Ali HAIDAR

FAP-MAS Project Report

January 9, 2020

1 Abstract Description

In this report we provide a modeling for solving the FAP (frequency assigning problem) that is well explained in this article[1]. We develop a multi-agent system approach to solve this problem where we propose to map it to a distributed constraint optimization problem. This DCOP is first is modeled and mapped from FAP then we run some DCOP algorithms using an already existing tool (FRODO) on some hard instances. We provide some benchmarks for those algorithms and we discuss the results.

2 Modeling as DCOP

2.1 General definition

We start from the approach of a CSP (constraint optimization problem) which basicall consists of 3 elements **(V, D, C)**:

- **V for variables:** set links between antennas are the variables in the FAP where they could have different frequencies.
- **D for domains:** set of frequencies that a link could choose from would be the domain of this variable.
- **C for constraints:** set of constraints over the links where two links must not interfere according some fixed constants (**k**)

$$|F1 - F2| > k_{1_2}$$

To add the approach of distributed optimization we introduce two new elements **(A,∅)** where we assign an agent of the set **A** to each variable(link) using the function ∅. In our case each agent will be responsible for assiging the right frequency to each link in the instance of the problem.

Last we introduce the last two elements of the model to have a full DCOP model, to our DCSP model we have a cost function and an objective function:

- Cost function and A is the set of already assigned frequencies so that we can keep track if the frequency has been already used and then minimize the number of different used frequencies

$$C(|f_i - f_j|) = \begin{cases} 1, & \text{constraint satisfied} \\ 0, & iff f_i \in A \text{ or } f_j \in A \\ inf, & \text{otherwise} \end{cases} \quad (1)$$

- Minimize the sum of the cost of the violated constraints:

$$\min_{\forall f_i, fj \in F} \Sigma C(f_i, fj) \quad (2)$$

2.2 Simple instance demonstration

We run the DPOP algorithm which is based on dynamic programming on a simple instance having (4 antennas, 3 frequencies and 4 inter-site constraints), the instance is defined below:

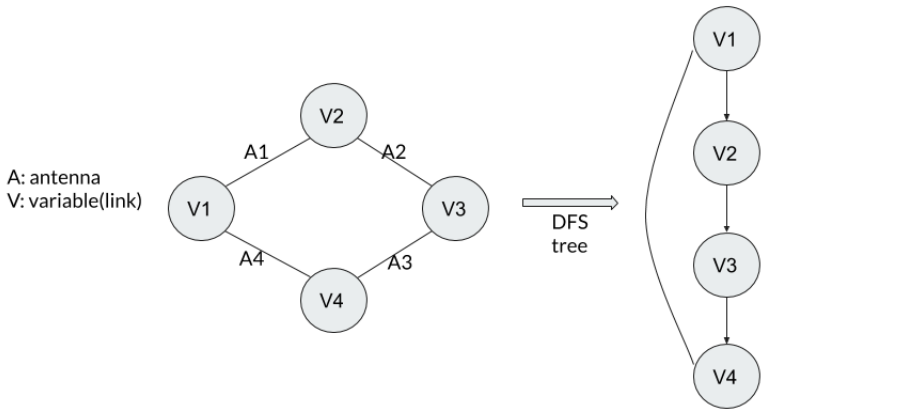

Figure 1: Instance graph to DFS tree.

Starting from the variables defining the links (V1, V2, V3, V4), we generate the DFS tree based on the connections of the antenna sites. Then we define the constants as follows:

- $|V_1 - V_2| > k_{12} = 5$
- $|V_1 - V_4| > k_{14} = 2$
- $|V_2 - V_3| = k_{23} = 3$
- $|V_3 - V_4| = k_{34} = 3$

and the domain for the variables is $\mathbf{D}$[a = 5, b = 8, c = 12], we denote each choice with a letter to make the calculations easier.

The iterations start from the leaf nodes where in the case of this instance it's (V4), we generate the tables of the possibilities and we apply the util phase of the algorithm as shown in figure 2. We keep repeating this phase until we reach the root node (V1) having the summed up hyper cubes, V1 starts the value phase by assigning the value that minimizes the aggregated cost function.

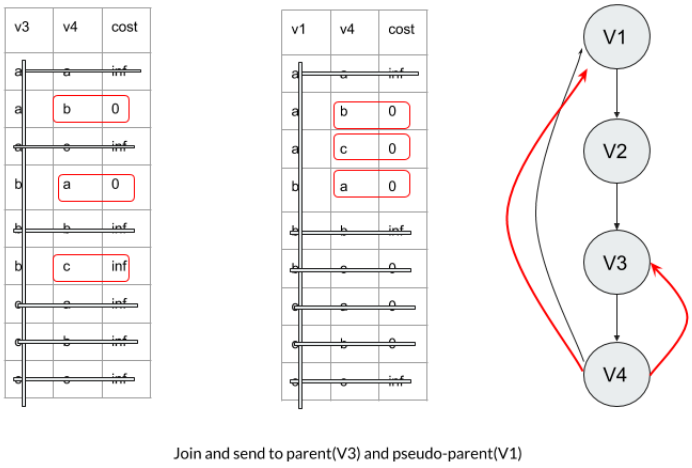

Figure 2: Iteration from the leaf node.

For the sake of simplicity we don't provide all the tables calculated but the result is already shown in figure 3 where it shows the variables with their assigned frequencies after the algorithm finishes from root going down to leafs.

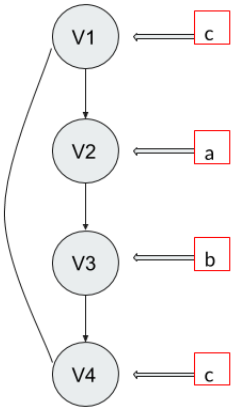

Figure 3: Value phase of the DPOP.

2.3 Analysis and Comments

After running the DPOP algorithm over this simple instance we can tell that the computation of the hyper cubes to be sent by messages between the agents is really expensive, especially when it comes to complicated graphs where we have a lot of antennas and the possibility of having interfered links would be happening more often. But for the FAP problem we could reach a solution that satisfies the constraints, but regarding the fact that we have to minimize the number of the different frequencies used, we can't tackle this in this instance since we have only 3 frequencies and we are obliged to use them all. In general the model of the problem using the cost function **C** explained above we can guarantee that the minimization of the cost function at the level of each node and choosing the minimum at each iteration we end up at the **value** phase by choosing the minimum number of frequencies.

3 Algorithms and Testing

3.1 Testing flow

To test the mapping that we are doing from FAP to DCOP we use a set of Radio Link Frequency Assignment benchmark problems (RLFAP) build from a real network, with simplified data.Those instances are provided in separate files for each problem, so we convert those instances (using an ad-hoc converter) to XCSP instances which is an xml format dedicated to a tool called FRODO. It will help us run different DCOP solutions methods and also compare those solutions over our instances with respect to several features (time, number of messages, etc.). The illustration of the flow is shown in the figure below:

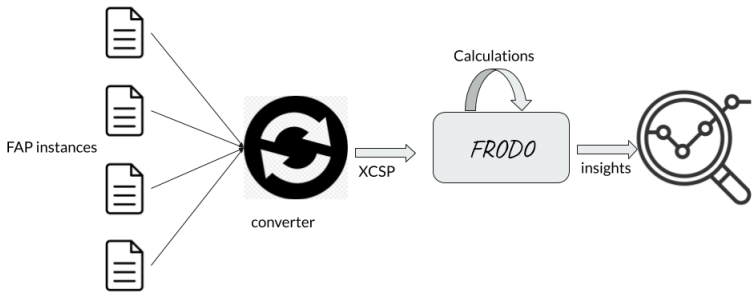

Figure 4: Testing DCOP solutions flow.

The next two subsections shows the results of testing the available DCOP algorithms on FRODO. And those algorithms are split into two main categories **complete** and **incomplete**

due to the fact that the FAP problem is NP-complete so usually it's not that easy to find the solution especially when the instances gets really dense and the number of constraints increases as well as the variables, and then finding the optimal answer would not be always the desired case (an approximate one would be sufficient).We run FRODO java jar on a machine with 16GB of RAM, i7 7700HQ processor giving it the maximum heap space available.

3.2 Mock Test

To test that everything is right we write the instance analysed above in the example using the format given by the FAP website separated in files (var,ctr,dom). And we convert this instance using the converter generating the xml file to be loaded to FRODO.After running DPOP and also ADOPT on this simple instance we obtain the same result we got when we try finding the solution by hand.The illustration below shows the final result and the graph generated by FRODO with the assigned agents.FRODO took V4 as the root of the DFS tree while in the example in section 2.2 we take V1 and we solve based on that but for the result we can guarantee that it got the same results (V1 = 12 , V2 = 5, V3 = 8 , V4 = 5).

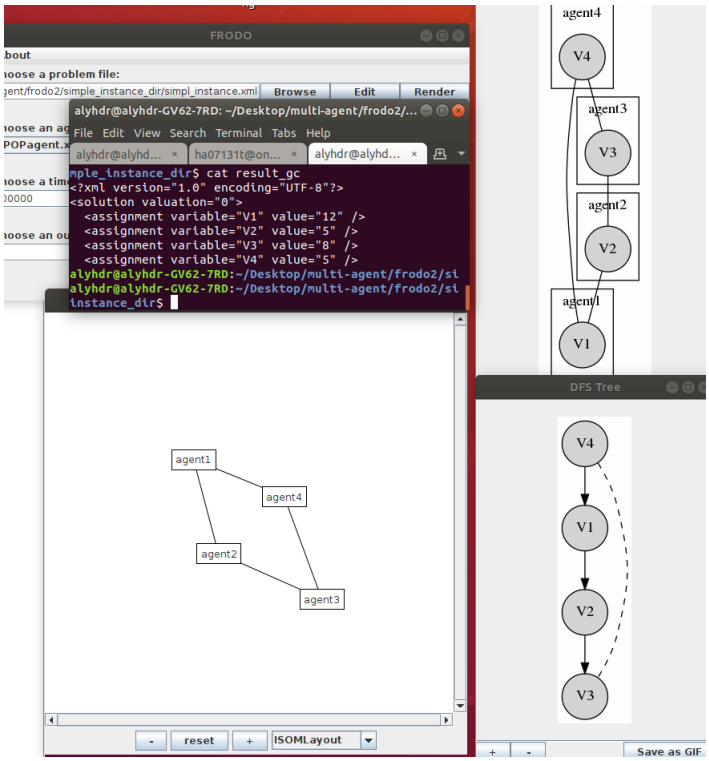

Figure 5: Simple instance FRODO solution (DPOP algorithm).

3.3 Complete Algorithms

First we consider testing on the complete algorithms and for those we have two available methods.First is the ADOPT approach which is based on asynchronous communication between the agents and cost evaluation at the local view optimizing at the end the global cost to minimize. Second we take into account the approach based on dynamic programming which is DPOP that stores all the tables of costs in a bottom-up fashion. After translating some instances of FAP using the ad-hoc converter we run ADOPT with the default timeout of 10m.

Unfortunately the instances are too huge like first one is 916 variables so we started encountering timeout when running those algorithms and it is evident because they are not supposed to be fast in there implementation and for DPOP we got some times out of memory due to the augmentation of the hyper cubes used by the algorithm.

We try also the algorithm SBB (synchronous branch and bound) which basically is a backtracking search algorithm that uses heuristics to cut as much as possible from the search base.This algorithm managed to find a solution for scene 1 and 2 without any violation but not the optimal solution mentioned on the FAP website the table below shows the results and the statistics reported by the algorithm.In figure 6 it shows that SBB used 38 frequencies where the optimal solution is 16 for the first instance scene 1 keeping in mind that the number of frequencies available in the domains is 48 in total, therefore it is not that good. While for instance 2 it used 27 versus the optimal solution which is 14 and that is quite fine.

On average the complete algorithms are not expected to work much on those instances where we have complicated constraints and a lot of antennas where frequencies could interfere.

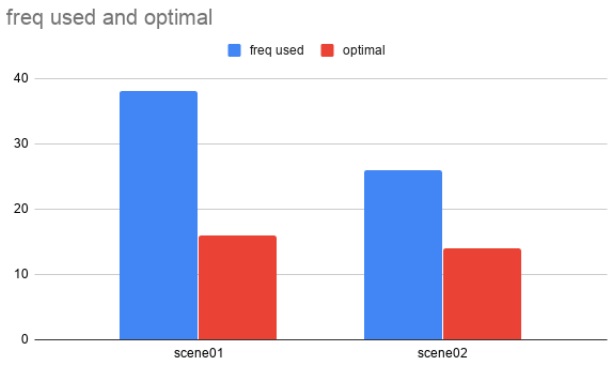

Figure 6: Comparison of the number of used frequencies find by SBB and the optimal one.

And it's worth showing the table of statistics for this algorithm to be compared down with the approximate ones that they didn't even manage to satisfy all the constraints and get a cost of 0.

SBB	complete algo		
	Simulated Time	nb_mess_sent	size_mess_sent
scene01	7,891	2,202,854	244,337,219
scene02	1,031	412,786	23,140,619

Table 1: Table showing the performance of SBB over the two tested instances.

3.4 Approximate Algorithms

In this section we cover the tests over the approximate algorithms which basically tries to get an approximation of the optimal solution, the good thing about such algorithms that they are not so complex talking computation wise. While on the other hand they could get stuck on the local minimum and won't reach the optimal solution.We run 3 different algorithms (DSA,MGM,Max-sum) but the three of them didn't get results where we got constraint violation resulting the cost to go to infinity. But we provide a comparison with respect to performance considering (simulated time,number of messages sent,size of messages sent), all the results are stored in a spread sheet with all the tables but in this report we show a bar graph comparison of the algorithms on different converted instances.Also we show the number of violated constraints reached by the algorithm (i.e the total cost) so that when the cost = 0 then there is no constraint violation and the solution is found but maybe not the optimal one.

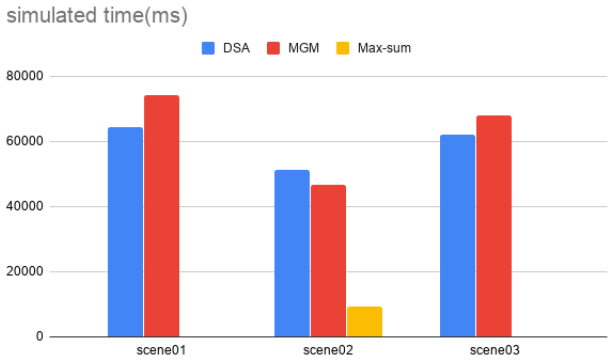

Figure 7: Comparison of the simulated time in (ms).

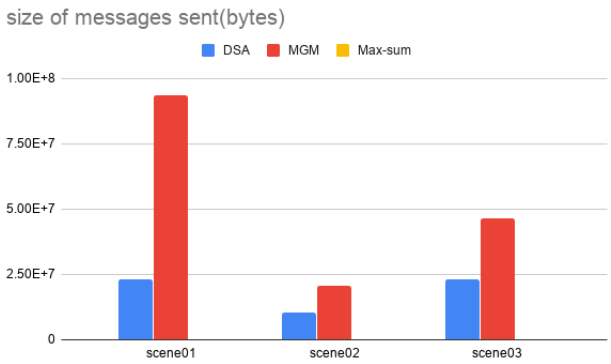

Figure 8: Comparison of the total size of messages sent by each agent in bytes.

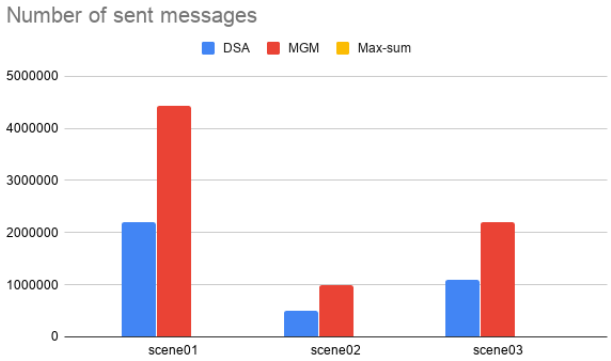

Figure 9: Comparison of the total number of messages sent by each agent.

We show in the figure below 10 the comparison between the three algorithms over the number of violated constraints and there is a huge difference between the results recieved by DSA and MGM for scene 2 and the ones by Max-sum where DSA and MGG violated 23,22 constraints respectively out of 1235 the total number of constraints to be satisfied in this instance, while Max-sum violated 1011 and this relates to the features compared above where the algorithm finished fast getting stuck and not exchanging a lot of messages.

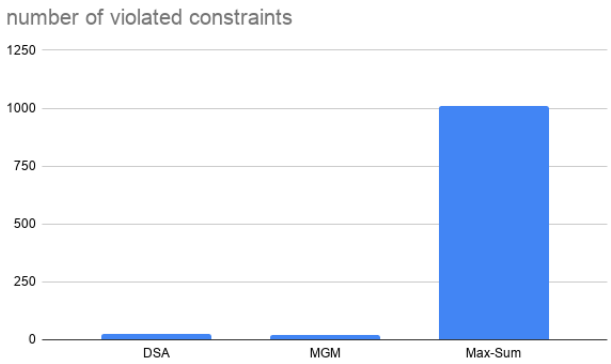

Figure 10: number of violated constraints by each algorithm on instance scene02

4 Conclusion

After running the complete algorithms over the instanecs from FAP we can tell that they are not efficient at all so if one wants to test those, FRODO provides and advanced mode to run it in a distributed manner (i.e on separate machines). In this case we can probably get the results faster and also not run into memory issues with the algorithms that uses a lot of messages between the agents. Regarding the approximate ones, not of all of them performed well where we got the best results with MGM and for the others like (Max-sum) it got stuck in local minimum. As a matter of fact DCOP frame work is good and we can translate any other problem not just FAP ,and model it in such one so that we can run different algorithms and compare.

References

[1] Olivier Roussel,Christophe Lecoutre . *XML Representation of Constraint Networks: Format XCSP 2.1.*

[2] The CELAR Radio Link Frequency Assignment Problems
http://www7.inra.fr/mia/T/schiex/Doc/CELAR.shtml

[3] FRODO:a FRamework for Open/Distributed Optimization
https://manual.frodo-ai.tech/FRODO_User_Manual.html