each phase uses the sample to round the polytope and provide it as input to the next phase. This improves the efficiency of the random walk in the next phase. For sampling, we propose an improved variant of Billiard Walk (Section 3 and Algorithm 1) that enjoys faster arithmetic complexity per step. We also handle efficiently, alas not formally, the potential arithmetic inaccuracies near to the boundary. We apply sufficiently small numerical perturbations such that the walk always stays inside the body. For a different approach with guarantees, we refer to [16, 9]. We accompany the MMCS algorithm with a powerful MCMC diagnostic, namely the estimation of Effective Sample Size (ESS), to identify a satisfactory convergence to the uniform distribution. However, our method is flexible and we can use any random walk and combination of MCMC diagnostics to decide convergence.

The open-source implementation of our algorithms[1] provides a complete software framework to sample efficiently in metabolic networks. We demonstrate the efficiency of our tools by performing experiments on multiple metabolic networks, coming from the BiGG database and having a great range of dimensions, and by comparing with the state-of-the-art software package, `cobra` (Section 5.2). Our implementation is faster than `cobra` for low dimensional models, with a speed-up that ranges from 10 to 100 times; this gap on running times increases for bigger models (Table 1). We measure the quality of the sample that our software produces using two widely used diagnostics, i.e., the ESS and the potential scale reduction factor (PSRF) [21]. The highlight of our method is the ability to sample from the most complicated human metabolic network that is accessible today, namely Recon3D. In Figure 3 we estimate marginal univariate and bivariate flux distributions in Recon3D which validate (a) the quality of the sample by confirming a mutually exclusive pair of biochemical pathways, and that (b) our method indeed generates steady states. In particular, our software can sample $1.44 \cdot 10^5$ points from a 5 335-dimensional polytope in a day using modest hardware. Combined with the nature of our MMCS algorithm (see Section 4) this indicates the capability of our approach for sampling the flux space of the most complex metabolic networks available. To our understanding this task is out of reach for existing software. Lastly, MMCS algorithm is a quite general sampling scheme and so it has the potential to also address other hard computational problems like multivariate integration and volume estimation of polytopes.

A preliminary version of this paper appeared in [14]. The current full version contains additional and more detailed experimental results, all the proofs of the various statements and theorems, the pseudocode of all the algorithms, an updated discussion of previous work, and a more detailed presentation of our approach and tools.

Paper outline. In the next section we present a thorough description of both the biological and the geometrical notions that are the backbone of our framework. Section 3 details the Billard Walk random walk, while we introduce our Multiphase Monte Carlo Sampling algorithms in Section 4. Finally, in Section 5 we present our open-source implementation and various experimental results to highlight he potential our approach. We conclude in Section 6, where we also present some future directions.

[1] https://github.com/GeomScale/volume_approximation/tree/v1.1.0-2

2 Random Sampling on Metabolic Networks' Flux Space

2.1 GEMs: an overview

Metabolism, the process that modifies molecules and maintains the living state of a cell or an organism through a set of chemical reactions, penetrates most of the different levels of living entities horizontally [61]. Reactions begin with a particular molecule(s) (reactants) that is converted into some other molecule(s) (products). This process is catalyzed by proteins called *enzymes*; enzymes accelerate the reaction by lowering its activation energy. Enzymes are not consumed during the reaction, and they do not alter the equilibrium of the reaction. Each enzyme binds only with a specific substrate allowing for complex regulation at the cellular level; it is the complementary geometric shapes that make a substrate–enzyme pair to fit exactly into one another ("lock – key" relationship). Thus, this one-to-one relationship, allows us to use enzymes to denote a reaction; e.g., by Phosphoenolpyruvate carboxylase (PEP) is the enzyme that catalyzes the following reaction:

$$\text{phosphate} + \text{oxaloacetate} \xrightarrow{\text{PEP}} \text{phosphoenolpyruvate} + HCO_3{}^-.$$

In turn, products returned from a reaction may be the reactants in another reaction. In addition, a reaction can be either reversible or irreversible, meaning the products of a reaction can or cannot convert back to the reactants. The reaction catalyzed by hexokinase (HEX) described above is irreversible. Linked reactions, where the product of the first acts as the substrate for the next, build up metabolic pathways. Each pathway is responsible for a certain function. We can link together the aggregation of all the pathways that take place in an organism (and their corresponding reactions) and represent them mathematically using the reactions' stoichiometry. Therefore, at the species level, metabolism is a network of its metabolic pathways and we call these representations *metabolic networks*.

As discussed in [56] a great range of constraints govern the cells' operations; for a thorough overview on the constraints cells operate under, you may see [54], Chapter 16.5. (Bio)physico-chemical- (e.g., thermodynamics, nutrient uptake, oxygen availability etc.) as well as connectivity-, capacity- and rates-related constraints are applied on the functions of such a network. Each of the aforementioned constraint categories include multiple constraints, such as thermodynamics- and gene-expression-oriented constraints that add extra complexity in the model. The more constraints a model incorporates, the more accurate the flux distributions it returns. Thermodynamics, metabolome, physiological and labelling data can be also integrated in such models [59]. Apparently, being a knowledge base, a GEM is refined from time to time through the integration of new information for the organism and the comparison with experimental data. Moreover, even if a GEM has no information but the reactions of the network, certain constraints rule the model i.e., the stoichiometry of the reactions, their reversibility and the flux range that apply [20].

Over the last few years though and especially for the case of prokaryotic taxa, automatic reconstruction approaches for genome-scale metabolic models of relatively high quality have been developed [46]. This way, starting from a metagenomic dataset, obtaining GEMs for all the species present in a microbial community becomes now possible [68].

2.2 From metabolism to computational geometry

The dynamic mass balance on a chemical compound is the difference between the sum of the fluxes of all the reactions that form it and the sum of all the reactions that degrade it. In general, the following ordinary differential equation expresses such a mass balance:

$$\frac{d\omega_i}{dt} = \sum_k s_{ik} v_k = \langle s_i \cdot v \rangle, \tag{1}$$

where ω_i is the total mass of the i-th metabolite, s_{ik} is the stoichiometric coefficient for this metabolite in the k-th reaction, and v_k is the flux of the k-th reaction. By considering all the differential equations expressing the dynamic mass balance of all the compounds present in a metabolic network, we have

$$\frac{d\omega}{dt} = Sv, \tag{2}$$

where S is the stoichiometric matrix having as rows the vectors s_i. In this setting, S is the map of the linear transformation that sends the flux vector to a vector of time derivatives of the concentration vector [54].

To solve (2) there are two main approaches [64]: (a) *Constraint based metabolic models*, where it is assumed that the system is at a steady-state; in other words the metabolite concentrations are relative constant. We can also use gene expression data and metabolic flux measurements, e.g., from ^{13}C (carbon-13) labelling experiments, to further constrain and validate the model. (b) *Dynamic metabolic models*, where we use mathematical functions (based on generalised Michaelis-Menten kinetics) to describe v with measured or estimated values of k and we approximate numerically the solution of the system [65]. Subsequently, we can compare the simulations of the model to measured time-series profiles of metabolite concentrations. We shall briefly review both approaches. Notably, the fluxes will generally depend on the concentrations of metabolites in the network and the parameters or the kinetic rates k.

In metabolic networks analysis mass and energy are considered to be conserved [53]. As many homeostatic states are close to steady states [62], we commonly use the latter in metabolic networks analysis.

Stoichiometric coefficients are the number of molecules a biochemical reaction consumes and produces. The coefficients of all the reactions in a network, with m metabolites and n reactions ($m < n$), form the *stoichiometric matrix* $S \in \mathbb{R}^{m \times n}$ [54]. The nullspace of S corresponds to the steady states of the network:

$$S \cdot v = 0, \tag{3}$$

where $v \in \mathbb{R}^n$ is the *flux vector* that contains the fluxes of each chemical reaction of the network. As all the fluxes are bounded, for each coordinate v_i of the vector v, there are constants $v_{ub,i}$ and $v_{lb,i}$, such that $v_{lb,i} \leq v_i \leq v_{ub,i}$, for $i \in [n]$. Hence, in total we have $2n$ constants and $2n$ constraints.

We obtain the constraints from explicit experimental information. In cases where there is no such information, reactions are left unconstrained by setting arbitrary large values

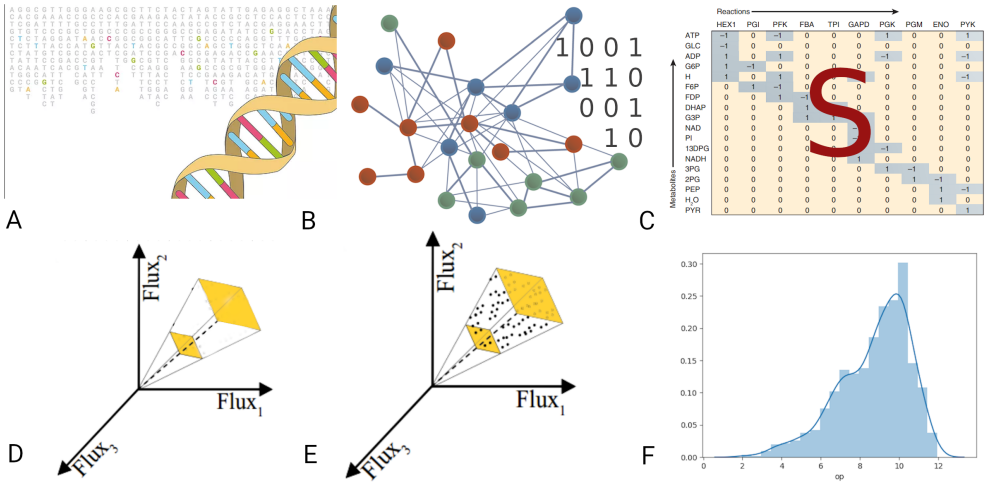

Figure 1: From DNA sequences to distributions of metabolic fluxes. (A) The genes of an organism provide us with the enzymes that it can potentially produce. Enzymes are like a blueprint for the reactions they can catalyze. (B) Using the enzymes we identify the reactions in the organism. (C) We construct the stoichiometric matrix of the metabolic model. (D) We consider the flux space under different conditions (e.g., steady states); they correspond to polytopes containing flux vectors addressing these conditions. (E) We sample from polytopes that are typically skinny and of high dimension. (F) The distribution of the flux of a reaction provides great insights to biologists.

to their corresponding bounds according to their reversibility properties; i.e., if a reaction is reversible, then its flux might be negative [44]. The constraints define a n-dimensional box containing both the steady and the dynamic states of the system. If we intersect that box with the nullspace of S, then we define a polytope that encodes all the possible steady states and their flux distributions [54]. We call it the steady-state *flux space*. Figure 1 illustrates the complete workflow from building a metabolic network to computing a flux distribution.

To study the global features of a metabolic network we use this polytopal representation of the set of steady states. To obtain an accurate picture of the whole solution space we sample from it uniformly distributed points. Subsequently, the biologists can study the properties of certain components of the whole network and deduce significant biological insights [54].

2.3 Metabolic networks through the lens of random sampling

Efficient uniform random sampling on polytopes resulting from metabolic networks is a very challenging task both from the theoretical (algorithmic) and the engineering (implementation) point of view.

First, the dimension of the polytopes is of the order of certain thousands. This requires, for example, advanced engineering techniques to cope with memory requirements

and to perform linear algebra operations with large matrices; e.g., in Recon3D [7] we compute the nullspace of a $8\,399 \times 13\,543$ matrix. Second, the polytopes are rather skinny (Section 5); this makes it harder for sampling algorithms to move in the interior of polytopes and calls for novel practical techniques to sample.

There is extended on-going research concerning advanced algorithms and implementations for sampling metabolic networks over the last decades. Markov Chain Monte Carlo algorithms such as Hit-and-Run (HR) [63] have been widely used to address the challenges of sampling. Two variants of HR are the non-Markovian Artificial Centering Hit-and-Run (ACHR) [36] that has been widely used in sampling metabolic models, e.g., [58], and Coordinate Hit-and-Run with Rounding (CHRR) [26]. The latter is part of the `cobra` toolbox [29], the most commonly used software package for the analysis of metabolic networks. CHRR enables sampling from complex metabolic network corresponding to the highest dimensional polytopes so far. There are also stochastic formulations where the inclusion of experimental noise in the model makes it more compatible with the stochastic nature of biological networks [45]. The recent study in [19] offers an overview as well as an experimental comparison of the currently available samplers.

These implementations played a crucial role in actually performing in practice uniform sampling from the flux space. However, they are currently limited to handle polytopes of dimension, say, less than or equal to 2 500 [19, 26]. This is also the order of magnitude of the most complicated, so far, metabolic network model built, Recon3D [7]. By including 13 543 metabolic reactions and involving 4 140 unique metabolites, Recon3D provides a representation of the 17% of the functionally of annotated human genes. To our knowledge, there is no method that can efficiently handle sampling from the flux space of Recon3D. Moreover, the dimension of the polytopes will keep rising and not only for the ones corresponding to human metabolic networks. Sampling in polytopes derived from networks of networks are the next big thing in metabolic networks analysis [3, 55].

Regarding the sampling process, from the theoretical point of view, we are interested in the convergence time, or *mixing time*, of the Markov Chain, or geometric *random walk*, to the target distribution. Given a d-dimensional polytope P, the mixing time of several geometric random walks (e.g., HR or Ball Walk) grows quadratically with respect to the sandwiching ratio R/r of the polytope [41, 43]. Here r and R are the radii of the smallest and largest ball with center the origin that contains, and is contained, in P, respectively; i.e., $rB_d \subseteq P \subseteq RB_d$, where B_d is the unit ball. It is crucial to reduce R/r, that is to put P in a well rounded position where $R/r = \widetilde{\mathcal{O}}(\sqrt{d})$; the $\widetilde{\mathcal{O}}(\cdot)$ notation means that we are ignoring polylogarithmic factors. A powerful approach to obtain well roundness is to put P in *near isotropic position*. Roughly speaking, the idea is to apply a linear transformation to the polytope obtain a polytope that is (as much as possible) close to ball, that in turn has sandwiching ration equal to 1. In general, $K \subset \mathbb{R}^d$ is in isotropic position if the uniform distribution over K is in isotropic position, that is $\mathbb{E}_{X\sim K}[X] = 0$ and $\mathbb{E}_{X\sim K}[X^T X] = I_d$, where I_d is the $d \times d$ identity matrix. Thus, to put a polytope P into isotropic position one has to generate a set of uniform points in its interior and apply to P the transformation that maps the point-set to isotropic position; then we iterate this procedure until P is in c-isotropic position [17, 43], for a constant c that indicates how close we are to a ball. In [1] they prove that $\mathcal{O}(d)$ points suffice to achieve 2-isotropic position. Alternatively in [26]

they compute the maximum volume ellipsoid in P, they map it to the unit ball, and then apply to P the same transformation. They experimentally show that a few iterations suffice to put P in John's position [33]. Moreover, there are a few algorithmic contributions that combine sampling with distribution isotropization steps, e.g., the multi-point walk [4] and the annealing schedule [35].

An important parameter of a random walk is the *walk length*, that is the number of the intermediate points that a random walk visits before producing a single sample point. The longer the walk length of a random walk is, the smaller the distance of the current distribution to the stationary (target) distribution becomes. For the majority of random walks there are bounds on the walk length to bound the mixing time with respect to a statistical distance. For example, HR generates a sample from a distribution with total variation distance less than ϵ from the target distribution after $\widetilde{O}(d^3)$ [43] steps, in a well rounded convex body and for log-concave distributions. Similarly, CDHR mixes after a polynomial, in the diameter and the dimension, number of steps [38, 49] for the case of uniform distribution. However, extended practical results show that both CDHR and HR converge after $\mathcal{O}(d^2)$ steps [11, 17, 26]. The leading algorithms for uniform polytope sampling are the Riemannian Hamiltonian Monte Carlo sampler [39] and the Vaidya walk [15], with mixing times $\widetilde{\mathcal{O}}(md^{2/3})$ and $\widetilde{\mathcal{O}}(m^{1/2}d^{3/2})$ steps, respectively, where $m \geq d$ is the number of linear constraints that define the polytope. However, it is not clear if these random walks can outperform CDHR in practice, because of their high cost per step and numerical instability.

Billiard Walk [23] is a random walk that employs linear trajectories in a convex body with boundary reflections; alas with an unknown mixing time. The closest guarantees for its mixing time are those of HR and stochastic billiards [18]. Interestingly, [23] shows that, experimentally, Billiard Walk converges faster than HR for a proper tuning of its parameters. The same conclusion follows from the computation of the volume of zonotopes [10]. It is not known how the sandwiching ratio of P affects the mixing time of Billiard Walk. Since Billiard Walk employs reflections on the boundary, we can consider it as a special case of Reflective Hamiltonian Monte Carlo [16, 9].

For almost all random walks the theoretical bounds on their mixing times are pessimistic and unrealistic for computations. Hence, if we terminate the random walk earlier, we generate samples that are usually highly correlated. There are several *MCMC Convergence Diagnostics* [57] to check if the quality of a sample can provide an accurate approximation of the target distribution. For a dependent sample, a powerful diagnostic is the *Effective Sample Size* (ESS). It is the number of effectively independent draws from the target distribution that the Markov chain is equivalent to. For autocorrelated samples, ESS probabilistically bounds the estimation error of a value of interest when dependent samples are used [22] and provides information about the quality of the sample. The effective sample size, N_{eff}, of N samples generated by a process with autocorrelations ρ_t is

$$N_{\text{eff}} = \frac{N}{\sum_{t=-\infty}^{+\infty} \rho_t} = \frac{N}{1 + 2\sum_{t=0}^{+\infty} \rho_t}. \tag{4}$$

In the MCMC setting, we cannot compute the integral involved in the computation of the autocorrelation ρ_t at lag t in Equation (4). Thus, in general, given an autocorrelated sample, the exact value of N_{eff} is unknown. In practice, these quantities are estimated by

the generated sample itself, using a proper estimator $\hat{\rho}_t$ of the autocorrelation. A second popular MCMC Diagnostic is the potential scale reduction factor (PSRF) [21]. In [21] they split the sample into subsets, called *chains*. Then, they define a statistic measure as the ratio between the average sample variance within each chain and the pooled variance of samples across chains. This statistic approximates from above the value 1. The better the convergence of the empirical distribution to the target distribution is, the closest to 1 is the value of the statistic. In applications we usually set a threshold to declare convergence. In [21] they recommend terminating simulation when PSFR < 1.1 which is widely considered as the primary option. In particular, about the 90% of the papers [67] that use PSRF set the threshold to 1.1 or smaller and only the rest set a larger threshold. Moreover, to improve PSRF ≥ 1.2 to values ≤ 1.1 is a computational hard task [6].

Last but not least, we use classical statistical tools for the first time in metabolic network analysis. More precisely, we use bivarate copula estimation for the joint distribution between two reaction fluxes (see Figure 3). Using copulas we capture the dependence between two fluxes. When the mass is concentrated on the main/down diagonal it implies a positive/negative dependency between the fluxes. For a detailed introduction to copulas we refer the reader to [50].

3 Efficient Billiard walk

The geometric random walk of our choice to sample from a polytope is based on Billiard Walk [23], which we modify to reduce its cost per step by a factor. In particular, we remember various computations of the previous iterations and we precompute the normals of the facets of the polytope.

For a polytope $P = \{x \in \mathbb{R}^d \,|\, Ax \leq b\}$, where $A \in \mathbb{R}^{k\times d}$ and $b \in \mathbb{R}^k$, Billiard Walk starts from a given point $p_0 \in P$, selects uniformly at random a direction, say u_0, and it moves along the direction of u_0 for length L; it reflects on the boundary if necessary. This results a new point p_1 inside P. We repeat the procedure from p_1. Asymptotically it converges to the uniform distribution over P. The length is $L = -\tau \ln \eta$, where η is a number chosen uniformly (at random) in $(0, 1)$, that is $\eta \sim \mathcal{U}(0, 1)$, and τ is a predefined constant. It is useful to set a bound, say ρ, on the number of reflections to avoid computationally hard cases where the trajectory may stuck in corners. In [23] they set $\tau \approx \text{diam}(P)$, the diameter of P, and $\rho = 10d$. Our choices for τ and ρ depend on a burn-in step that we detail in Section 5.

At each step of Billiard Walk, we compute the intersection point of a ray, say $\ell := \{p + tu,\ t \in \mathbb{R}_+\}$, with the boundary of P, ∂P, and the normal vector of the tangent plane of P at the intersection point. The inner vector of the facet that the intersection point belongs to is a row of A. To compute the point $\partial P \cap \ell$, where the first reflection of a Billiard Walk step takes place, we need to compute the intersection of ℓ with all the hyperplanes that define the facets of P. This corresponds to solve (independently) the following m linear equations

$$a_j^T(p_0 + t_j u_0) = b_j \Rightarrow t_j = (b_j - a_j^T p_0)/a_j^T u_0,\ j \in [k], \tag{5}$$

and keep the smallest positive t_j; a_j is the j-th row of the matrix A. We solve each equation

Algorithm 1 Billiard Walk(P, p, ρ, τ, W)

Input: polytope P; point $p \in P$; upper bound on the number of reflections ρ; parameter τ to adjust the length of the trajectory; walk length W.
Output: a point in P (uniformly distributed in P).

```
for j = 1, ..., W do
  L ← −τ ln η;  η ∼ U(0,1)    {length of the trajectory}
  i ← 0    {current number of reflections}
  p_0 ← p    {initial point of the step}
  pick a uniform vector u_0 on the boundary of the unit ball {initial direction}
  while i ≤ ρ do
    ℓ ← {p_i + t u_i, 0 ≤ t ≤ L}    {this is a segment}
    if ∂P ∩ ℓ = ∅ then
      p_{i+1} ← p_i + L u_i    break
    end if
    p_{i+1} ← ∂P ∩ ℓ;    {point update}
    the inner vector, s, of the tangent plane at p,
      s.t. ||s|| = 1,  L ← L − |P ∩ ℓ|, u_{i+1} ← u_i − 2(u_i^T s)s    {direction update}
    i ← i + 1
  end while
  if i = ρ then
    p ← p_0
  else
    p ← p_i
  end if
end for
return p
```

in $\mathcal{O}(d)$ operations and so the overall complexity is $\mathcal{O}(dk)$, where k is the number of rows of A and thus an upper bound on the number of facets of P. A straightforward approach for Billiard Walk would consider that each reflection costs $\mathcal{O}(kd)$ and so the per step cost is $\mathcal{O}(\rho kd)$. However, our improved version performs more efficiently both *point* and *direction updates* by remembering some computations from the previous iteration and introducing a preprocessing step. The preprocessing step involves the normal vectors of the facets and takes k^2d operations. So the amortized per-step complexity of Billiard Walk becomes $\mathcal{O}((\rho + d)k)$. The pseudo-code of this approach appears in Algorithm 1.

Lemma 1. The amortized cost per step complexity of Billiard Walk (Algorithm 1) is $\mathcal{O}((\rho + d)k)$ after a preprocessing step that takes $\mathcal{O}(k^2d)$ operations, where ρ is the maximum number of reflections per step.

Proof. The first reflection of a Billiard Walk step costs $\mathcal{O}(kd)$. During its computation, we store all the values of the inner products $a_j^T x_0$ and $a_j^T u_0$. At the reflection $i > 0$, we start

from a point x_i and the solutions of the corresponding linear equations are

$$\begin{aligned} a_j^T(p_i + t_j u_i) = b_j \Rightarrow a_j^T(p_{i-1} + t_{i-1}u_{i-1}) + t_j a_j^T(u_{i-1} - 2(u_{i-1}^T a_r)a_r) = b_j \\ \Rightarrow t_j = \frac{b_j - a_j^T(p_{i-1} + t_{i-1}u_{i-1})}{a_j^T(u_{i-1} - 2(u_{i-1}^T a_r)a_r)}, \quad \text{for } j \in [k], \end{aligned} \tag{6}$$

$$\text{and } \; u_{i+1} = u_i - 2(u_i^T a_l)a_l, \tag{7}$$

where a_r, a_l are the normal vectors of the facets that ℓ hits at reflection $i-1$ and i respectively, and t_{i-1} the solution of the reflection $i-1$. The index l of the normal a_l corresponds to the equation with the smallest positive t_j in (6). We solve each of the equations in (7) in $\mathcal{O}(1)$ time, based on our bookkeeping from the previous reflection. We also store the inner product $u_i^T a_l$ in (7) from the previous reflection. After computing all $a_i^T a_j$ as a preprocessing step, which takes $k^2 d$ operations, the total per-step cost of Billiard Walk is $\mathcal{O}((d+\rho)k)$. □

4 Multiphase Monte Carlo Sampling algorithm

To sample steady states in the flux space of a metabolic network, with m metabolites and n reactions, we introduce a Multiphase Monte Carlo Sampling (MMCS) algorithm; it is multiphase because it consists of a sequence of sampling phases.

Let $S \in \mathbb{R}^{m \times n}$ be the stoichiometric matrix and let v_{lb}, $v_{ub} \in \mathbb{R}^n$ be the bounds on the fluxes. The flux space is the bounded convex polytope

$$\text{FS} := \{v \in \mathbb{R}^n \,|\, Sv = 0,\ v_{lb} \le v \le v_{ub}\} \subset \mathbb{R}^n. \tag{8}$$

The dimension, d, of FS is less than the dimension of the ambient space; that is $d \le n$. To work with a full dimensional polytope we restrict the box induced by the inequalities $v_{lb} \le v \le v_{ub}$ to the nullspace of S. Let the H-representation of the box be $\left\{ v \in \mathbb{R}^n \,\middle|\, \begin{pmatrix} I_n \\ -I_n \end{pmatrix} v \le \begin{pmatrix} v_{ub} \\ v_{lb} \end{pmatrix} \right\}$, where I_n is the $n \times n$ identity matrix, and let $N \in \mathbb{R}^{n \times d}$ be the matrix of the nullspace of S, that is $SN = 0_{m \times d}$. Then $P = \{x \in \mathbb{R}^d \mid Ax \le b\}$, where $A = \begin{pmatrix} I_n N \\ -I_n N \end{pmatrix}$ and $b = \begin{pmatrix} v_{ub} \\ v_{lb} \end{pmatrix}$, is a full dimensional polytope (in $\mathbb{R}^d$). After we sample (uniformly) points from P, we transform them to uniformly distributed points (that is steady states) in FS by applying the linear map induced by N.

MMCS generates, in a sequence of sampling phases, a set of points, that are almost equivalent to n independent uniformly distributed points in P, for a given n. At each phase, it employs Billiard Walk (Section 3) to sample approximate uniformly distributed points, rounding to speedup sampling, and uses the Effective Sample Size (ESS) diagnostic to decide termination. The pseudo-code of the algorithm appears in Algorithm 2.

Overview. Initially we set $P_0 = P$. At each phase $i \ge 0$ we sample at most λ points from P_i. We generate them in chunks; we also call them *chain* of sampling points. Each chain contains at most l points (for simplicity consider $l = \mathcal{O}(1)$). To generate the points in each

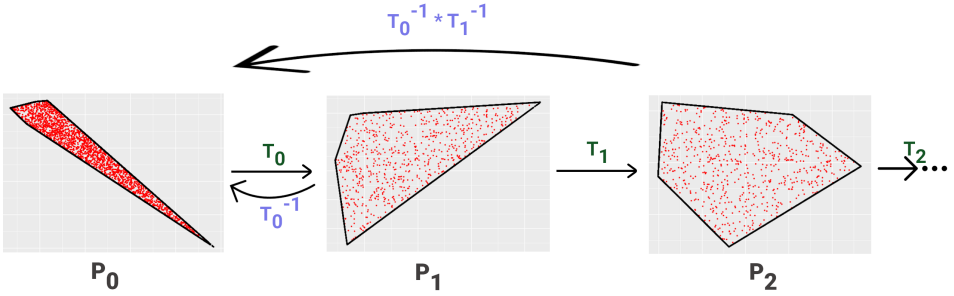

Figure 2: An illustration of our Multiphase Monte Carlo Sampling algorithm. The method is given an integer n and starts at phase $i = 0$ to sample from P_0. During each phase, it samples a maximum number of points λ. If the sum of Effective Sample Size at a phase becomes larger than n before the total number of samples in P_i reaches λ, then the algorithm terminates. Otherwise, we proceed to a new phase. At the end, we map back to P_0 all the generated samples of each phase, using the product of the inverse trasformations.

chain we employ Billiard Walk, starting from a point inside P_i; the starting point is different for each chain. We repeat this procedure until the total number of samples in P_i reaches the maximum number λ; we need $\frac{\lambda}{l}$ chains. To compute a starting point for a chain, we pick a point uniformly at random in the Chebychev ball of P_i (that is, the largest ball inscribed in P_i) and we perform $\mathcal{O}(\sqrt{d})$ burn-in Billiard Walk steps to obtain a warm start. The burn-in steps are the prelinary steps we perform (and we thrown away) before we start producing useful samples.

After we have generated λ sample points we perform a rounding step on P_i to obtain the polytope of the next phase, P_{i+1}. In particular, we compute a linear transformation, T_i, that puts the sample into isotropic position and then $P_{i+1} = T_i(P_i)$. The efficiency of Billiard Walk improves from one phase to the next one because the sandwiching ratio decreases and so the average number of reflections decreases and thus the convergence to the uniform distribution accelerates (Section 5.2). That is we obtain faster a sample of better quality. Finally, the (product of the) inverse transformations maps the samples to $P_0 = P$. Figure 2 depicts the whole procedure.

Termination. There are no bounds on the mixing time of Billiard Walk [23], hence for termination we rely on ESS. MMCS terminates when the minimum ESS among all the univariate marginals is larger than a requested value. We chose the marginal distributions (of each flux) because they are essential for systems biologists, see [5] for a typical example. In particular, after we generate a chain, the algorithm updates the ESS of each univariate marginal to take into account all the points that we have sampled in P_i, including the ones in the newly generated chain. We keep the minimum, say n_i, among all marginal ESS values. If at the j-th phase $\sum_{i=0}^{j} n_i$ becomes larger than n before the total number of samples in P_i reaches the upper bound λ, then MMCS terminates. Otherwise, we proceed to the next phase. In summary, MMCS terminates when the sum of the minimum marginal ESS values of each phase reaches n.

Algorithm 2 Multiphase Monte Carlo Sampling$(P, n, l, \lambda, \rho, \tau, W)$

Input: A full dimensional polytope $P \in \mathbb{R}^d$;
requested effectiveness $n \in \mathbb{N}$ (number of sampled points);
l length of each chain; λ maximum number of points generated in each phase λ;
upper bound on the number of reflections ρ; parameter τ to adjust the length of the trajectory; walk length W.

Output: a set n of approximate uniformly distributed points $S \in P$

Set $P_0 \leftarrow P,\ sum\_ess \leftarrow 0,\ S \leftarrow \emptyset,\ i \leftarrow 0,\ T_0 = I_d$
while $sum\_ess < n$ **do**
  $sum\_point\_phase \leftarrow 0,\ U \leftarrow \emptyset$
  while $sum\_point\_phase < \lambda$; **do**
    Set $Q \leftarrow \emptyset$; Generate a starting point $q_0 \in P_i$;
    for $j = 1, \ldots, l$ **do**
      $q_j \leftarrow$Billiard_Walk$(P_i, q_{j-1}, \rho, \tau, W)$, Store the point q_j to the set Q
    end for
    $S \leftarrow S \cup T_i^{-1}(Q),\ U \leftarrow U \cup Q$
    $sum\_point\_phase \leftarrow sum\_point\_phase + l$,
    Update ESS n_i of this phase
    if $sum\_ess + n_i \geq n$ **then**
      break
    end if
  end while
  $sum\_ess \leftarrow sum\_ess + n_i$
  Compute T such that $T(U)$ is in isotropic position
  $P_{i+1} \leftarrow T(P_i),\ T_{i+1} \leftarrow T_i \circ T,\ i \leftarrow i + 1$
end while
return S

Rounding step. This step is motivated by the theoretical result in [1] and the rounding algorithms [43, 17]. We apply the linear transformation T_i to P_i so that the sandwiching ratio of P_{i+1} is smaller than that of P_i. This also improves (progressively) the efficiency of Billiard Walk (Section 5). To find the suitable T_i we compute the SVD decomposition of the matrix that contains the sample row-wise [2].

Updating the Effective Sample Size. We would like to update the ESS estimator of each univariate marginal for each new chain that the random walk generates. The effective sample size of a sample of points generated by a process with autocorrelations ρ_t at lag t is given by Equation (4); its exact value is unknown. Following [22], we efficiently compute ESS using a finite sum of monotone estimators $\hat{\rho}_t$ of the autocorrelation at lag t and by exploiting Fast Fourier Transform. Since the noise on the autocorrelation estimate $\hat{\rho}_t$ increases as t increases, we use the truncated sum introduced in [22]. In particular, if we sum every pair of consecutive $\hat{\rho}_t + \hat{\rho}_{t+1}$ where t being an even number, then the value of the sum

is guaranteed to be positive. Thus, in [22] they truncate the sum when they find the first negative pair. We also compute the smoothing from [22] to impose monotonicity in the sequence of autocorrelations. The latter improves the robustness of the estimation. Furthermore, given M chains of samples, the autocorrelation estimator $\hat{\rho}_t$ is

$$\hat{\rho}_t = 1 - \frac{C - \frac{1}{M}\sum_{i=1}^{M} \hat{\rho}_{t,i}}{B}, \tag{9}$$

where C and B are the within-sample variance estimate and the multi-chain variance estimate given in [21] and $\hat{\rho}_{t,i}$ is an estimator of the autocorrelation of the i-th chain at lag t. To update the ESS, for every new chain of points the algorithm generates, we compute the estimator of its autocorrelation. Then, using Welford's algorithm we update the average of the estimators of autocorrelation at lag t, as well as the between-chain variance and the within-sample variance estimators [21]. Finally, we update the ESS using these estimators.

Lemma 2 (Complexity of MMCS per phase)**.** Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$, where $A \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^k$, be a full dimensional polytope in $\mathbb{R}^d$. To sample a set of points (approximately) uniformly distributed in P, MMCS (Algorithm 2) performs $\mathcal{O}(W(\rho + d)k\lambda + \lambda^2 d + d^3)$ arithmetic operations per phase, where W is the walk length of Billiard Walk, ρ is an upper bound on the number of reflections, and λ and upper bound on the points generated at each phase.

Proof. The cost per step of Billiard Walk is $\mathcal{O}((\rho + d)k)$. In each phase we generate, using Billiard Walk, at most λ points with walk length W. Thus, the cost to generate these points is $\mathcal{O}(W(\rho + d)k\lambda)$.

To compute the starting point of each chain the algorithm picks a random point uniformly distributed in the Chebychev ball of P (using standard algorithms to sample from a ball) and performs $\mathcal{O}(1)$ Billiard Walk steps starting from it. The computation of the point takes $\mathcal{O}(d)$ operations and the steps require $\mathcal{O}(W(\rho + d)k)$ operations. The total number of chains is $\mathcal{O}(\lambda/l) = \mathcal{O}(\lambda)$, as $l = \mathcal{O}(1)$. Thus, the total cost to generate all the starting points is $\mathcal{O}(d\lambda + W(\rho + d)k\lambda)$. The update of ESS for each univariate marginal requires $\mathcal{O}(1)$ operations, since $l = \mathcal{O}(1)$.

If the termination criterion has not been met after generating λ points, then the algorithm computes a linear transformation to put the set of points to isotropic position. We do this by computing the SVD decomposition of the matrix that contains the set of points row-wise. This corresponds to an SVD of a $\lambda \times d$ matrix and takes $\mathcal{O}(\lambda^2 d + d^3)$ operations [31]. □

In Section 5 we discuss how to tune the parameters of MMCS to make it more efficient in practice. We also comment on the (practical) complexity of each phase, based on the tuning.

5 Implementation and Experiments

This section presents the implementation of our approach and the tuning of various parameters. We present experiments in an extended set of the BiGG models [37], including the

most complex metabolic networks, the human Recon2D [66] and Recon3D [7].

BIGG models database consists of a set of high-quality published GEMs aligned in a common BIGG models scheme so that they share a common list of reactions and metabolites. Flux bounds and reversibility are inherited from the initial GEM models. We end up to sample from polytopes of thousands of dimensions and show that our method enables flux sampling even in polytopes derived from the most complex metabolic models available. We analyze various aspects of our method such as the run-time, the efficiency, and the quality of the output. We compare against the state-of-the-art software for the analysis of metabolic networks, which is the `Matlab toolbox` of `cobra` [29]. Our implementation for low dimensional networks is two orders of magnitude faster than `cobra`. As the dimension grows, this gap on the run-time increases. The workflow of `cobra` for sampling first performs a rounding step and then samples using Coordinate Directions Hit-and-Run (CDHR).

In [32] the authors provide a `C++` implementation of the sampling method that `cobra` uses and they show that their implementation is approximately 6 times faster than `cobra`. Nevertheless, we choose to compare against `cobra`, since it additionally provides efficient preprocessing methods that are crucial for the experiments, and give an implicit comparison with [32]. The fast mixing of billiard walk allows us to use all the generated samples to approximate each flux distribution and so we compute a better flux distribution estimation. To estimate each marginal flux distribution, using the samples, we exploit Gaussian kernel density estimation. This is a non-parametric way to estimate the probability density function of a random variable. For more details we refer to [34]. We provide a complete open-source software framework to handle big metabolic networks. The framework loads a metabolic model in some standard file formats (e.g., `mat` and `json` files) and performs an analysis of the model, e.g., it estimates the marginal distributions of a given reaction flux. All the results are reproducible using our publicly available code[2].

The core of our implementation is in `C++` to optimize performance while the user interface is implemented in `R`. The package employs `eigen` [25] for linear algebra, `boost` [47] for random number generation, `mosek` [48] as the linear programming solver, and expands `volesti` [13], an open-source package for high dimensional sampling and volume approximation. All experiments were performed on a PC with `Intel Core i7-6700 3.40GHz` × `8 CPU` and `32GB RAM`. In the sequel, MMCS refers to our implementation.

5.1 Parameter tuning for practical performance

We give details on how we tune the various parameters presented in Section 4 in our implementation.

Parameters of Billiard Walk. To employ Billiard Walk (Section 3), we have to efficiently select values for the parameter τ that controls the length of the trajectory in each step, for the maximum number of reflections per step ρ, and for the walk length W of the random walk. We have experimentally found that if we set $W = 1$, then the empirical distribution converges faster to the uniform distribution. Thus, we get a higher ESS faster than the case of $W > 1$.

[2] https://github.com/GeomScale/volume_approximation/tree/v1.1.0-2

To set τ in phase i, first we set $\tau = 6\sqrt{d}r$, where r is the radius of the Chebychev ball of P_i. Next, we start from the center of the Chebychev ball, we perform $100 + 4\sqrt{d}$ Billiard Walk steps, and we store all the points in a set Q. Then we set $\tau = \max\{\max_{q \in Q}\{||q - p||_2\}, 6\sqrt{d}r\}$. For the maximum number of reflections we have found experimentally that $\rho = 100d$ is violated in less than 0.1% of the total number of Billiard Walk steps in our experiments.

Rounding step. In each phase i of our method, if the minimum value of ESS among all the marginals has not reached the requested threshold, then we use the generated sample to perform a rounding step by mapping the points to isotropic position. After we compute the SVD decomposition of (the matrix corresponding to) the point-set, we rescale the singular values so that the smallest one is 1; this is to improve numerical stability as suggested in [17].

For the the maximum number of Billiard Walk points per phase we follow the theoretical results in [4]. We have experimentally found that to improve the roundness from phase to phase, it suffices to set this number to $\lambda = 20d$, where d is the dimension of the polytope. The factor of 20 was the smallest integer s.t. MMCS rounds all the instances in the BiGG database [37] and the additional ones in Table 1. When, in any phase, the ratio between the maximum and the minimum singular value is smaller than 3, then we do not perform any new rounding step. In this case, we stay on the current phase until we reach the requested ESS value.

Remark 3. Given the stoichiometric matrix $S \in \mathbb{R}^{m \times n}$ of a metabolic network with flux bounds $v_{lb} \leq v \leq v_{ub}$, the total number of operations per phase that our implementation MMCS (Algorithm 2) performs, according the parameterization given in this section, is $\mathcal{O}(nd^2)$, where d is the dimension of the nullspace of S and n is the number of reactions occur in the metabolic network.

We note that our parameterization has to be considered jointly as a potential choice you make for a parameter affects your choice for a different one. Bertsimas04

5.2 Experiments

We test and evaluate our software on 17 models from the BIGG database as well as Recon2D and Recon3D from [51]. In particular, we sample from models that correspond to polytopes of dimension less than 100; the simplest model in this setting is the well known bacteria *Escherichia Coli*. We also sample from models that correspond to polytopes of dimension a few thousands; this is the case for Recon2D and Recon3D. We do not employ parallelism for any implementation, thus we report only sequential running times. We assess the quality of our results by employing both the Effective Sample Size (ESS) and the potential scale reduction factor (PSRF) [21]. In particular, we compute the PSRF for each univariate marginal of the sample that MMCS outputs. Following [21], a convergence is satisfying according to PSRF when all the marginals have PSRF smaller than 1.1.

To compare with `cobra`, we set the walk length of Coordinate Directions Hit-and-Run (CDHR) according to the empirical suggestion made in [26], i.e., equal to $8d^2$, where d

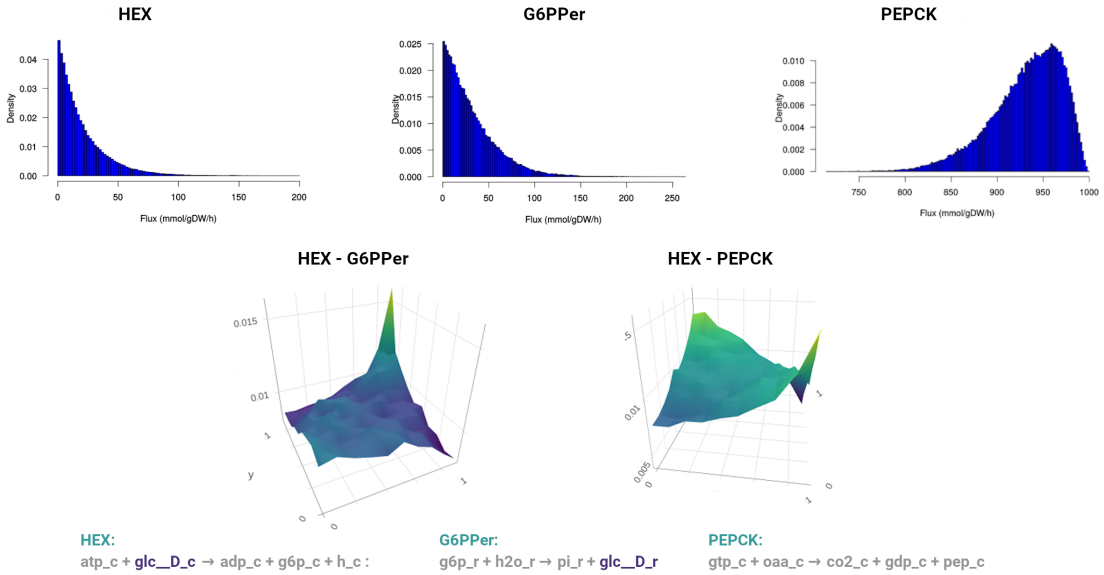

Figure 3: Flux distributions in the most recent human metabolic network Recon3D [7]. We estimate the flux distributions of the reactions catalyzed by the enzymes Hexokinase (D-Glucose:ATP) (HEX), Glucose-6-Phosphate Phosphatase, Edoplasmic Reticular (G6PPer) and Phosphoenolpyruvate carboxykinase (GTP) (PEPCK) (up). Using copulas, we estimate the joint distribution between pairs of reactions (HEX - G6PPer and HEX - PEPCK) (down).

is the dimension of the polytope we sample. In [26] they argue that this choice of walk length empirically guarantees a PSRF < 1.1 after about 1 000 samples; this is also confirmed by our experiments. For Recon2D, we follow the paradigm in [26] which shows that the method converges for walk length equal to $1.57e{+}08$. To have a fair comparison, we let `cobra` sample a minimum number of 1 000 points. If in the computed sample there is a marginal with PSRF larger than 1.1, then we continue sampling until all PSRFs are smaller than 1.1.

In Table 1, we report the results of MMCS and `cobra`. For `cobra`, we report only the run-time of the sampling phase (we do not add to it the preprocessing time). We run MMCS until we get a value of ESS equal to 1 000; i.e. we stop when the sum over all phases of the minimum values of ESS among all the marginals is larger than 1 000. All the marginals of the MMCS samples reported in Table 1 have PSRF < 1.1. This is a strong statistical evidence on the quality of the generated sample.

Comparing runtime performance, MMCS is one or two orders of magnitude faster than `cobra` and this gap becomes much larger for higher dimensional models such as Recon2D and Recon3D. Considering the experiments reported in [32], they report the run-time of CDHR for each model until it generates a sample with PSFR 1.2; for Recon3D they report ~ 1 day. Interestingly, for Recon3D, MMCS achieves PSRF 1.2 after ~ 1 hour while it reaches PSRF 1.1 after ~ 1 day.

For some models –we report them in Table 2– we introduce a further improvement to obtain a better convergence. If there is a marginal in the generated sample from MMCS that has a PSRF larger than 1.1, then we do not take into account the k first phases,

model	m	n	d	MMCS Time (sec)	MMCS N	cobra Time (sec)	cobra N
e_coli_core	72	95	24	6.50e-01	3.40e+03 (5)	7.20e+01	4.61e+06
iLJ478	570	652	59	9.00e+00	5.40e+03 (5)	4.54e+02	2.79e+07
iSB619	655	743	83	1.70e+01	8.20e+03 (5)	9.56e+02	5.51e+07
iHN637	698	785	88	2.00e+01	6.80e+03 (4)	1.03e+03	6.19e+07
iJN678	795	863	91	2.50e+01	8.10e+03 (4)	1.17e+03	6.62e+07
iNF517	650	754	92	1.70e+01	6.20e+03 (4)	1.33e+03	6.77e+07
iJN746	907	1054	116	5.70e+01	8.70e+03 (5)	2.22e+03	1.07e+08
iAB_RBC_283	342	469	130	5.20e+01	1.07e+04 (5)	7.85e+03	4.05e+08
iJR904	761	1075	227	2.98e+02	1.62e+04 (4)	8.81e+03	4.12e+08
iAT_PLT_636	738	1008	289	3.25e+02	1.04e+04 (2)	1.73e+04	6.68e+08
iSDY_1059	1888	2539	509	2.813e+03	2.31e+04 (5)	6.66e+04	2.07e+09
iAF1260	1668	2382	516	6.84e+03	5.33e+04 (6)	7.04e+04	2.13e+09
iEC1344_C	1934	2726	578	4.86e+03	3.95e+04 (4)	9.42e+04	2.67e+09
iJO1366	1805	2583	582	6.02e+03	5.14e+04 (5)	9.99e+04	2.71e+09
iBWG_1329	1949	2741	609	3.06e+03	4.22e+04 (4)	1.05e+05	2.97e+09
iML1515	1877	2712	633	4.65e+03	5.65e+04 (5)	1.15e+05	3.21e+09
Recon1	2766	3741	931	8.09e+03	1.94e+04 (2)	3.20e+05	6.93e+09
Recon2D	5063	7440	2430	2.48e+04	5.44e+04 (2)	$\sim$ 140 days	1.57e+11
Recon3D	8399	13543	5335	1.03e+05	1.44e+05 (2)	–	–

Table 1: Several, 17, metabolic networks from [37]; also Recon2D and Recon3D from [51]. The semantics of the tables are as follows: (m) the number of Metabolites, (n) the number of Reactions, (d) the dimension of the polytope; (N) is the total number of sampled points $\times$ walk length; for MMCS we stop when the sum of the minimum value of ESS among all the univariate marginals in each phase is 1 000 (we report the number of phases in parenthesis); for `cobra` we set the walk length to $8d^2$ and 1.57e+08 for Recon2D following [26], sample at least 1 000 points and stop when all marginals have PSRF < 1.1; the run-time of `cobra` for Recon2D is an estimation of the sequential time and we report it to have a rough comparison with our implementation.

starting with $k = 1$ until we get both ESS equal to 1 000 and all the PSRF values smaller than 1.1 for all the marginals. By "we do not take into account" we mean that we neither store the generated sample –for the first k phases– nor we sum up its ESS to the overall ESS considered for termination by MMCS. Note that for these models it is not practical to repeat MMCS runs for different k until we get the required PSRF value. We can obtain the final results –reported in Tables 1– in one pass. We simply drop a phase when the ESS reaches the requested value but the PSRF is not smaller than 1.1 for all the marginals. In Table 2, we separately report the MMCS runs for different k just for performance analysis reasons.

Interestingly, the total number of Billiard Walk steps –and consequently the run-time– does not increase as k increases in Table 2. This means that the performance of our method improves for these models when we do not take into account the k first phases of MMCS. This happens because the performance of Billiard Walk improves as the polytope becomes more rounded from phase to phase.

model	k	Time (sec)	PSRF < 1.1	M	N
iAF1260	0	6955	41%	6	56100
	1	6943	56%	6	54100
	2	6890	76%	6	55200
	3	6867	95%	6	53200
	4	6840	100%	6	53300
iBWG_1329	0	3067	50%	4	42100
	1	3189	97%	5	48800
	2	4652	100%	5	56500
iEC1344	0	4845	77%	4	41100
	1	4721	96%	4	42500
	2	4682	100%	4	39500
iJO1366	0	3708	66%	5	51500
	1	6022	100%	5	51400

Table 2: During our experiments we do not take into account the sample of the k first phases, thus we do not also count the value of the Effective Sample Size (ESS) in these phases, before we start storing the generated sample and sum up the ESS of each phase. In all cases MMCS stops when the sum of ESS reaches 1000. For each case we report the total run-time, the percentage of the marginals that have PSRF smaller than 1.1, the total number of phases (M) needed (including the k first phases), and the total number of Billiard Walk steps (N), including those performed in the k first phases.

In Table 3, we analyze the performance of Billiard Walk for the model iAF1260. We sample $20d$ points per phase with walk length equal to 1 and we report the average number of reflections, the ESS, the run-time, and the ratio $\psi = \sigma_{\max}/\sigma_{\min}$ per phase. The latter is the ratio of the maximum over the minimum singular value of the point-set. The larger this ratio is the more skinny the polytope of the corresponding phase is. As the method progresses from the first to the last phase, the average number of reflections and the run-time decrease and the ESS increases. This means that as the polytope becomes more rounded from phase to phase, the Billiard Walk step becomes faster and the generated sample has better quality. This explains why the total run-time does not increase when we do not take into account the first k phases: the initial phases are slow and they contribute poorly to the quality of the final sample; the last phases are fast and contribute with more accurate samples.

To illustrate further this behavior in MMCS, in Table 4 we present the ESS and the ratio ψ per phase for the runs of 5 models of different dimensionality presented in Table 1. Notice that the number of phases does not necessarily depend on the dimension, but on the roundness of the initial polytope; that is, the more rounded the polytope that MMCS takes as input the less the number of phases that MMCS constructs. Moreover, interestingly, the ratio ψ doe not decreases monotonically but for very skinny instances it may also increase from a phase to the next one. In some cases ψ has almost the same value for two or three consecutive phases until Billiard Walk captures correctly the shape of the body and round (see e.g., the case of iSDY_1059 model).

Sampling from iAF1260				
Phase	Avg. #reflections	ESS	ψ	Time (sec)
1st	7819	67	43459	2271
2nd	4909	68	922	1631
3rd	3863	77	582	1278
4th	3198	71	360	1080
5th	1300	592	29	454
6th	1187	4821	3.5	417
7th	1181	4567	2.8	415

Table 3: We sample $20d = 10\,320$ points per phase with Billiard Walk and walk length equal to 1, where $d = 516$ is the dimension of the corresponding polytope. For each phase we report the average number of reflections per BIlliard Walk step, the minimum value of Effective Sample Size among all the univariate marginals, the ratio $\psi = \frac{\sigma_{\max}}{\sigma_{\min}}$ between the maximum and the minimum singular value of the SVD decomposition of the generated sample, and the run-time.

A copula is a bivariate probability distribution for which the marginal probability distribution of each variable is uniform. This implies a positive dependency when the mass of the distribution concentrates along the up-diagonal (Figure 3 HEX - G6PPer case) and a negative dependency when the mass is concentrated along the down-diagonal (Figure 3 HEX - PEPCK case). Thus, in HEX-PEPCK copula, we show that the PEPCK reaction operates when there is no glc__D_c available and does not operate when the latter is present. Thus, in their copula we observe a negative dependency between HEX and PEPCK. Contrary, in the HEX-G6PPer copula, we see a positive dependency between HEX, i.e., the reaction that consumes glc__D_c and G6PPer, that produces it. This leads to a biological paradox; ATP, a compound that "provides" energy to drive many processes in living cells, and glucose are consumed to produce G6P (HEX reaction) and at the same time, the G6P is consumed to produce glucose. This example highlights the caution required to study such models. Missing constraint types, especially thermodynamics, may lead to multiple non-vital scenarios for the cell. Approaches such as the one described by Saldida et al. [59] highlight the potential benefits of adding constraint-types. The last line of Figure 3 presents the reactions and their stoichiometry. The paradox described in Figure 3, where ATP is consumed pointlessly might occur due to numerous reasons, in our case the model does not require for an uptake flux for glucose, leading to a non-vital for the cell scenario. To consider further regulations that apply on the occurring processes in the cell, further constraints need to be included in the model (Section 2.1). However, addressing such challenges are out of the scope of this study.

6 Conclusions and future work

We propose a novel method for sampling that can sample from a convex polytope in a few thousands of dimensions within a day on modest hardware. In this way are able, for the first time, to perform accurate sampling from the latest human metabolic network, Recon3D.

Roundness progress in MMCS										
Phase	e_coli_core d=24		iJN746 d=116		iAT_PTL_636 d=289		iSDY_1059 d=509		Recon1 d=931	
	ESS	ψ	ESS	ψ	ESS	ψ	ESS	ψ	ESS	ψ
1st	50	1646.7	18	3511.0	197	1168.1	58	62504.2	682	1109.8
2nd	50	274.2	17	6218.2	950	20.3	76	811.7	509	21.5
3rd	40	210.5	18	2586.5	–	–	75	880.5	–	–
4th	88	69.2	750	3.7	–	–	79	702.3	–	–
5th	875	2.5	332	3.5	–	–	761	3.8	–	–

Table 4: The ESS and the roundness (given by the ratio $\psi = \sigma_{\max}/\sigma_{\min}$) for the runs of 5 models in Table 1. The run stops when our implementation detects that the total ESS exceeds 1000.

Regarding future work, parallelism could lead to a speedup in the run-time of our method as the algorithm is rather straightforward to parallelize. An additional improvement would be to exploit the sparsity of the stoichiometric matrix S and sample directly from the low dimensional polytope in $\mathbb{R}^n$ without projecting to a lower dimensional space.

Moreover, our method could be extended to any log-concave distribution restricted to the flux space and combined with bayesian metabolic flux analysis, to sample from multivariate, possibly multi-modal target distribution [28] addressing multiple challenges of the method from the biological point of view (e.g., unrealistic assumptions, uncertainty etc.). Last but not least, flux sampling in metabolic models built out from multiple metabolic networks, e.g., representing a microbial community, could also lead to important biological insights.

Finally, we can use our sampling method to perform numerical integration and volume computation of polytopes, or non-linear convex bodies like spectrahedra, e.g., [12, 42]. For this we should modify or adapt the current algorithms and implementations to benefit of our sampling strategy. This is an important and very interesting research direction.

Acknowledgements

The authors are grateful the anonymous reviewers of SoCG 2021 and JoCG for the various useful comments and suggestions that helped us to improve the presentation of our results.

References

[1] R. Adamczak, A. Litvak, A. Pajor, and N. Tomczak-Jaegermann. Quantitative estimates of the convergence of the empirical covariance matrix in log-concave ensembles.

Journal of the American Mathematical Society, 23(2):535–561, 2010.

[2] S. Artstein-Avidan, H. Kaplan, and M. Sharir. On radial isotropic position: Theory and algorithms, 2020.

[3] D. B. Bernstein, F. E. Dewhirst, and D. Segre. Metabolic network percolation quantifies biosynthetic capabilities across the human oral microbiome. *Elife*, 8:e39733, 2019.

[4] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, July 2004.

[5] S. Bordel, R. Agren, and J. Nielsen. Sampling the solution space in genome-scale metabolic networks reveals transcriptional regulation in key enzymes. *PLOS Computational Biology*, 6(7):1–13, 07 2010.

[6] S. P. Brooks and A. Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998.

[7] E. Brunk, S. Sahoo, D. C. Zielinski, A. Altunkaya, A. Dräger, N. Mih, F. Gatto, A. Nilsson, G. A. P. Gonzalez, M. K. Aurich, et al. Recon3D enables a three-dimensional view of gene variation in human metabolism. *Nature biotechnology*, 36(3):272, 2018.

[8] A. Cakmak, X. Qi, A. E. Cicek, I. Bederman, L. Henderson, M. Drumm, and G. Ozsoyoglu. A new metabolomics analysis technique: steady-state metabolic network dynamics analysis. *Journal of bioinformatics and computational biology*, 10(01):1240003, 2012.

[9] F. Cazals, A. Chevallier, and S. Pion. Improved polytope volume calculations based on hamiltonian monte carlo with boundary reflections and sweet arithmetics. *Journal of Computational Geometry*, page Vol. 13 No. 1 (2022), 2022.

[10] A. Chalkis, I. Z. Emiris, and V. Fisikopoulos. Practical volume estimation of zonotopes by a new annealing schedule for cooling convex bodies. In A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, and T. de Wolff, editors, *Mathematical Software – ICMS 2020*, pages 212–221, Cham, 2020. Springer International Publishing.

[11] A. Chalkis, I. Z. Emiris, and V. Fisikopoulos. A practical algorithm for volume estimation based on billiard trajectories and simulated annealing. *J. Experimental Algorithmics*, 2023. To appear.

[12] A. Chalkis, I. Z. Emiris, V. Fisikopoulos, P. Repouskos, and E. Tsigaridas. Efficient sampling in spectrahedra and volume approximation. *Linear Algebra and its Applications*, 648:205–232, 2022.

[13] A. Chalkis and V. Fisikopoulos. volesti: Volume approximation and sampling for convex polytopes in R, 2020. https://github.com/GeomScale/volume_approximation.

[14] A. Chalkis, V. Fisikopoulos, E. Tsigaridas, and H. Zafeiropoulos. Geometric Algorithms for Sampling the Flux Space of Metabolic Networks. In K. Buchin and E. Colin de Verdière, editors, *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages

21:1–21:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[15] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Fast MCMC Sampling Algorithms on Polytopes. *Journal of Machine Learning Research*, 19(55):1–86, 2018.

[16] A. Chevallier, S. Pion, and F. Cazals. Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations. Research Report RR-9222, INRIA Sophia Antipolis, France, 2018.

[17] B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2):133–160, 2016.

[18] A. B. Dieker and S. S. Vempala. Stochastic billiards for sampling from the boundary of a convex set. *Mathematics of Operations Research*, 40(4):888–901, 2015.

[19] S. Fallahi, H. J. Skaug, and G. Alendal. A comparison of Monte Carlo sampling methods for metabolic network models. *PLOS One*, 15(7):e0235393, 2020.

[20] A. M. Feist, M. J. Herrgård, I. Thiele, J. L. Reed, and B. Ø. Palsson. Reconstruction of biochemical networks in microorganisms. *Nature Reviews Microbiology*, 7(2):129–143, 2009.

[21] A. Gelman and D. B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472, 1992. Publisher: Institute of Mathematical Statistics.

[22] C. J. Geyer. Practical Markov Chain Monte Carlo. *Statist. Sci.*, 7(4):473–483, 11 1992.

[23] E. Gryazina and B. Polyak. Random sampling: Billiard walk algorithm. *European Journal of Operational Research*, 238(2):497 – 504, 2014.

[24] S. Gudmundsson and I. Thiele. Computationally efficient flux variability analysis. *BMC bioinformatics*, 11(1):1–3, 2010.

[25] G. Guennebaud, B. Jacob, et al. *Eigen v3*, 2010.

[26] H. S. Haraldsdóttir, B. Cousins, I. Thiele, R. M. Fleming, and S. Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11):1741–1743, 2017.

[27] H. S. Haraldsdóttir, B. Cousins, I. Thiele, R. M. Fleming, and S. Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11):1741–1743, 01 2017.

[28] M. Heinonen, M. Osmala, H. Mannerström, J. Wallenius, S. Kaski, J. Rousu, and H. Lähdesmäki. Bayesian metabolic flux analysis reveals intracellular flux couplings. *Bioinformatics*, 35(14):i548–i557, 2019.

[29] L. Heirendt, S. Arreckx, T. Pfau, S. N. Mendoza, A. Richelle, A. Heinken, H. S. Haraldsdóttir, J. Wachowiak, S. M. Keating, V. Vlasov, et al. Creation and analysis of biochemical constraint-based models using the cobra toolbox v. 3.0. *Nature protocols*, 14(3):639–702, 2019.

[30] H. A. Herrmann, B. C. Dyson, L. Vass, G. N. Johnson, and J.-M. Schwartz. Flux sampling is a powerful tool to study metabolism under changing environmental conditions. *NPJ systems biology and applications*, 5(1):1–8, 2019.

[31] G. H.Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[32] J. F. Jadebeck, A. Theorell, S. Leweke, and K. Noh. Hops: high-performance library for non uniform sampling of convex constrained models. *Bioinformatics*, 2020.

[33] F. John. Extremum Problems with Inequalities as Subsidiary Conditions. In G. Giorgi and T. H. Kjeldsen, editors, *Traces and Emergence of Nonlinear Programming*, pages 197–215. Springer, Basel, 2014.

[34] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407, 1996.

[35] A. T. Kalai and S. Vempala. Simulated annealing for convex optimization. *Mathematics of Operations Research*, 31(2):253–266, 2006.

[36] D. E. Kaufman and R. L. Smith. Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research*, 46(1):84–95, 1998.

[37] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. Ø.. Palsson, and N. E. Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2016.

[38] A. Laddha and S. Vempala. Convergence of Gibbs Sampling: Coordinate Hit-and-Run Mixes Fast, 2020.

[39] Y. T. Lee and S. S. Vempala. Convergence rate of riemannian hamiltonian monte carlo and faster polytope volume computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 1115–1121, New York, NY, USA, 2018. Association for Computing Machinery.

[40] N. E. Lewis, H. Nagarajan, and B. O. Palsson. Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. *Nature Reviews Microbiology*, 10(4):291–305, 2012.

[41] L. Lovász, R. Kannan, and M. Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997.

[42] L. Lovász and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 57–68. IEEE, 2006.

[43] L. Lovász and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithms. *J. Computer & System Sciences*, 72:392–417, 2006.

[44] M. Lularevic, A. J. Racher, C. Jaques, and A. Kiparissides. Improving the accuracy of flux balance analysis through the implementation of carbon availability constraints for intracellular reactions. *Biotechnology and bioengineering*, 116(9):2339–2352, 2019.

[45] M. MacGillivray, A. Ko, E. Gruber, M. Sawyer, E. Almaas, and A. Holder. Robust analysis of fluxes in genome-scale metabolic pathways. *Scientific Reports*, 7, 12 2017.

[46] D. Machado, S. Andrejev, M. Tramontano, and K. R. Patil. Fast automated reconstruction of genome-scale metabolic models for microbial species and communities. *Nucleic acids research*, 46(15):7542–7553, 2018.

[47] J. Maurer and S. Watanabe. Boost random number library. Software, 2017.

[48] MOSEK ApS. *The MOSEK optimization toolbox for R manual. Version 9.2.*, 2019.

[49] H. Narayanan and P. Srivastava. On the mixing time of coordinate hit-and-run, 2020.

[50] R. B. Nelsen. *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[51] A. Noronha, J. Modamio, Y. Jarosz, E. Guerard, N. Sompairac, G. Preciat, A. D. Daníelsdóttir, M. Krecke, D. Merten, H. S. Haraldsdóttir, A. Heinken, L. Heirendt, S. Magnúsdóttir, D. A. Ravcheev, S. Sahoo, P. Gawron, L. Friscioni, B. Garcia, M. Prendergast, A. Puente, M. Rodrigues, A. Roy, M. Rouquaya, L. Wiltgen, A. Žagare, E. John, M. Krueger, I. Kuperstein, A. Zinovyev, R. Schneider, R. M. T. Fleming, and I. Thiele. The Virtual Metabolic Human database: integrating human and gut microbiome metabolism with nutrition and disease. *Nucleic Acids Research*, 47(D1):D614–D624, 10 2018.

[52] J. D. Orth, I. Thiele, and B. Ø.. Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.

[53] B. Ø.. Palsson. Metabolic systems biology. *FEBS letters*, 583(24):3900–3904, 2009.

[54] B. Ø.. Palsson. *Systems biology*. Cambridge university press, 2015.

[55] O. Perez-Garcia, G. Lear, and N. Singhal. Metabolic network modeling of microbial interactions in natural and engineered environmental systems. *Frontiers in microbiology*, 7:673, 2016.

[56] J. L. Reed. Shrinking the metabolic solution space using experimental datasets. 2012.

[57] V. Roy. Convergence Diagnostics for Markov Chain Monte Carlo. *Annual Review of Statistics and Its Application*, 7(1):387–412, 2020.

[58] P. A. Saa and L. K. Nielsen. ll-ACHRB: a scalable algorithm for sampling the feasible solution space of metabolic networks. *Bioinform.*, 32(15):2330–2337, 2016.

[59] J. Saldida, A. P. Muntoni, D. de Martino, G. Hubmann, B. Niebel, A. M. Schmidt, A. Braunstein, A. Milias-Argeits, and M. Heinemann. Unbiased metabolic flux inference through combined thermodynamic and 13c flux analysis. *bioRxiv*, 2020.

[60] J. Schellenberger and B. Ø.. Palsson. Use of randomized sampling for analysis of metabolic networks. *Journal of biological chemistry*, 284(9):5457–5461, 2009.

[61] J. R. Schramski, A. I. Dell, J. M. Grady, R. M. Sibly, and J. H. Brown. Metabolic theory predicts whole-ecosystem properties. *Proceedings of the National Academy of Sciences*, 112(8):2617–2622, 2015.

[62] S. S. Shishvan, A. Vigliotti, and V. S. Deshpande. The homeostatic ensemble for cells. *Biomechanics and Modeling in Mechanobiology*, 17(6):1631–1662, 2018.

[63] R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[64] R. W. Smith, R. P. van Rosmalen, V. A. M. Dos Santos, and C. Fleck. Dmpy: a python package for automated mathematical model construction of large-scale metabolic systems. *BMC systems biology*, 12(1):1–16, 2018.

[65] N. J. Stanford, T. Lubitz, K. Smallbone, E. Klipp, P. Mendes, and W. Liebermeister. Systematic construction of kinetic models from genome-scale metabolic networks. *PloS one*, 8(11):e79195, 2013.

[66] N. Swainston, K. Smallbone, H. Hefzi, P. D. Dobson, J. Brewer, M. Hanscho, D. C. Zielinski, K. S. Ang, N. J. Gardiner, J. M. Gutierrez, S. Kyriakopoulos, M. Lakshmanan, S. Li, J. K. Liu, V. S. Martínez, C. A. Orellana, L.-E. Quek, A. Thomas, J. Zanghellini, N. Borth, D.-Y. Lee, L. K. Nielsen, D. B. Kell, N. E. Lewis, and P. Mendes. Recon 2.2: from reconstruction to model of human metabolism. *Metabolomics*, 12(7):109, June 2016.

[67] D. Vats and C. Knudson. Revisiting the gelman-rubin diagnostic, 2020.

[68] F. Zorrilla, F. Buric, K. R. Patil, and A. Zelezniak. metagem: reconstruction of genome scale metabolic models directly from metagenomes. *Nucleic acids research*, 49(21):e126–e126, 2021.