Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

Time Frames from Simulation

Kolja Kauder

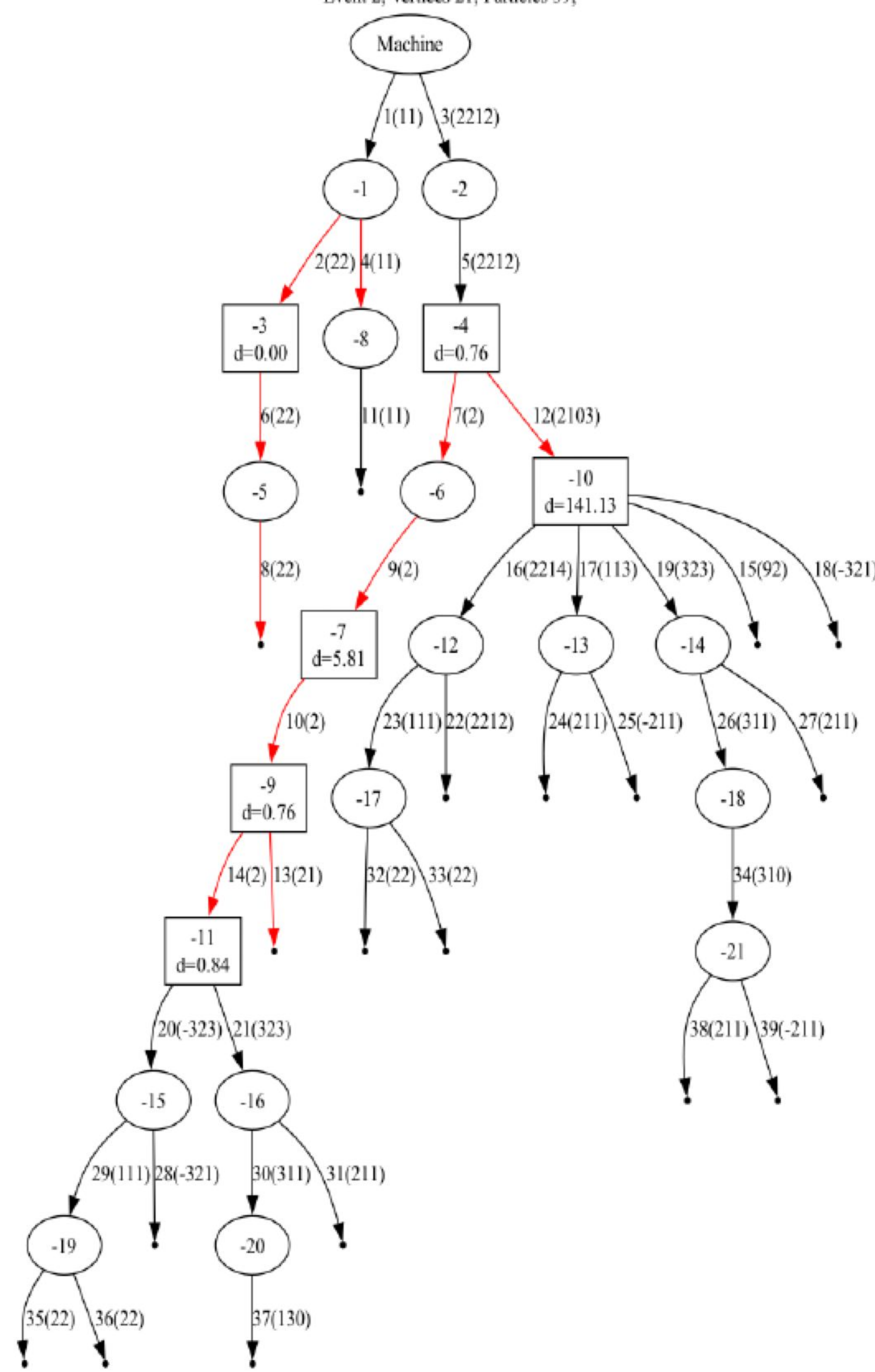
Joint ePIC SRO and Electronics & DAQ Meeting

March, 21 2023

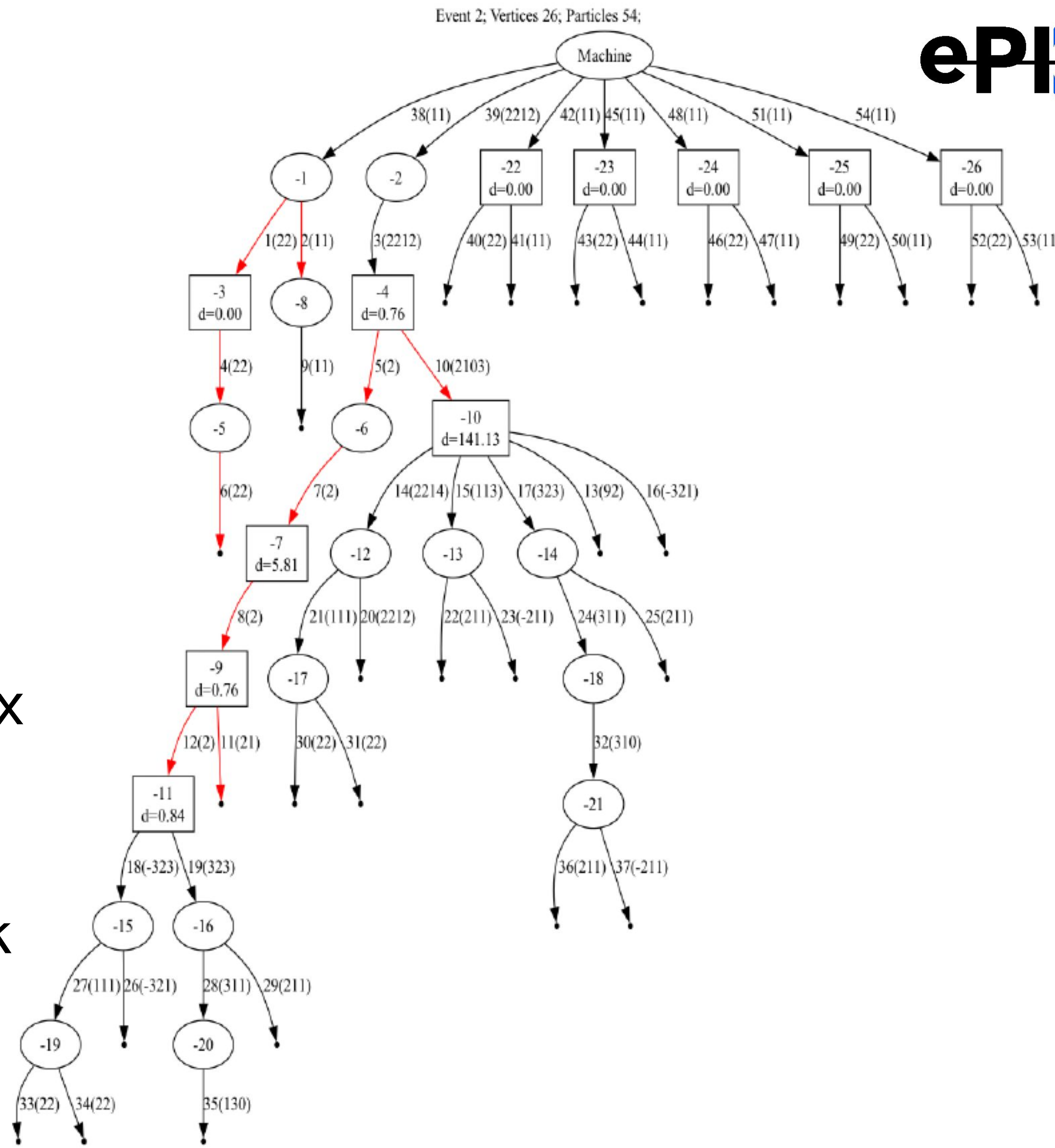
HepMC structure

- Directed Acyclical (topologically sorted) Graph
- `GenVertex` holds time and position
  - status could be used to distinguish background types - needs EDM change
- `GenParticle` holds momentum, PID, …

Ignoring the "Machine" node, a collection of DAGs → just add more!



Combining Events



Same pythia event, with five added e+gas events → **time slice**

All vertices have their own time

Machine node isn't a true `GenVertex` and shouldn't be used for merging

- but can be used to shift the whole slice to an absolute machine clock time

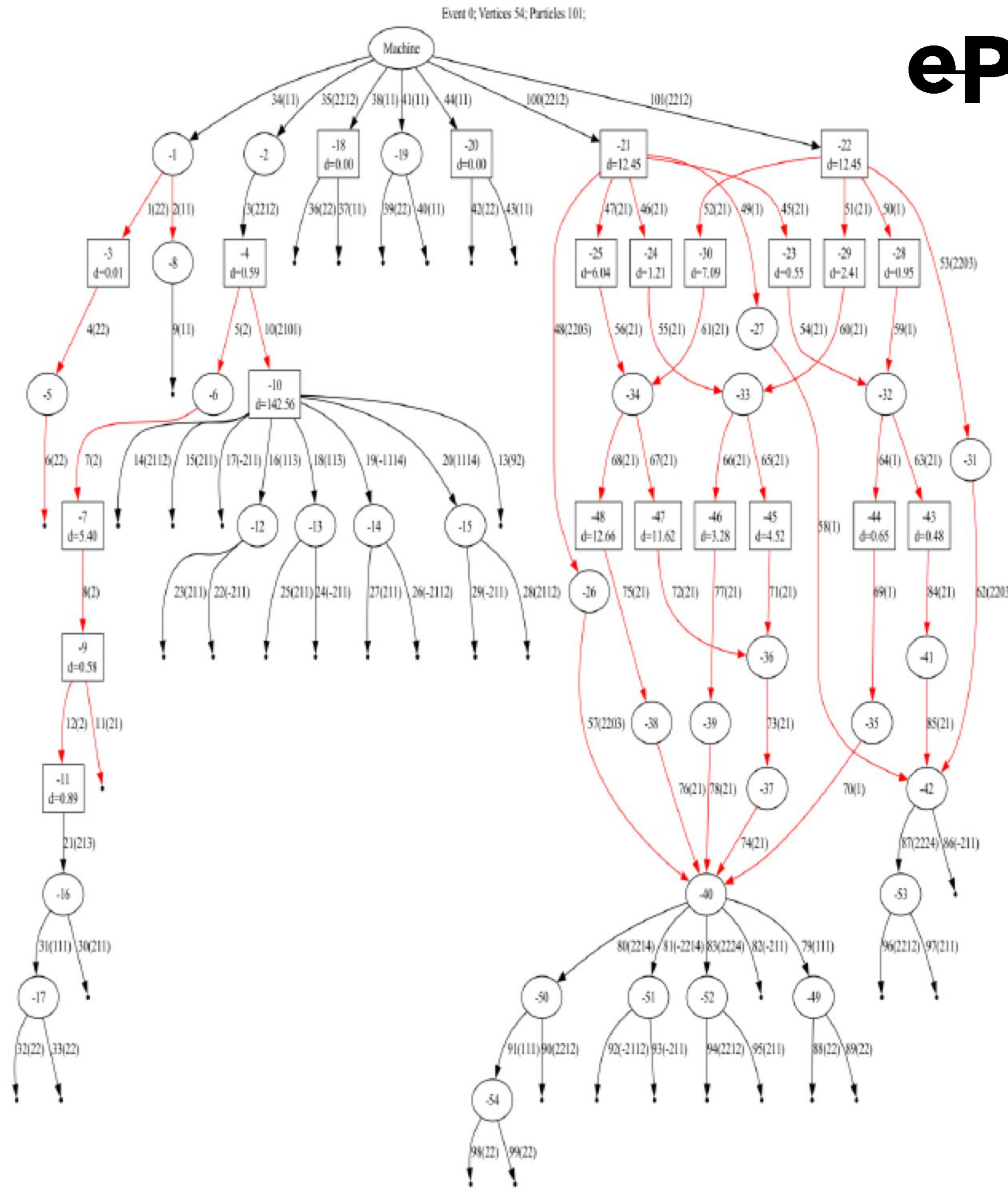
More Complexity

Event with e+gas and p+gas

- p+gas == pythia event → add more and more signal and BG events for a longer time slice
- Not showing SR admixture, O(5000) additional particles in 2 μs
- These are huge files, 17 GB ASCII for 10k events

→ Consider cuts

→ HepMC3 is a library, less wasteful formats exist



Development Status

Now fully ported to C++ (Not yet merged)

+ Output seems sane, consistent with python version - more testers are welcome!
+ Support all formats (hepmc.root, hepmc.gz, …)
+ (Marginally) faster
+ Consistent memory usage (python: 2GB - 6GB for the same input depending on the time of day or phase of the moon)
- Readability
- Conciseness (`rng.choice(a=events,size=nEvents,p=probs, replace=False` is a powerful one-liner, hard to replicate in C++)
- Less native connection to npsim

Note: Found a minor bug in the code in the process → if you want to keep using the python version, ping us to backport the fix!

Usage

- README and wiki are **not** (yet) up-to-date
- However, "-h" output is comprehensive
  - Shoutout to p-ranav's argparse for C++
  - … and to Copilot for doing a lot of boilerplate conversion

```
airbox:~/deveic/HEPMC_Merger/build % ./SignalBackgroundMerger -h
Usage: Merge signal events with up to three background sources. [--help] [--version] [--signalFile VAR] [--signalFreq VAR] [--bg1File VAR] [--bg
1Freq VAR] [--bg2File VAR] [--bg2Freq VAR] [--bg3File VAR] [--bg3Freq VAR] [--outputFile VAR] [--rootFormat] [--intWindow VAR] [--nSlices VAR] [
--squashTime] [--rngSeed VAR] [--verbose]

Optional arguments:
  -h, --help         shows help message and exits
  -v, --version      prints version information and exits
  -i, --signalFile   Name of the HEPMC file with the signal events [nargs=0..1] [default: "small_ep_noradcor.10x100_q2_10_100_run001.hepmc"]
  -sf, --signalFreq  Signal frequency in kHz. Default is 0 to have exactly one signal event per slice. Set to the estimated DIS rate to randomiz
e. [nargs=0..1] [default: 0]
  -bg1, --bg1File    Name of the first HEPMC file with background events [nargs=0..1] [default: "small_hgas_100GeV_HiAc_25mrad.Asciiv3.hepmc"]
  -bf1, --bg1Freq    First background frequency in kHz. Default is the estimated hadron gas rate at 10x100. Set to 0 to use the weights in the c
orresponding input file. [nargs=0..1] [default: 342.8]
```

Details from the python version in the Backup, identical for C++

→ **except freq is now the true frequency in kHz from the wiki, not its inverse**

Behind the Scenes

Current Sources of Background:

- Beam gas (e/p; FXT), 6+ events per 2 μs time slice (integration window)
- Synchrotron radiation (from the project), <photons/ts> ~ 5000

Math:

- Draw from Poisson distributions to determine how many BG events to inject
- Distribute uniformly through the time slice
- By default, use exactly one signal event per slice, but can instead be set to be the same as above

Sidebar:

- Equivalent to drawing time steps from exponential distribution until time slice is exhausted; using both for historical reasons

To Do:

- Skip events, for batch processing (trivial)
- Bethe Heitler Bremsstrahlung for the lumi detector and low-Q^2 tagger: Need to correlate background time to bunch crossing
- Consider merging after Geant4

Potential improvements:

- Refactor to unify Poisson/Exponential use
- Investigate surprisingly large memory footprint (3.8GB)
- Speed-up. There's an I/O bound but some random numbers could be used more cleverly

However, all of the above issues are almost completely caused by the special treatment of SR →very soon to be obsolete thanks to Andrii Natochii!

Beyond HepMC

- AFAIK, MAPS run continually, integration time is a natural split point for larger slices, ideally before digitization
  - needs to be in EICrecon
  - Edge effects?
  - I'm told the correlation against the RHIC/EIC clock is very non-trivial
- Digitization: EICrecon digi hits integrates over the entire event presented to it; faster detector need to instead generate new hits

```
// There is previous values in the cell
auto& hit = cell_hit_map[sim_hit.getCellID()];

// keep earliest time for hit
auto time_stamp = hit.getTimeStamp();
hit.setTimeStamp(std::min(hit_time_stamp, hit.getTimeStamp()));

// sum deposited energy
auto charge = hit.getCharge();
hit.setCharge(charge + (std::int32_t) std::llround(sim_hit.getEDep() * 1e6));
```

Supplementary slides

Usage for Signal

```
% python3 ./signal_background_merger.py --help

Merge signal events with up to three background sources.

options:
  -i SIGNALFILE, --signalFile SIGNALFILE
                        Name of the HEPMC file with the signal events


  -sf SIGNALFREQ, --signalFreq SIGNALFREQ
                        Poisson-mu of the signal frequency in ns.
                        Default is 0 to have exactly one signal event per slice.
                        Set to the estimated DIS rate to randomize.
```

Single particles, PYTHIA, …

Option: DIS (or so) **freq.** from the Wiki

Default: Exactly **one event** in every time slice.
Could add: At least one event / slice (to exclude pure BG)

Poisson determines **how many** events in a slice. **"Position"** in the slice is **uniformly random**

- It's possible this should be 0, or the mid-point, depending on how the DAQ "triggers"

Usage for FXT BG

```
options:
  -bg1 BG1FILE, --bg1File BG1FILE
                Name of the first HEPMC file with background events
  -bf1 BG1FREQ, --bg1Freq BG1FREQ
                Poisson-mu of the first background frequency in ns. Default is
                the estimated hadron gas rate at 10x100. (Set to 0 to use the
                weights in the corresponding input file)
```

Ex.: h-gas

From the Wiki

See SR slide

- **Poisson** determines **how many** events in a slice. **"Position"** in the slice is **uniformly random**
- Same options, same meaning for `-bg2, -bf2` (Ex.: e-gas)
- Input files **"roll over"** when the end is reached. This could lead to artifacts. Randomizing (i.e. jumping around in the HepMC file) is very inefficient but possible. Better to generate a large enough background pool (though that's a lot of disk space).
  - Better yet to generate events on the fly

Usage for SR BG

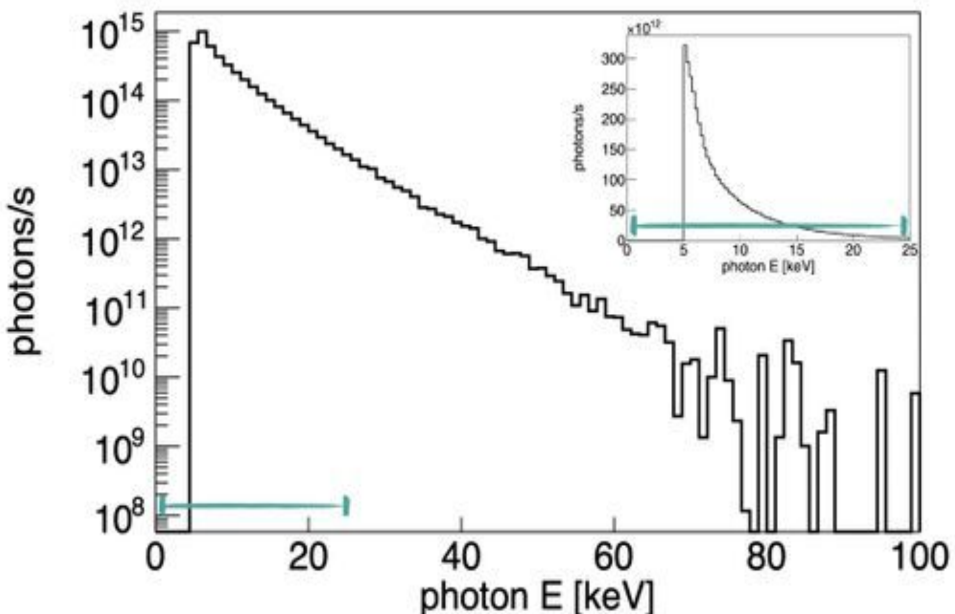
```
options:
  -bg3 BG3FILE, --bg3File BG3FILE
                Name of the third HEPMC file with background
                events
  -bf3 BG3FREQ, --bg3Freq BG3FREQ
                Poisson-mu of the third background frequency in
                ns. Default is 0 to use the weights in the
                corresponding input file. (Set to a value >0 to
                specify a poisson mu instead.)
```

Ex.: Synchrotron Radiation

From SynRad

- **Poisson** determines **how many** events in a slice. **"Position"** in the slice is **uniformly random**
- Details deserve their own slide

Synchrotron Radiation details



Spectrum from SynRad.
Important: Internally, this "histogram" is a lookup table for 1.8M individual SR photons

- Each photon in SynRad's output comes with its own **rate R_p**
- Use the **average rate $\langle R_p \rangle$** as **μ** in a Poisson distribution to determine the **number N of SR photons** in a given slice
- Using **rate as weight**, draw N individual photons from the spectrum and place them uniformly

- Weighted draw means the entire list needs to be in memory