ENEE759U - CAD of Digital Circuit

Final Project

Automated ILP Scheduling

Submitted by: Mumtahina Islam Sukanya (UID 120205538)

Github Repo Link: https://github.com/imumtahina/AutomatedILPScheduler

- **Introduction**

The objective of this project is to develop an automated approach to efficiently schedule computational tasks represented as a Data Flow Graph. The nodes from DFGs are given in edgelist format.

DFG is a graphical representation where nodes represent operations and edges represent data dependencies between them. Scheduling these operations optimally is critical to maximizing hardware utilization, minimizing execution time, and reducing overall resource consumption.

In the edgelist files, the edges and their respective weights are provided. The weights here represent the number of registers used in each edge data transfer. The automated script processes the given DFG first and evaluates design specification from the given arguments. After that, it produces the ILP file and checks for infeasibility before running the ILP solver which was glpk in this case. Finally, the output from the ILP solver is displayed back on the terminal.

This scheduler can run in three modes based on provided arguments.

1) minimize memory under latency L where L is an input argument (MM-LC)
   Execution example: python3 auto.py -g rand_DFG_s10_1.edgelist -l 7
2) minimize latency under memory M where M is an input argument (ML-MC)
   Execution example: python3 auto.py -g rand_DFG_s10_1.edgelist -m 20
3) perform latency-memory Pareto-optimal
   Execution example: python3 auto.py -g rand_DFG_s10_1.edgelist -l 5 -m 20

The L input argument puts a latency constraint on the schedule which dictates the maximum timestamp limit for the scheduling. The M argument is a memory constraint on the schedule which dictates the maximum number of registers the schedule can utilize at a time. When both arguments are provided, the scheduler optimizes both latency and memory with respect to each other.

- **Implementation details**

The entire implementation of the project was done is four distinct steps.

Step 1: Pre-processing the graph representation

- Read edgelist representation of graph using networkx
- Determine all the nodes and associated edges
- Parse memory costs/weights from the graph edgelist input
- Perform ASAP and ALAP on the graph and compute slack

Note: It was found that the script provided to generate the benchmarks were not producing a DAG structure with a single source, this made the DFGs not traversable from a single source point, so extra edges were added to the benchmarks to make the graph traversable.

Step 2: Pre-processing design specification

Define Schedule objective from the arguments provided.

- If only latency argument is found, schedule objective will be MM-LC
- If only memory constraint argument is found, schedule objective will be ML-MC
- If both latency and memory constraint argument provided, pareto-optimal simulation was run.

Step 3: Automatically generate ILP Formulation

- Generate .lp file contents:
    - Generate the function to be minimized
    - Generate unique start time constraints / execution constraints
    - Generate dependency constraints from all edges
    - Generate latency and memory constraints
    - Generate opening , closing lines and variable array for the LP file
- Infeasibility:
    - If provided latency constraint is found to be lower than critical path delay, then the problem becomes infeasible.

- If number of memory element constraint is less than any edge weight then the problem becomes infeasible since that particular edge will never be contained within the memory constraint.

Step 4: Solve and extract the solutions using ILP solver GLPK

- Run ILP solver (glpk)
- Parse output and display back
- For the pareto-optimal case, use both the arguments to run both kinds of schedule objective algorithm and then display the pareto-optimal results.

Equations used for the constraints:

- Execution Constraints:

  Start time of each operation is unique. Here, $x_{il} = 1$ means node i is scheduled at time l

  $\sum_l x_{il} = \sum_{l=t_s}^{l=t_L} x_{il} = 1$ where t_L is time of operation I computed with ALAP and t_s is time of operation I computed with ASAP

- Dependency Constraints:

  For every edge from i to j if the distance between them is d_j

  $\sum_l l.x_{il} \geq \sum_l l.x_{jl} + d_j$

- Latency Constraints:

  This constraint keeps in check that all operations are scheduled within the latency constraint time, L.

  $\sum_l l.x_{il} \leq L$

- Memory Constraints:

  For every timestep t, number of registers used right after timestep t has to be less than or equal to memory constraint M.

  $m_t = \sum_{edges\ (i,j)} w_{ij}.x_1.x_2$

  where w is the weight for each edge (i,j), x1 is whether node i was scheduled before or on t and x2 is whether node j was scheduled after or on t+1.

But this equation makes the problem non-linear. To make it linear, we use the following three equations to convert $y = x_1 \wedge x_2$ to a linear form.

$y \geq x_1 + x_2 - 1$

$y \leq x_1$

$y \leq x_1$

- **Evaluation results**

MM-LC:

When only latency argument is provided, the solver tries to minimize memory.

Here, as we increase the latency boundaries and relax the constraints, the memory usage is less and less.

```
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 8
Scheduling MM-LC
Executing ILP
The minimized memory is  80
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 9
Scheduling MM-LC
Executing ILP
The minimized memory is  71
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 10
Scheduling MM-LC
Executing ILP
The minimized memory is  66
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 11
Scheduling MM-LC
Executing ILP
The minimized memory is  59
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 12
Scheduling MM-LC
Executing ILP
The minimized memory is  34
```

If the given latency constraint is not long enough for the critical path, it raises an infeasibility flag stating the minimum latency required.

```
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 5
Traceback (most recent call last):
  File "/home/msukanya/Desktop/759u/test/project/auto.py", line 596, in <module>
    main(sys.argv[1:])
  File "/home/msukanya/Desktop/759u/test/project/auto.py", line 69, in main
    raise Exception(f'Latency constraint makes the problem infeasible, the -l value has to be at least', some)
Exception: ('Latency constraint makes the problem infeasible, the -l value has to be at least', 8)
```

ML-MC:

When only memory argument is provided, the solver tries to minimize latency.

And as shown below, if less than minimum required memory constraint is imposed, it raises an infeasibility flag.

```
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -m 90
Scheduling ML-MC
Executing ILP
The minimized latency is  8
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -m 8
Traceback (most recent call last):
  File "/home/msukanya/Desktop/759u/test/project/auto.py", line 596, in <module>
    main(sys.argv[1:])
  File "/home/msukanya/Desktop/759u/test/project/auto.py", line 63, in main
    raise Exception(f'Memory constraint makes the problem infeasible, the -m value is not enough for edge', node1, 'to', node2
, 'with weight', weight)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^
Exception: ('Memory constraint makes the problem infeasible, the -m value is not enough for edge', 0, 'to', 4, 'with weight',
10)
```

Both:

When both the arguments are given, both the solvers are called, and it minimizes the latency and memory both. This outputs both cases as a trade-off option in terms of both latency and memory.

```
msukanya@ENEELDREN033354:~/Desktop/759u/test/project$ python3 auto.py -g rand_DFG_s10_1.edgelist -l 10 -m 86
both
Scheduling ML-MC
Executing ILP
The minimized latency for memory constraint  86 is 8
Scheduling MM-LC
Executing ILP
The minimized memory for latency constraint  10 is 66 _
```