The CELAR Radio Link Frequency Assignment Problems

Paul Viallard
paul.viallard@etu.univ-st-etienne.fr

Omar Elsabrout
omar.elsabrout@etu.univ-st-etienne.fr

2 January 2019

Abstract

As the technical problems grow with time to be more difficult and sophisticated to solve, researchers thrive to explore new solution techniques to solve those problems and export them into real-life applications. Our scope is on huge constrained problems and the usage of multi-agent systems to solve them. Our consideration of multi agents is due to the fact of extremely big number of variables constraints involved in these problems. In our experimental project, we utilize FRODO[2] an open-source framework for distributed constraint optimization. The purpose of our project is to understand the inner workings of distributed constraint solving algorithms and report the best possible solutions we reach for the CLEAR, Centre d'Électronique de l'Armement, radio link frequency assignment.

1 Introduction

As a practical application to our understanding of distributed constraint problem solving, we aim to solve the CLEAR radio link frequency assignment problems made available in the framework of the European project EUCLID CALMA (Combinatorial Algorithms for Military Applications) build from a real network with simplified data. It contains benchmarks of current solvers in order to evaluate our work and grow the community in general.

To go further into details, the cordiality of our constraints is always two, meaning that all of our constraints are binary and do not involve more than two variables. Those variables are non-linear and have finite domains. Moreover, these provided problems are real-world size problems. To draw the picture, the largest of them are composed of around one thousand variables and almost five thousand constraints.

In our report, we provide a description of the problems and data in hand, our approach to model the problems and prepare the data for the FRODO solver, our configuration of FRODO and the reasons behind them and finally results and our analysis of the experiment. All source code of Python scripts and XML files are available for further testing.

2 Problem Description

Explicitly, the radio link frequency assignment problem tackles the task of giving different frequencies to radio data links in pairs to avoid interference. Each radio link is represented by a variable whose domain is the set of all frequencies that are available for this link.

The essential constraints involve two variables F_1 and F_2:

$$|F_1 - F_2| > K_{12}$$

As described in the problem's documentation, the two variables represent two radio links which are close to each other and it may cause an interference. Naturally, the constant K_{12} depends on the position of the two links and also on the physical environment. It is obtained using a mathematical model of electromagnetic waves propagation which is out of the scope of our work. We are more interested in solving the problems than interested in the actual physical details (as long as they do not contribute in our results).

In addition, for each two radio links, two frequencies must be assigned in a way that one is for the communications from A to B and the other is for the communications from B to A. In the case of the CELAR instances, a technological constraint appears which states that the distance in frequency from A to B and from B to A must be exactly equal to 238.

2.1 Criteria Optimization

In order to evaluate the obtained frequency assignments, we need standards upon which we build our judgment of this obtained solution. In the scope of our project, we focus on two main criteria. First, minimization of frequency values. In other words, the frequencies assigned to data links must be minimized for related engineering power consumption concerns. The second criterion is minimizing the number of used frequencies as there will be data links far from each other so they can use the same frequencies. This facilitates installing future data links with the same domains of frequencies.

2.2 Provided Information

For each problem instance, we are provided by domains, variables, constraints and criteria all in separate text files. These files together describe the details of the problem. However, the domain file specifies the numerical values for the available domains to which variables can be assigned to. Moreover, variable files assign them to the aforementioned domains. Next, constraint files involve variables using their number from the variable files. Lastly, criteria files explain the cost of soft constraints (if any) as guidance towards the best feasible solver. All further details of syntax are available in the documentation of CLEAR.

3 Modeling

When it comes to modeling, we discuss two different parts regarding transforming the problems fro the provided format into another format fit for FRODO. The first part is our objective functions regarding cost assignment for constraints to abide the two target criteria previously mentioned in subsection 2.1. On the other hand, the second part discusses the relation between the number of agents and variables whether it can be optimized or it is not worthy of our computation resources.

3.1 Objective Cost Functions

We propose our own objective functions to replace the essential constraints introduced by the CLEAR problem itself. The goal behind this move is to customize the resulting cost out of each constraint to achieve minimization of frequency values and minimizing also the number of used frequencies. Furthermore, we also alter the concept of hard constraints in our problems to be completely formed of only soft constraints with an extremely high cost. The reason behind this decision is that it makes the development part easier to debug, it gives a clearer analysis of the performance of the objective function and lastly it leads the investigation to know whether we can reach a fairly expensive solution if the problem is unfeasible or not.

Our first objective cost function, named **condition**, targets the minimization of number of frequencies as the formula:

$$if(|F_1 - F_2| \geq K_{12}), \text{then return } 0, \text{else return } a_t$$

a_t here corresponds to a vector a that has 5 costs starting with the cost of a hard constraint then the 4 costs provided for soft constraints in descending order. It is represented in an example as the following:

$$a_0 = 10000 \qquad \text{hard} \qquad (1)$$
$$a_1 = 1000 \qquad \text{soft} \qquad (2)$$
$$a_2 = 100 \qquad \text{soft} \qquad (3)$$
$$a_3 = 10 \qquad \text{soft} \qquad (4)$$
$$a_4 = 1 \qquad \text{soft} \qquad (5)$$

On the other hand, our second objective cost function, named **absolute**, tackles the minimization of frequency values as the formula:

$$a_t||F_1 - F_2| - K_{12}|$$

Such formula follows the same notation as the previous one with the modification that involves the values of F_1, F_2 and K_{12} into the return value of the total cost. Both formulas are used for all the tested problems with different resulting solutions.

3.2 Agents and Variables

Our default strategy for choosing the number of agents is to assign it the same as the number of variables involved in each problem. With this being decided, we aim to have each agent in charge of each link, or in this context a broadcasting tower, as it leaves the problem simply modeled and realistically applicable. Without a doubt, we can change the strategy with another for more optimization. However, it would require more knowledge of the nature of frequencies and the basis upon which the constraints are formed. Due to the huge number of constraints and variables presented in each problem, optimizing the strategy requires a heavier study and considered out of the scope of our interest in the behavior of our solver.

4 Parser

Our implementation of a parser tool to read the problem's text files is built in a Python 3.6 environment. We utilize a Python based library named LXML[1] to construct XML files representing the input problems for FRODO. The parser reads the files `var.txt`, `dom.txt`, `ctr.txt` and `cst.txt` for each provided instance corresponding to the variable names, used numerical domains, constraints applied on variables and preferred cost per constraint respectively. With the exception of instances 1, 2 and 3 of containing only hard constraints, we ignore costs files and modify the previously mentioned formulas accordingly. All further details on the actual format of text files are available in the CLEAR documentation so we do not repeat them in our report to avoid redundancy. Nonetheless, we move on to explain the construction fo our XML files in the format presented in figure 1.

For each instance we generate two different XML files for the two objective cost functions. Those two XML files are similar in the parts regarding variables, agents, constraints and domains. On the other hand, they are different when it comes to defining predicates to which the constraints are referenced.

5 Configuration

When it comes to the discussion of the environmental setup of our experiments, we are dictated to define the configuration of our used hardware, used framework and the parameters of such framework. The reason behind the need to explain such configurations is to state the conditions of the experiment to be able to subjectively compare it with other benchmarks.

Our hardware configuration includes an Intel Core i7-8700K processor with 12 cores of 3.7GHz and 24 threads. Such a processor is assumed to be sufficient for the task but it takes long periods to calculate a solution for instances.

```xml
<instance>¬
  <presentation format="XCSP 2.1_FRODO" maxConstraintArity="2" maximize="false" name="scen07"/>¬
  <agents nbAgents="400">¬
    <agent name="agent001"/>¬
    <agent name="agent002"/>¬
    <agent name="agent003"/>¬
    ...¬
    <agent name="agent400"/>¬
  </agents>¬
  <domains nbDomains="8">¬
    <domain name="dom0" nbValues="49">48 16 30 ... 792</domain>¬
    <domain name="dom1" nbValues="45">44 16 30 ... 792</domain>¬
    <domain name="dom2" nbValues="23">22 30 58 ... 792</domain>¬
    <domain name="dom3" nbValues="37">36 30 44 ... 778</domain>¬
    <domain name="dom4" nbValues="25">24 16 30 ... 792</domain>¬
    <domain name="dom5" nbValues="7">6 142 170 240 380 408 478</domain>¬
    <domain name="dom6" nbValues="43">42 30 44 ... 792</domain>¬
    <domain name="dom7" nbValues="23">22 16 30 ... 394</domain>¬
  </domains>¬
  <variables nbVariables="400">¬
    <variable agent="agent001" domain="dom1" name="var13"/>¬
    <variable agent="agent002" domain="dom1" name="var14"/>¬
    <variable agent="agent003" domain="dom1" name="var15"/>¬
    ...¬
    <variable agent="agent400" domain="dom1" name="var912"/>¬
  </variables>¬
  <predicates nbPredicates="2">¬
    <predicate name="gt">¬
      <parameters> int X1 int X2 int K int C</parameters>¬
      <expression>¬
        <functional>if(gt(abs(sub(X1, X2)), K), 0, C)</functional>¬
      </expression>¬
    </predicate>¬
    <predicate name="eq">¬
      <parameters> int X1 int X2 int K int C</parameters>¬
      <expression>¬
        <functional>if(eq(abs(sub(X1, X2)), K), 0, C)</functional>¬
      </expression>¬
    </predicate>¬
  </predicates>¬
  <constraints nbConstraints="2865">¬
    <constraint arity="2" name="var13_var14_eq_238" reference="eq" scope="var13 var14">¬
      <parameters> var13 var14 238 100000000</parameters>¬
    </constraint>¬
    <constraint arity="2" name="var13_var15_gt_59" reference="gt" scope="var13 var15">¬
      <parameters> var13 var15 59 1</parameters>¬
    </constraint>¬
    </constraint>¬
   ...¬
    <constraint arity="2" name="var911_var912_eq_238" reference="eq" scope="var911 var912">¬
      <parameters> var911 var912 238 100000000</parameters>¬
    </constraint>¬
  </constraints>¬
</instance>¬
```

Figure 1: A summarized representation of XML file of instance 7.

One other important factor is the available memory which is 64GB of RAM to save the calculated constraints in memory to speed up the search for variable assignments.

While on the other hand, we continue to state our configuration of the FRODO framework. There exist 19 algorithms available to use to solve our problem. However, not all of them have the same performance. Some of them run in exponential time and others are polynomial. Since we have limited hardware resources and instances of problems with huge number of constraints. We decided to use only one polynomial time algorithm which is MGM with two different implemented objective cost functions. The actual detailed comparison of performances among the 19 algorithms can be presented by doing a literature review on their papers which is not the aim of our project since we can never run all algorithms on our hardware. We focus on the performances of our implemented cost functions using MGM. We run the following command to use FRODO:

```
java -cp [frodo.jar path] frodo2.algorithms.AgentFactory [instance
XML path] frodo2/agents/MGM/MGMagentJaCoP.xml > [output file path]
```

Next, we read the output file with another parser we implemented to save the results and prepare it for plotting and analysis.

6 Results

Finally, we reach the part where we present the output of our experiments. As we mentioned before, we ran our models on the problems 2, 5, 6, 9 and 11 since each one represents a sub-sample of a group of similar problems. Moreover, after several trials of running all the algorithms available in FRODO, we came to realize some of them have exponential time complexity so they do not terminate on our hardware setup. Hence, we use only the MGM algorithm with different objective functions, **condition** and **absolute** respectively, as discussed in subsection 3.1. The results of each experiment are described in the following table:

Prob. No.	Cost Type	Variables	Optimal Cost	Simulation Time (ms)	Messages
2	Hard	200	0	26,106	988,000
5	Condition	400	25,800,000	41,396	2,078,400
5	Absolute	400	25,700,000	41,500	2,078,400
6	Condition	200	117,145	26,126	1,057,600
6	Absolute	200	929,344	26,340	1,057,600
9	Condition	680	312,663	41,629	3,282,400
9	Absolute	680	2,997,494	43,326	3,282,400
11	Condition	680	6,900,000	43,685	3,282,400
11	Absolute	680	755,328	42,695	3,282,400

We state the number of each instance with the type of the cost function, number of variables, optimal total cost, simulation in time in milliseconds and finally the number of sent messages among all agents.

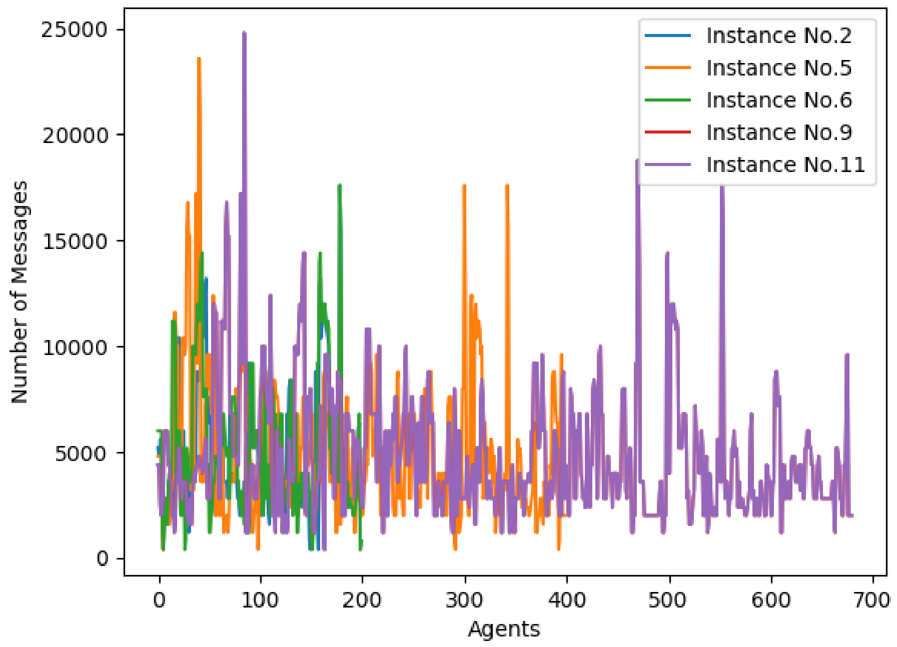

Figure 2: A graph showing the number of messages sent and received per agent across the tested problem instances.

As we can remark, the instance 2 is a special case since it does not contain any soft constraints with multiple costs. Since the problem is feasible. It was solved in a fairly short simulation time and number of messages. More interestingly, we find that the usage of different cost functions does not mean necessarily a different number of messages and simulation time. Furthermore, we deduce that the number of variables and agents is not the only factor for the total optimal cost. However, it also depends on the used cost and the difficulty of the problem whether it is solvable or not. For instance, we find instance 9 and 11 have the same number of variables and the same number of messages. On the other hand, they have different costs on both cost functions.

We observe in more details the number of messages sent among agents in Figure 2 where we find in each instance roughly the same number of message except an arbitrary group of agents where most of the communication to solve the problem is executed. Those agents are represented by a spike in the graph. In addition, we deduce that the number of messages per agent is proportional to the number of agents involved in solving the problem as in instance 11 where it dominates all the numbers.

7 Conclusion

To sum up, we introduced the CLEAR radio link frequency assignment problem, we described the problem in details by explaining the variables and the constraints. Next, we presented our approach of modeling and objective cost functions. Then, we briefly presented the development of our parser to convert the problem text file format into an XML format ready to be processed by FRODO with stating the configuration used in FRODO. Finally, we presented our results and explained our insights of the correlations among optimal cost, simulation time and number of variables. For sure, we find the CLEAR problem interesting and requires more dedication to have advanced results to compete with mentioned benchmarks on the website.

References

[1] S. Behnel, M. Faassen, and I. Bicking. lxml - XML and HTML with python. http://lxml.de/ Accessed on 13 August 2012.

[2] T. Léauté, B. Ottens, and R. Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the IJ-CAI'09 Distributed Constraint Reasoning Workshop (DCR'09)*, pages 160–164, Pasadena, California, USA, July 13 2009. `https://frodo-ai.tech`.