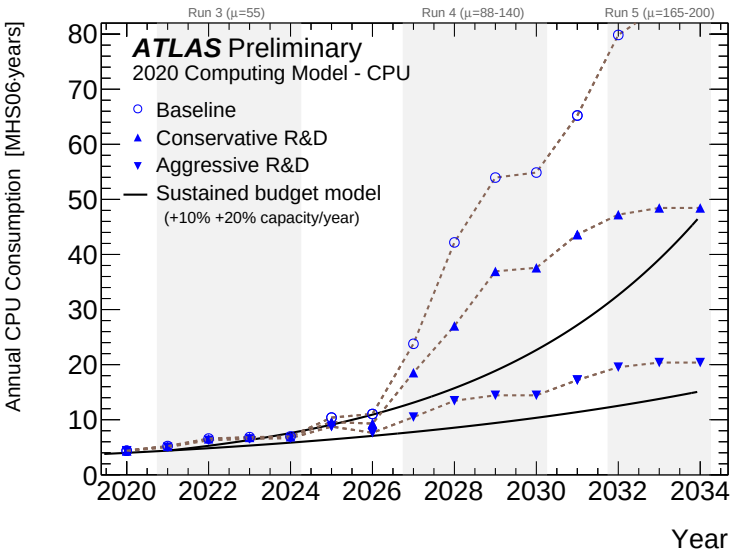
Computing challenges for the HL-LHC

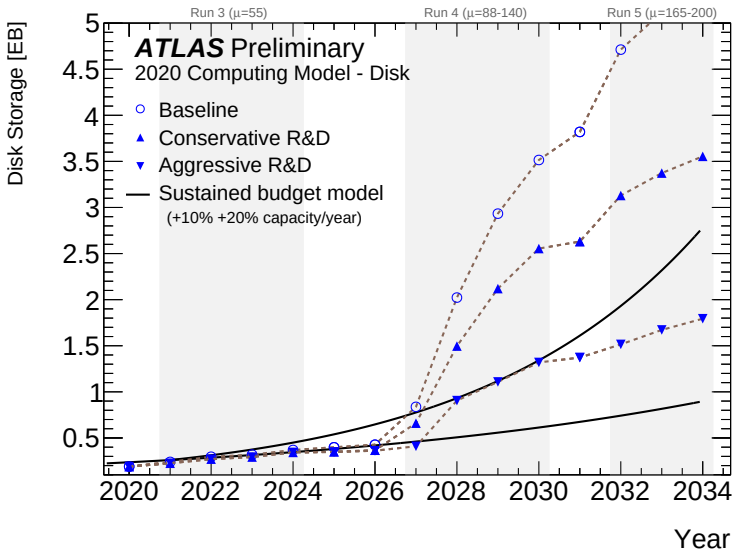
arXiv:2004.13687*
HSF Physics Event Generator WG

Marek Schönherr
IPPP, Durham University

*edited by A. Valassi, E. Yazgan, J. McFayden

Introduction





- HL-LHC computing needs well above expected technology evolution
- main challenges are computing and storage
- resources provided through WLCG infrastructure, but also other HPC
- report focusses on the MC event generator component

Introduction

Current situation:

- currently about 20% of computing budget for MC event generation
- around $15 \cdot 10^9$ MC events/a (3x data events)

Situation for HL-LHC:

- need for more advanced/complex MC (higher multiplicity, higher orders, etc.)
  $\rightarrow$ event generation will become more expensive
- MC limitations not priced in in HL-LHC studies as major source of uncertainty
  $\rightarrow$ limits physics potential of HL-LHC

$\Rightarrow$ **speedups in MCs will need to be addressed**

Collaborative challenges

Diverse software landscape:

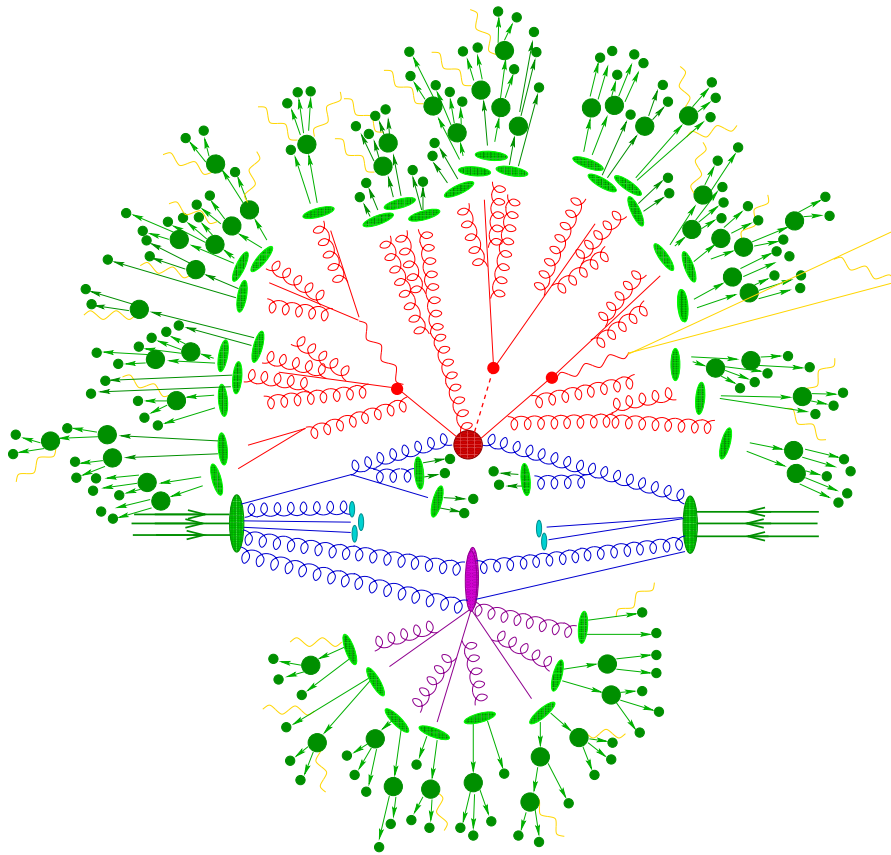
- SHERPA, MG5_aMC, POWHEG, PYTHIA, HERWIG, ALPGEN, etc.
- different orders LO, NLO, NNLO, different showers, different NP models
- **every MC event generator essentially a prototype**

Diverse human environment:

- broad spectrum of skills needed for development and support of MC: theorists, experimentalists in research & computing, software engineers and system performance specialists
- training: theo. & exp. lack training in software development
  software eng. & exp. lack training in theoretical physics
- career: software development not recognised for theorists,
  generator support not for experimentalists

Technical challenges

- hard scattering
  - (mostly) constant program flow
  - evaluation of very large expressions
  - large memory requirements
    few GB for e.g. $V + 2j$ @ NLO
  - challenge: generate good weight distribution in finite time
- parton showers
  multiple interactions
  hadronisation
  hadron decays
  - mostly Markov Chain techniques
  - highly variable program flow
  - generally small expressions
  - low memory requirements



Technical challenges

Matrix elements

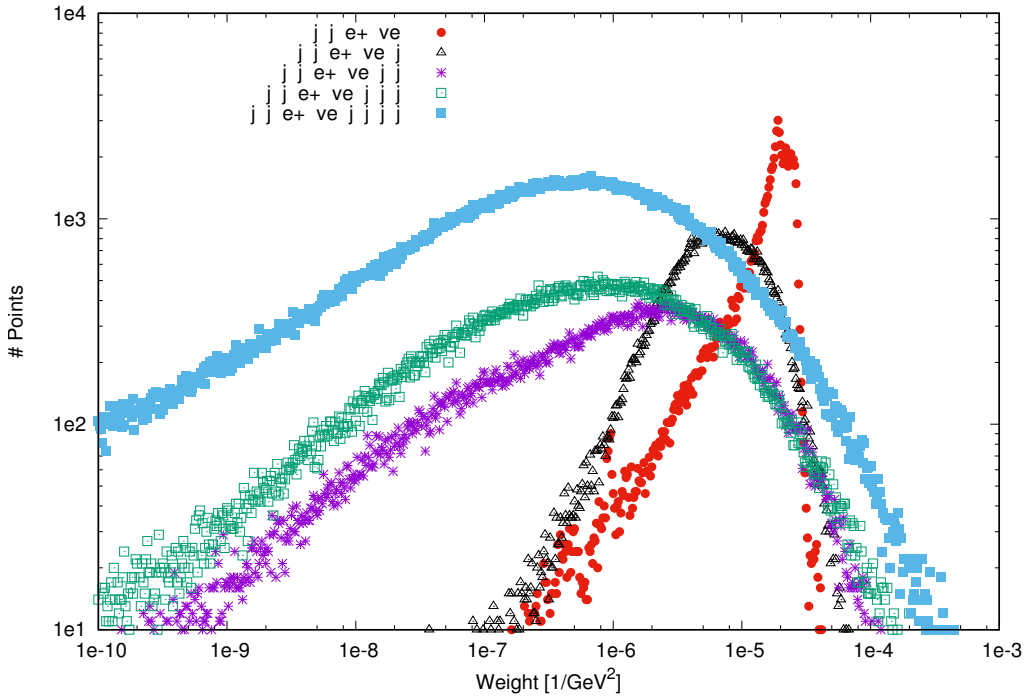
- ideal weight distribution: uniform weights
- when all events contribute the same to any given observable, the fewest events are needed to reach a given statistical power of a sample
  $\Rightarrow$ **unweighted events**
- integrate $f(\mathbf{x})$ by drawing point with sampling function $g(\mathbf{x})$, assign weight $w(\mathbf{x}) = f(\mathbf{x})/g(\mathbf{x})$ and hit-or-miss against w_{max}

Parton showers and non-perturbative event phases

- generally set up to generate unweighted distributions through probabilistic implementations

Unweighted event generation

LO weight distribution



Unweighting

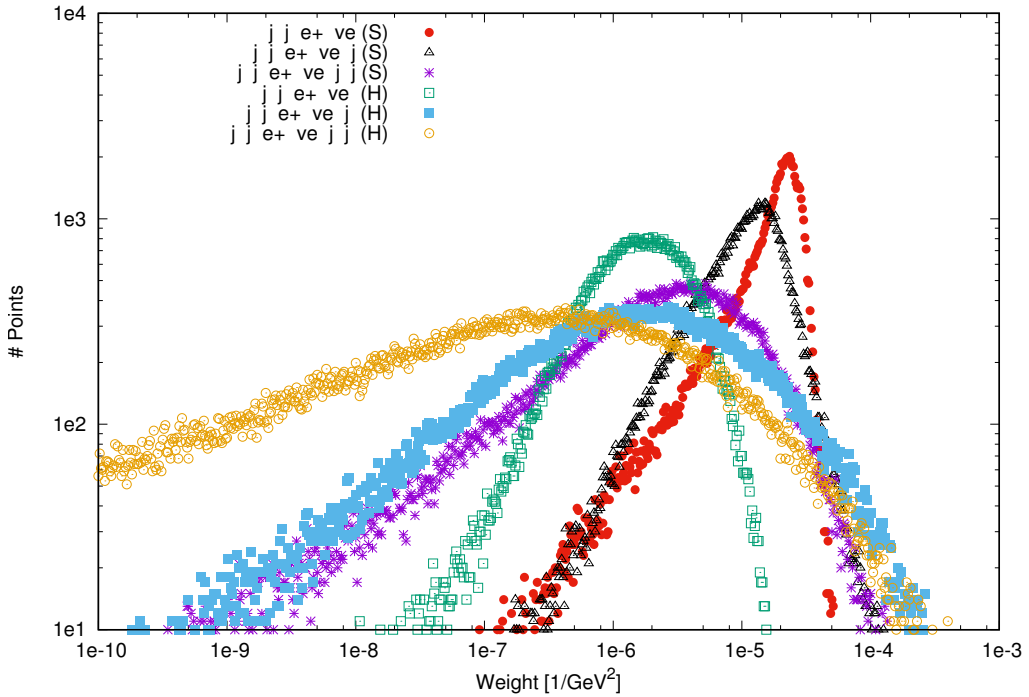
- hit-and-miss against maximum of weight distribution, $w_{\max}$
- if $w_{\max}$ is artificially lowered (or event beyond $w_{\max}$ encountered)
  evts must acquire rel. weight wrt. $w_{\max}$

- higher multiplicity $\rightarrow$ wider weight distribution
  $\Rightarrow$ worse unweighting efficiency
- bad unweighting efficieny at NLO dominated by H events

$\Rightarrow$ **performance limited by unweighting efficiency**

Unweighted event generation

NLO weight distribution



Unweighting

- hit-and-miss against maximum of weight distribution, $w_{\max}$
- if $w_{\max}$ is artificially lowered (or event beyond $w_{\max}$ encountered)
  evts must acquire rel. weight wrt. $w_{\max}$

- higher multiplicity $\rightarrow$ wider weight distribution
  $\Rightarrow$ worse unweighting efficiency
- bad unweighting efficieny at NLO dominated by H events

$\Rightarrow$ **performance limited by unweighting efficiency**

Sources for degredation of statistical power

Negative weights

- NLO matching methods (to var. degree) introduce neg. weights

$$N_{\text{eff}} = N\,(1-2f)^2$$

- typically arise as correction for previous overestimate

$\Rightarrow$ **neg. weighted events must be kept to minimum**

Parton shower weights

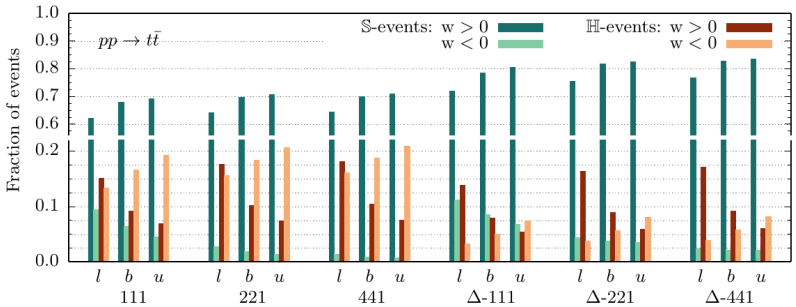
- parton showers operate at LO and leading colour, such that all splitting functions positive definite
- corrections beyond introduce weights
  $\rightarrow$ spoil hard-won uniform weights of unweighted ME configurations
  first ideas for improvements, eg. resampling

Olson et.al arXiv:1912.02436

Negative weight fraction

Efforts in MG5_aMC and SHERPA

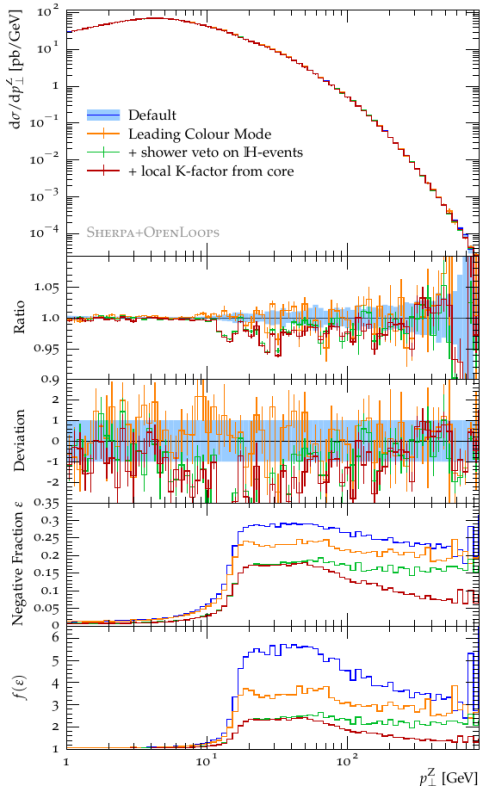
arXiv:2002.12716



- combination of folding (no impact on dists), Sudakov form factors and multiple shower starting scales on H-events (impact on dists)

CERN-THESIS-2020-024

	Negative Weight Fraction
Default	18.1%
Leading Colour Mode	14.0%
+ shower veto on ℍ-events	9.6%
+ local K-factor from core	9.1%



Phase space biasing

Experiments are often interested in specific regions of phase space (flavour, momenta, etc.)

- high-p_{T} tails
- specific jet flavours
- specific hadron decay chains

For the former, slicing and biasing are methods of choice but come at costs:

- slicing: artefacts around slicing values from spill-overs
- biasing: lowered unweighting efficiency due to modified integrand structure

For the latter, whether an event is in a specific phase space is often only known at the end of event generation.
The use of post-generation event filters can be highly inefficient.

Modernisation of software

All event generators are old software projects with decades old code.

- constructed with different design paradigms in mind, generally do not fit current hardware design of HPCs
- data parallelism, vectorisation, GPU for parts w/ const. program flow task-parallelism, multithreading can reduce memory/core
  $\rightarrow$ necessitate a ground level rewrite of core code
- all Markov-chain MC components not suited (non-constant programm flow)
  $\rightarrow$ best trivially parallelised
- may necessitate chopping apart event generators into units for different architectures
  $\rightarrow$ necessitates strict factorisation of event generation phases, prohibits cross talk needed for higher accuracy

Cross-experiment collaboration

- Perturbative parts of event generation (at least the matrix elements) have very little degree of freedom
  $\rightarrow$ collaboration between experiments (at least ATLAS & CMS) for largest MC samples (V+jets, $t\bar{t}$+jets, etc) gives trivial factor 2 saving
- each experiment would do their own shower and non-perturbative modelling with their favourite tune
- needs coordination and supra-experimental infrastructure

Conclusions

- **matrix elements**
  unweighting is the bottleneck
  - high complexity of the integrand (large memory requirements, interferences lead to complex structure not fully captured by employed techniques)
  - poor understanding of its structure (unweighting efficiency) (machine learning currently does not seem to be the answer)
  - reduce negative weight fraction (mostly from matching)
- **parton showers & non-perturbative modeling**
  mostly fine (small part of the overall computational budget)
  - may need attention when improvements lead to large weight spread
  - filtering needs improvements, ideally built-in biasing tools
- future NNLO accuracy will add larger terms to evaluate, more complicated basic functions (polylogs, etc) and negative weights

Conclusions

What needs to be done?

- gain a detailed understanding of current CPU costs
- survey generator codes on how to move to GPUs and vectorised code
  $\rightarrow$ software development
- optimise adaptive sampling algorithms ($g(\mathbf{x})$)
  $\rightarrow$ physics understanding
- negative weight reduction
  $\rightarrow$ physics understanding
- improve training and career opportunities in generator development

Connected activities in the UK within the ECHEP (Efficient Computing in HEP) and other national programs

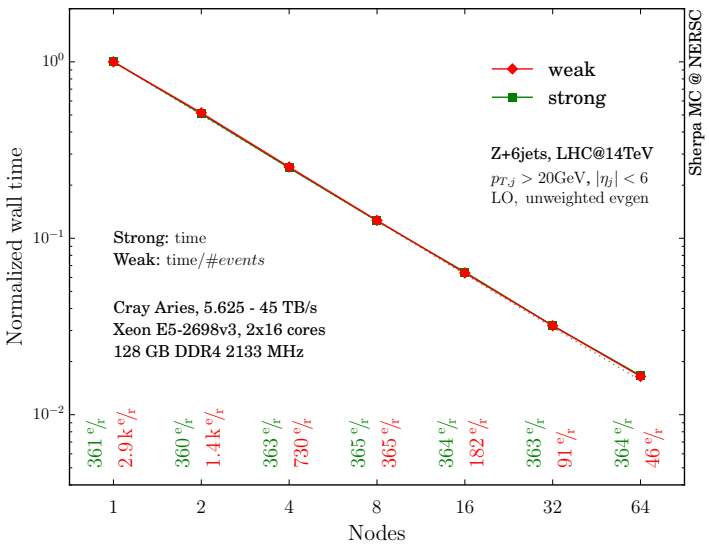
Backup

V plus multijet production

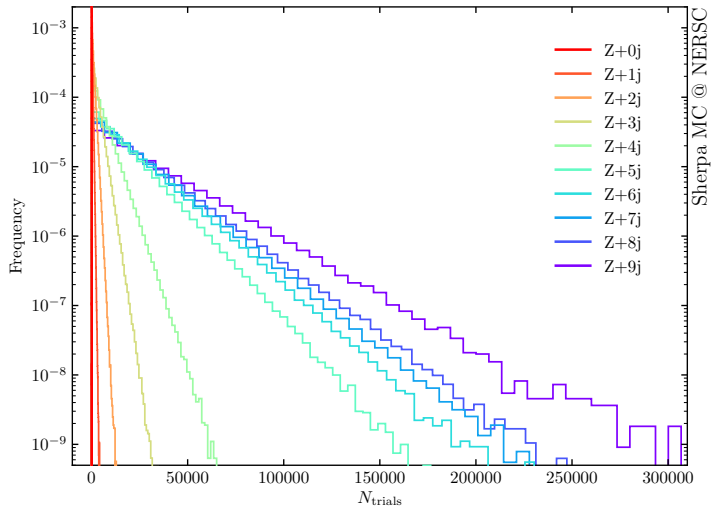
SHERPA+PYTHIA

$$pp \to V + 0, \ldots, 9\,\text{jets}$$

- generate unweighted events on large clusters
- memory is bottleneck



MPI scaling up to 64x32 cores



number of trials in unweighting

- write HDF5 files (MPI comp.)
- process w/ parton shower, large memory requirements for creating CKKW histories

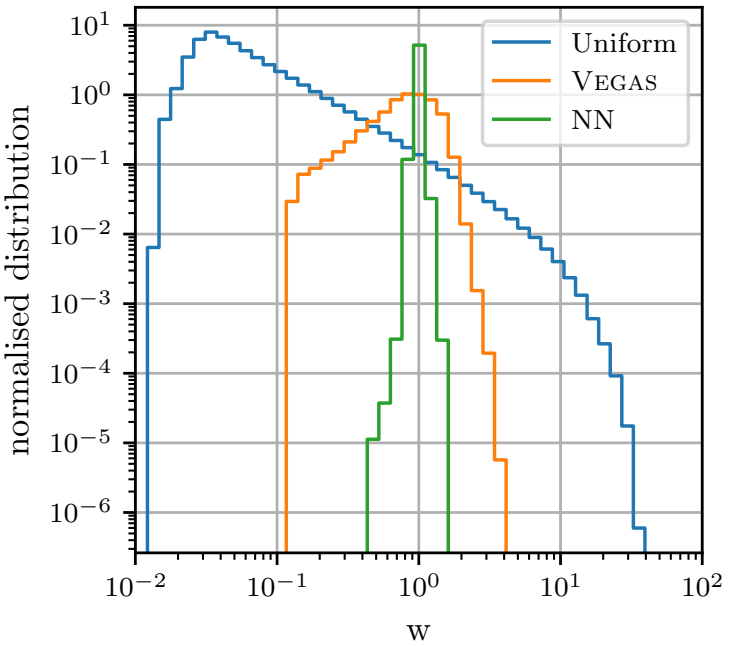
Improvements throught machine learning

Phase space sampling / unweighting

- replace old-school 1D machine learning algoirthms (VEGAS) by multivariate / NN techniques
- many tries, promising on toy examples

  Bendavid arXiv:1707.00028
  Klimek, Perelstein arXiv:1810.11509
  Otten et.al arXiv:1901.00875
  ...
- realistic implementation in working event generator

  Bothmann et.al arXiv:2001.05478
  Gao et.al arXiv:2001.10028



$gg \to 3g$

improvements only for cases that need no improvements, no better than VEGAS for bottleneck cases

not yet smart enough?

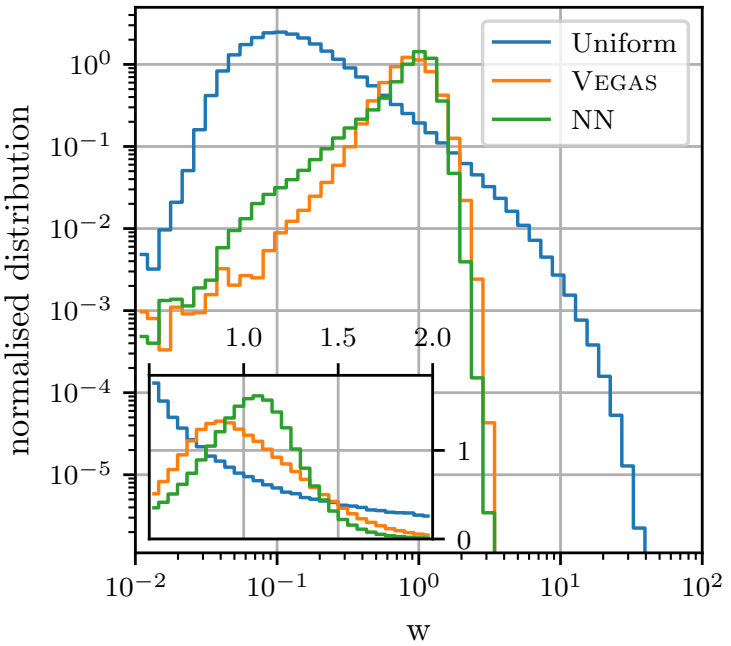
Improvements throught machine learning

Phase space sampling / unweighting

- replace old-school 1D machine learning algoirthms (VEGAS) by multivariate / NN techniques
- many tries, promising on toy examples

  Bendavid arXiv:1707.00028
  Klimek, Perelstein arXiv:1810.11509
  Otten et.al arXiv:1901.00875
  ...
- realistic implementation in working event generator

  Bothmann et.al arXiv:2001.05478
  Gao et.al arXiv:2001.10028



$gg \to 4g$

improvements only for cases that need no improvements, no better than VEGAS for bottleneck cases

not yet smart enough?