Project report

Matematyczne Narzędzia Komputerowe w Zastosowaniach Telekomunikacyjnych

Krystian Bytnar, Paweł Woźny, Ricardo Toledo Burgos, Dmytro Kuchynskyi

Description of the problem:

The aim of this problem was to optimize fps for GPU with use of dual-homing. The general scheme of the problem is shown in figure 1. We take a minimal level of fps as 30 and we look for minimal cost. Also optimize the number of servers and their position. We include the cost as infrastructure and data transfer.

The problem was solved by presenting two solutions - one in the form of an exact solution using a software GMPL (CBC/CLP) and the appropriate mathematical model. The second method was presented in language python using the library Networkx. At the same time this code in python helped us to generate random networks for the testing scenario, also it gave us the possibility of drawing the results with the python library 'matplotlib'.

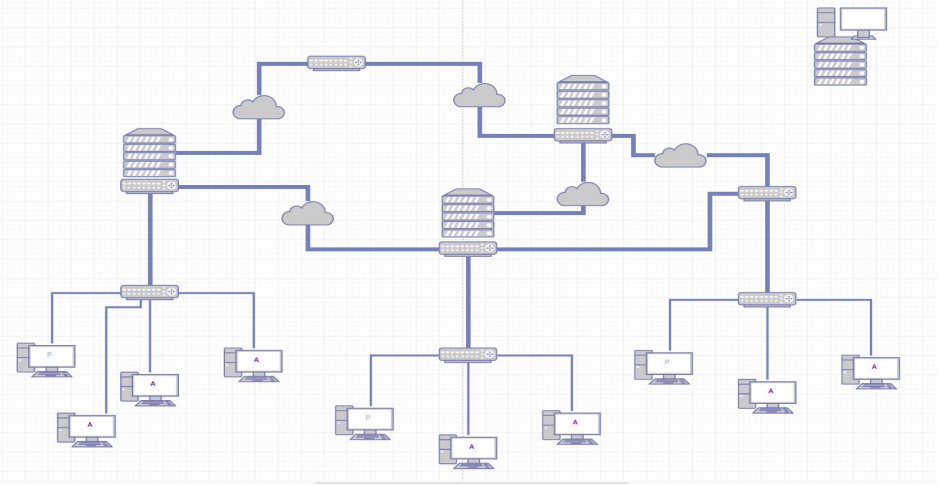

Figure 1. General scheme of the problem

Indices:

$e = 1,\ 2, ..., E$ Links

$v\ = 1,\ 2, ..., V$ Nodes

$d\ = 1,\ 2, ..., D$ Demands

$p\ = 1,\ 2, ..., P$ Paths

$vs\ = 1, 3, ..., VS$ Source nodes

Constans:

h_d Volume of demand d

t_d Sink node of demand d

a_{ev} = 1 if node v is the originating node of link e; 0, otherwise

b_{ev} = 1 if node v is the terminating node of link e; 0, otherwise

K_e Unit cost of link e

c_e Capacity of link e

Variables:

u_{edp} Binary variable corresponding to availability of the link e for all the demands d

ud_{edp} Binary variable to choose the link e for the flow of the all demands d

ue_{ed} Binary variable corresponding to flow of all the demands d allocated to link e

$f_{dv} \geq 0$ Outgoing flow for the demand d allocated to node v (continuous non-negative)

$x_{edp} \geq 0$ Flow realizing demand d allocated to link e (continuous non-negative)

$y_e \geq 0$ Flow realizing all the demands d allocated to link e (continuous non-negative)

Objective:

$$Minimize\ F = \sum_e (K_e * y_e)$$

Constraints:

Constraints to ensure path bifurcation:

$$\sum_e \sum_p (A_{ev} * u_{edp} - B_{ev} * u_{edp}) \begin{cases} 1 & if\ V \cap VS \\ -Pn & if\ v = t_d \end{cases} \quad d = 1,2,...,D; \quad v = 1,2,...,V \tag{1}$$

$$\sum_e (A_{ev} * u_{edp} - B_{ev} * u_{edp}) = 0 \quad d = 1,2,...,D; v = 1,2,...,V;\ p = 1,2,...,P;\ V \neq VS;\ v \neq t_d \tag{2}$$

$$\sum_p (u_{edp}) \geq ue_{ed} \qquad e = 1,2,...,E; \quad d = 1,2,...,D \tag{3}$$

$$\sum_e (A_{ev} * ue_{ed} - B_{ev} * ue_{ed}) = -2 \qquad d = 1,2,...,D; \quad v = t_d \tag{4}$$

$$\sum_p (u_{edp}) \leq Pn - 1 \qquad e = 1,2,...,E; \quad d = 1,2,...,D \tag{5}$$

Constraints (1) and (2) jointly force that for each demand d be assigned to each node source, denominate as server, a path to target node, thanks to sets VS and P we are able to determine exactly how many paths we need to find. The variable ‘u’ determines whether the link belongs to the selected path; the constraints (3) and (4) use the the variable ‘ue’ to limit the terminating links to the target node ‘2 in this case’; the constraint (5) force that for each demand d be assigned minimum two differents paths.

Constraints to send the flow through the the lowest cost route:

$$\sum_{e}\sum_{p}(A_{ev} * x_{edp} - B_{ev} * x_{edp}) \quad \begin{cases} f_{dv} & if \quad V \cap VS \\ -f_{dv} & if \; v = t_d \\ d = 1,2,...,D; & v = 1,2,...,V \end{cases} \tag{6}$$

$$\sum_{e}(A_{ev} * x_{edp} - B_{ev} * x_{edp}) = 0 \quad d = 1,2,...,D;\; v = 1,2,...,V;p = 1,2,...,P;\; V \neq VS; v \neq t_d \tag{7}$$

$$x_{edp} \leq h_d * u_{edp} \qquad e = 1,2,...,E; \quad d = 1,2,...,D; \quad p = 1,2,...,P \tag{8}$$

$$\sum_{vs}(f_{dv}) = h_d \qquad d = 1,2,...,D \tag{9}$$

The constraint (8) assures that the flow be greater than zero ('x' > 0) if, and only if, 'u' = 1, in this way is ensured that the flow goes through the paths set by the previous constraints, the constraints (6) and (7) concern to flow, they set the outgoing flow of each source node and the incoming flow of the node target, so that the total outgoing flow of the demand be equal to summation of the outgoing flows of each source node, and the total flow of the demand go to node target. Thanks to constraint (9) all the flow go through only one path from one source, in this way assures that as long as the flow is not greater than the capacity, only one path and one source will be used.

Constraints to add link capacity:

$$\sum_{p}\sum_{d}(x_{edp}) = y_e \quad e = 1,2,...,E \tag{10}$$

$$y_e \leq c_e \quad e = 1,2,...,E \tag{11}$$

The constraint (10) and (11) assure that the flow for each link will be not greater that the established capacity.

To the first scenario of testing we obtained a random network fig. 2 with 30 nodes with 2 servers and 16 hosts. In a fig. 3 we can see a solution which is provided from python solver.

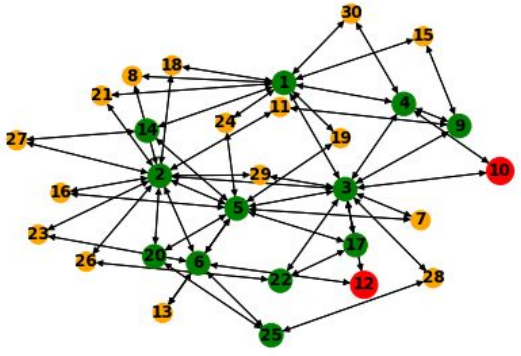

Figure 1. Random network

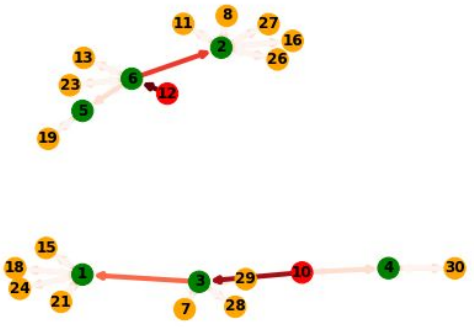

Figure 3. Solution given by a python

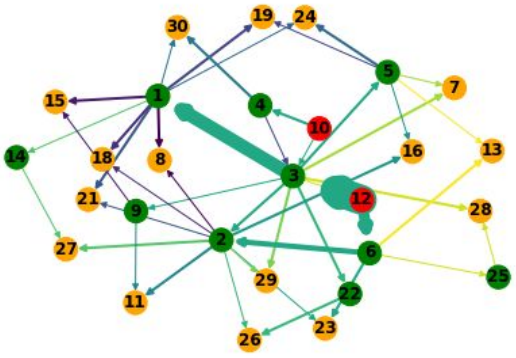

Figure 3. LInk available

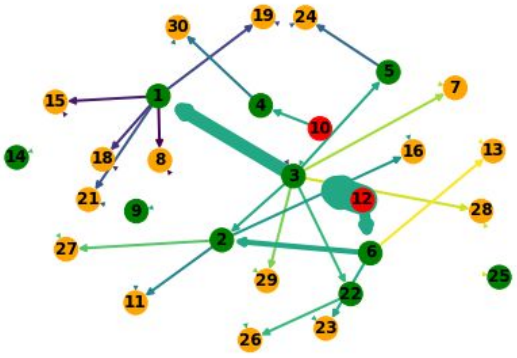

Figure 4. Flow

Figure 1: Random network used in python simplex solution

Figure 3-4, 7-8: solution given by CBC, the first one shows the flow and all of the links availables for do the the demand of each host, by way of assure all the time connectivity with at least one server, the second one shows only the links used to satisfy the demands.

Each program have the same value for the cost of using the network.

Solution found by python code (simplex algorithm):

```
Total cost: 106000
Used links ((node1, node2), flow): [((1, 15), 10), ((1, 18), 10), ((1, 21), 10), ((1, 24), 10), ((2, 8), 10), ((2,
11), 10), ((2, 16), 10), ((2, 26), 10), ((2, 27), 10), ((3, 1), 40), ((3, 7), 10), ((3, 28), 10), ((3, 29), 10),
((4, 30), 10), ((5, 19), 10), ((6, 2), 50), ((6, 5), 10), ((6, 13), 10), ((6, 23), 10), ((10, 3), 70), ((10, 4), 1
0), ((12, 6), 80)]
```

Solution found by GMLP program:

```
Result - Optimal solution found

Objective value:                106000.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             1.15
Time (Wallclock seconds):       1.21

Total time (CPU seconds):       3.41   (Wallclock seconds):       3.48
```

Random network 60 nodes is presented in a fig.5.

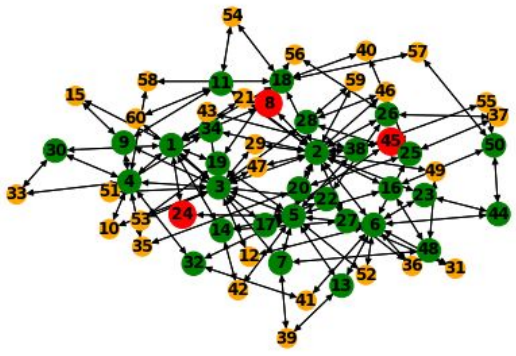

Figure 5. Random network 60 nodes

Figure 6. Solution by simplex python

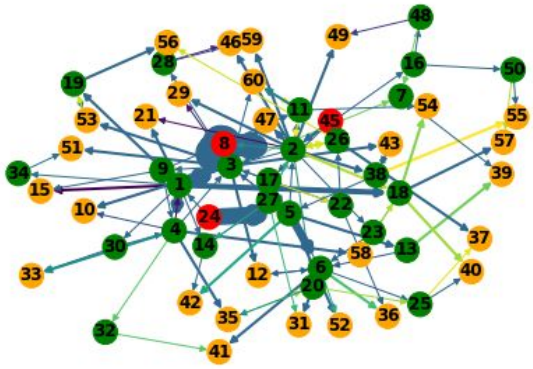

Figure 7. Flow and available links

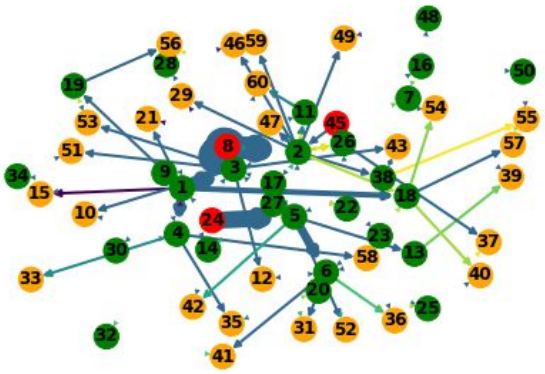

Figure 8. Active links and flow

Solution found by python code (simplex algorithm):

```
Total cost: 187000
Used links ((node1, node2), flow): [((1, 3), 16), ((1, 4), 20), ((1, 15), 10), ((1, 18), 30), ((1, 21), 10), ((1, 3
0), 10), ((1, 34), 10), ((2, 6), 10), ((2, 26), 10), ((2, 29), 10), ((2, 46), 10), ((2, 47), 10), ((2, 49), 10),
((2, 59), 10), ((3, 10), 10), ((3, 12), 10), ((3, 60), 10), ((4, 35), 10), ((4, 58), 10), ((5, 7), 10), ((5, 19), 2
0), ((5, 32), 10), ((5, 38), 20), ((5, 42), 10), ((5, 52), 10), ((6, 31), 10), ((7, 39), 10), ((8, 1), 93), ((17,
3), 14), ((17, 22), 10), ((18, 40), 10), ((18, 54), 10), ((18, 57), 10), ((19, 53), 10), ((19, 56), 10), ((22, 36),
10), ((24, 1), 13), ((24, 5), 80), ((26, 37), 10), ((30, 33), 10), ((32, 41), 10), ((34, 51), 10), ((38, 43), 10),
((38, 55), 10), ((45, 2), 70), ((45, 17), 24)]
```

Solution found by GMLP program:

```
Result - Optimal solution found

Objective value:                187000.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             59.43
Time (Wallclock seconds):       59.69

Total time (CPU seconds):       83.42   (Wallclock seconds):       83.72
```

Characteristic of alternative solution method of this problem:
We used server and host as sources and the capacity of link limits the fps. Also, used two sources to one endpoint and only one active link to the endpoint in a moment. In alternative methods we can use two active links simultaneously in real-time and use load-balancing in order to properly balance the transfer of fps.

Verification of implementation (test scenarios):

Verification of proper working from source to destination
1. Set hd from one source
2. Set hd transmitted from second source
3. View the hd that reached the destination as sum of the hd from sources

Verification of proper working of transit node
1. Count hd transmitted on links that are input to node
2. Count hd transmitted on links that are output from node
3. View that difference from hd transmitted to node and transmitted from node is 0

Verification of proper working of link
1. Count hd emitted from node on the one side of link

2. Count hd emitted from node on the second side of link
3. View that difference from hd emitted from node on one side and second side is 0

Conclusions:
GMPL solution and a precise solution that was done in Python 3 showed it's effectiveness and possibility to solve given optimization problems. Under the same conditions GMPL and Python 3 program solvers showed the same results. The values of total cost were the same for random networks with 30 and 60 nodes. Additionally, in this project we used NetworkX python library for generating random networks for testing. Note that not all randomly generated networks could handle the FPS demand. (that is our bit rate). The solution of the problem by the simplex method of library NETWORKx in python was be able to found the optimal solution with less time than the solution found by CBC, but the method CBC have better accuracy, after many test we could see that sometimes the solution given by CBC was better than the solution given by NETWORKx.