This is an overview of a possible approach to proof generation, inspired by Otti, seethis paper.

At its core, Entropy, similar to CirclesUBI, wants to solve the max-flow problem on a directed graph. It currently does so via the Edmonds-Karp algorithm which has complexity O(VE^2)

for V

vertices and E

edges. It is essential to prove that the solution we find is correct. The most straight-forward solution may seem to be to use a prover like RISC0 to prove that our algorithm ran correctly on some verifiably correct input. However, proving execution is expensive and not necessary for our case.

A feature of the max-flow problem we exploit is that it can be expressed as a linear-programming (LP) problem. LP is a special case of constraint satisfaction where constraints are linear. In this case, the problem becomes one of maximizing flow in the graph with respect to flow constraints on the edges. This is dual to the min-cut problem.

This formulation is interesting because it opens up the possibility for solution verification using linear programming separately from solving via a conventional algorithm, eliminating the need to prove execution. Otti is a compiler for proofs of optimization problems. For LP problems, it uses the strong-duality theorem of optimization, and proves that strong duality holds between solutions of the primal and dual LP formulation.

For Entropy, the max-flow problem would need to be solved using a fast graph algorithm and formulated as an LP problem for the purposes of proof generation.

There are many candidate fast algorithms that provide better complexity than Edmonds-Karp, seethis repo for example. There is also a proposal that would significantly reduce the number of edges of the graph over which max-flow is computed. These two factors should significantly speed up solving.

One draw-back of our approach is that it requires solving the dual problem as well, so that the strong-duality condition may be checked. One could use the value of the min-cut (=max-flow) to find a min-cut faster than starting from scratch.

Otti does the job of formulating the primal and dual LP problem as well as the requirements of the strong duality condition in terms of R1CS circuits for the proof. Witnesses for the proof will have to be provided in the correct form derived from the output of the graph algorithm. Note that once the maxflow problem is solved, we can conveniently forget about edges that have zero flow and prove facts only about the subgraph with non-zero flow.

There are broadly two directions of work here:

1. As mentioned earlier, the proof and witnesses are generated separately. R1CS witness generation from the solution produced by the graph algorithm will be required. The Otti workflow (repo) will have to be modified to work with this setup.

2. Hook-up other prover backends that support R1CS (e.g. Marlin) to compare performance/compile to vampir to have more provers to benchmark (the latter might be preferred for future Taiga integration plans). As is visible in the results of the Otti paper, the bottleneck is proving time, so reducing this should be prioritized.

A quick, bad sketch of the workflow:

[

Screenshot from 2023-11-23 01-50-06

944×526 28.5 KB

](https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/88b722a801e6117590dc6be44badd45b14eb2f51.png)