The Radio Link Frequency Assignment Problem

Zhantileuov Eldiyar

University of Jean Monnet & Ecole des Mines, France
zhantileuov.eldiyar@emse.fr

Abstract. The Radio Link Frequency Assignment Problem is an abstraction of a real-world military application, which is associated with the assignment of frequencies to a radio links. The RLFAP contains eleven large-scale instances, each of which has its own constraint and optimization requirements. In this report, this problem set is accomplished through the introduction of multi-agent systems and the FAP is modeled as a distributed constraint optimization problem (DCOP).

An open-source FRODO environment is used to evaluate the performances of several DCOP solution methods on RLFAP instances.

Keywords: Distributed Constraint Optimization Problem, FRODO (framework for distributed constraint optimization), Dynamic Programming Optimization Protocol (DPOP).

1 Introduction

The EUCLID CALMA is a group of European research bodies that was formed to investigate the use of AI techniques to aid military decisions [1]. In the framework of this project, the French "Centre d'Électronique de l'Armement" (CELAR) has made publicly available the Radio Link Frequency Assignment Problem (RLFAP or simply FAP). The RLFAP involves the assigning of frequencies to a set of radio links defined between pairs of sites in a network. The frequency assignment should satisfy certain restrictions in order to limit the interference between links. It is proposed to develop a multi-agent system to solve FAP and map it as a distributed constraint optimization problem (DCOP).

There are eleven large-scale CELAR instances with different optimization and constraint requirements. Each scenario consists of four separate files (var.txt, dom.txt, ctr.txt, cst.txt) that describe its variables, their domain, constraints and the main objective criteria. The number of variables and constraints on these variables reaches 916 and 5548, respectively, which makes the RLFAP a non-trivial problem for any algorithm (see Table 1 to find numerical values of eleven instances).

An open source FRODO framework is used to implement several DCOP solution methods on instances and compare their performance metrics, such as time and number of messages. And since this environment supports the file format based on XCSP 2.1 format, four separate files need to be parsed to obtain a file that describes DCOP. My implementation of a parser tool is written in python programming language.

2 Problem Description

2.1 The RLFAP Optimization criteria

The RLFAP instances can be classified into three optimization criteria, depending on the feasibility of the problem. If an instance can be solved without constraint violation, then its optimality is defined as one of the following two criteria [1]:

1. the optimal solution is one where the largest assigned frequency is minimal;
2. the optimal solution is one with the fewest number of different values in its variables.

If the problem cannot be solved without violating constraints, its optimality belongs to the third criteria:

3. the optimal solution is one where the weighted sum of violated constraints is minimal. In other words, the third type of optimization criteria is to find a solution that minimizes the objective function as follows:

$$a_1 \times nc_1 + a_2 \times nc_2 + a_3 \times nc_3 + a_4 \times nc_4 + \\ + b_1 \times nv_1 + b_2 \times nv_2 + b_3 \times nv_3 + b_4 \times nv_4 \tag{1}$$

   Where nc_i is the number of violated constraints of priority i, nv_i is the number of modified variable with mobility i . Mobility for the radio link states the cost for changing the frequency from its assigned default. The value of the weights a_i *and* b_i are given.

2.2 The RLFAP Constraints

In a network for any pair of sites A and B, there are two radio links: from A to B and vice versa, from B to A. In our problem the radio links are variables for which frequencies must be set (see Fig. 1.). This means that for each pair of sites, two values must be assigned. In addition, these values must be assigned in such a way as to prevent interference.

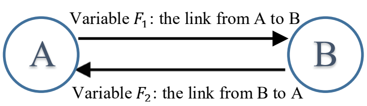

Fig. 1. Demonstration of radio links in a pair of sites A and B.

All constraints in the RLFAP is binary, i.e. involving only two variables. There are two types of constraints in instances:

1. the absolute difference of variables F_1 *and* F_2 in a pair of sites is greater than the constant number k_{12} :

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

2. the absolute difference of variables F_1 *and* F_2 in a pair of sites is equal to the 238:

$$|F_1 - F_2| = 238$$

Note that: the constant number k_{12} is not always correct, as it is overestimated to ensure the absence of interference.

The CELAR scenario consists of only one objective, which can be defined as one of three optimization criteria, but in each scenario, there are thousands of constraints with both types of restrictions. The Table 1 shows CELAR instances with the number of variables and constraints, the feasibility of the task, and the type of optimization criteria.

Instances	No. of variables	No. of constraints	Feasi-bility	Optimization criteria (Minimize)
Scen01	916	5548	Yes	Number of different values used
Scen02	200	1235	Yes	Number of different values used
Scen03	400	2760	Yes	Number of different values used
Scen04	680	3968	Yes	Number of different values used
Scen05	400	2598	Yes	Number of different values used
Scen06	200	1322	No	The maximum value used
Scen07	400	2865	No	Weighted constraint violation
Scen08	916	2744	No	Weighted constraint violation
Scen09	680	4103	No	Weighted constraint violation
Scen10	680	4103	No	Weighted constraint violation
Scen11	680	4103	No	Weighted constraint violation

Table 1. The list of CELAR instances.

3 Modeling the FAP as a DCOP

The framework of DCOP is a 7-uplet $\langle A, X, D, C, \phi, f_{ij}, F \rangle$. Where:

- ***X* (Variables)** – as a variable we take radio links. Each radio link is determined by one variable.
- ***D* (Domains)** – each domain defines a list of the different frequency values that each variable can take.
- ***C* (Constraints)** – are relations between some variables, in our problem between two variables (see the section 2.2 The RLFAP Constraints).
- ***A* (Agents)** – each agent is responsible for assigning value for a one variable, e.g. agent A_1 is responsible for assigning frequency to radio link from A to B.

Since in instances there are thousands of variables, we need an agent for each of them.

- $\boldsymbol{\phi}$ – is a function assigning variables from $\boldsymbol{X}$ to agents from $\boldsymbol{A}$,

$$\phi(x_i) = agent_i$$

- $\boldsymbol{f_{ij}}$ – a cost function for each pair x_i and x_j. According to the constraints defined (section 2.2), we derive the following cost functions:
  - for two links i, j constrained by $|x_i - x_j| > k_{ij}$ (k_{ij} is the constraint deviation of pair x_i and x_j):

$$f1_{ij}(x_i, x_j) = \begin{cases} |x_i - x_j| - k_{ij}, & if\ |x_i - x_j| > k_{ij} \\ 3 \times (k_{ij} - |x_i - x_j|), & otherwise \end{cases} \quad (2)$$

  - for two links i, j constrained by $|x_i - x_j| = 238$:

$$f2_{ij}(x_i, x_j) = \begin{cases} 0, & if\ constraint\ respected \\ ||x_i - x_j| - 238|, & otherwise \end{cases} \quad (3)$$

- F – an objective function that evaluates assignment $\mathcal{A} = \{(x_1, d_i), (x_2, d_j), \ldots (x_n, d_k)\}$ by summing up the cost functions from equation (1) and (2):

$$F(\mathcal{A}) = \sum_{x_i, x_j \in X} f1_{ij}(d_i, d_j) + f2_{ij}(d_i, d_j) \text{ where } x_i \leftarrow d_i \quad (4)$$

4 XCSP Generator

The tool for parsing text files (var.txt, dom.txt, ctr.txt, cst.txt) for a given instances is written in python programming language. A Python based library named XML is utilized to construct XML files that are used as input problems for the FRODO framework. At first, it reads text files; makes connections between variables and their domains; then it defines cost functions according to the constraint type. Finally, the parser generates a xml file. The output xml file is in XCSP 2.1 format, as required by FRODO (see Fig.2.). In other words, the parser describes the CELAR files in a machine-friendly language.

In addition, another important aspects that should be noted are the weighting index of the constraints and the mobility of assigned variable that are given in the last field of ctr.txt and var.txt [2]. The mobility of the variable is an index of the cost of changing its value, and therefore the weights of the constraints vary from 0 to 4, which indicates that 0 is a hard constraint, and values from 1 to 4 indicate an increase in weights, as indicated in ctr.txt field. Due to this additional information that can be provided in instances and due to the limitation of my hardware resources, please note that the instances contain a huge number of constraints, our main objective function is subject to change.


```xml
<instance>
   <presentation format="XCSP 2.1_FRODO" maxConstraintArity="2" maximize="false" name="example_prob1" />
   <agents nbAgents="10">⋯
   <domains nbDomains="3">
      <domain name="DOM0" nbValues="44">16  30  44  58  72  86 100 114 128 142 156 254 268 282 296 310 324 338 352 366 380 394 414 428 442 456 470 484 498 512 526
      <domain name="DOM1" nbValues="44">16  30  44  58  72  86 100 114 128 142 156 254 268 282 296 310 324 338 352 366 380 394 414 428 442 456 470 484 498 512 526
      <domain name="DOM2" nbValues="22">30  58  86 114 142 268 296 324 352 380 414 442 470 498 526 554 652 680 708 736 764 792</domain>
   </domains>
   <variables nbVariables="10">
      <variable agent="X1" domain="DOM1" name="X1" />
      <variable agent="X2" domain="DOM1" name="X2" />
      <variable agent="X3" domain="DOM2" name="X3" />
      <variable agent="X4" domain="DOM2" name="X4" />
      <variable agent="X5" domain="DOM1" name="X5" />
      <variable agent="X6" domain="DOM1" name="X6" />
      <variable agent="X7" domain="DOM1" name="X7" />
      <variable agent="X8" domain="DOM1" name="X8" />
      <variable agent="X9" domain="DOM1" name="X9" />
      <variable agent="X10" domain="DOM1" name="X10" />
   </variables>
   <functions nbFunctions="2">
      <predicate name="GRT" return="int">
         <parameters>int X1 int X2 int D</parameters>
         <expression>
            <functional>if(gt(abs(sub(X1, X2)), D), abs(sub(abs(sub(X1, X2)), D)), mul(3, abs(sub(abs(sub(X1, X2)), D))))</functional>
         </expression>
      </predicate>
      <predicate name="EQL" return="int">
         <parameters>int X1 int X2 int D</parameters>
         <expression>
            <functional>if(eq(abs(sub(X1, X2)), D), 0, abs(sub(abs(sub(X1, X2)), D))</functional>
         </expression>
      </predicate>
   </functions>
   <constraints nbConstraints="9">
      <constraint arity="2" name="constraint NUM0" reference="GRT" scope="X1 X2">
         <parameters>X1 X2 1</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM1" reference="GRT" scope="X3 X4">
         <parameters>X3 X4 1</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM2" reference="GRT" scope="X5 X6">
         <parameters>X5 X6 1</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM3" reference="EQL" scope="X7 X8">
         <parameters>X7 X8 238</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM4" reference="GRT" scope="X7 X9">
         <parameters>X7 X9 59</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM5" reference="GRT" scope="X7 X10">
         <parameters>X7 X10 84</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM6" reference="GRT" scope="X8 X9">
         <parameters>X8 X9 84</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM7" reference="GRT" scope="X8 X10">
         <parameters>X8 X10 59</parameters>
      </constraint>
      <constraint arity="2" name="constraint NUM8" reference="EQL" scope="X9 X10">
         <parameters>X9 X10 238</parameters>
      </constraint>
   </constraints>
</instance>
```


Fig. 2. Representation of XML file of instance 1 with 10 variable limit.

5 Experiments and results

In this report, I made experiments on the first scenario, where the objective function is to find a solution minimizing the number of different values used for the assignment of the variables. This instance contains 916 variables and 5548 constraints on these variables that are not violated. At first, I simplified the data set by minimizing the number of variables to 10. At the same time, the number of constraints and domains also changed, 9 constraints and 44 dummy domain. Once, I get simplified data files (var.txt, dom.txt, ctr.txt, cst.txt), they are sent to the python code, where we receive one file in .xml format (e.g. items_10.xml). Further, using FRODO framework, I have solved the problem by using Dynamic Programming Optimization Protocol (DPOP) and Distributed Stochastic Search Algorithm (DSA). There are 19 algorithms available to use to solve our problem. However, not all of them have the same performance. After solving the problem using two DCOP solution methods, we compare these methods with respect to some criteria, such as time and number of messages sent.

For data analyze the data, one simplified data set with 10 variables is not enough. Therefore, I created another dataset with 20, 30, 40 and 50 variables and took the same steps on this data.

The Table 2 and 3 shows the performance metrics of the DPOP and DSA, respectively, for 10, 20, 30, 40 and 50 variables.

Instance	No. of variables	No. constraints	Time (ms)	Cost	No. messages	Max. message size
Scen01	10	9	104 275	335	60	1 151 231
Scen01	20	22	102 790	902	147	1 378 043
Scen01	30	27	110 575	902	187	1 390 426
Scen01	40	71	Time out			
Scen01	50	130	Time out			

Table 2. The performance metrics of DPOP

Instance	No. of variables	No. constraints	Time (ms)	Cost	No. messages	Max. message size
Scen01	10	9	2 981	535	3 582	230
Scen01	20	22	2 876	1 086	8 756	230
Scen01	30	27	2 763	1 218	10 746	230
Scen01	40	71	3 730	7 465	28 258	230
Scen01	50	130	3 913	14 031	51 740	230

Table 2. The performance metrics of DSA

From the performances of two methods, we can see that DSA is computationally fast and the maximum message does not change with the increasing variables. However, in DPOP, I ran into a time out problem when there were 40 and 50 variables, and its computation time was for 35 times longer than in DSA. The DSA agents sent about 60 times of the number of messages than DCOP agents. The explanation for this may be that the DPOP agents have parent-child structure and the agents do not randomly send value and cost messages to all their neighbors. While the DSA agents perform local calculations and send messages to all their neighbors. In the terms of cost, the DPOP algorithm gave more optimal solution than the DSA. Below you can find graphical representations of performance metrics (see Fig.3. and Fig.4.).

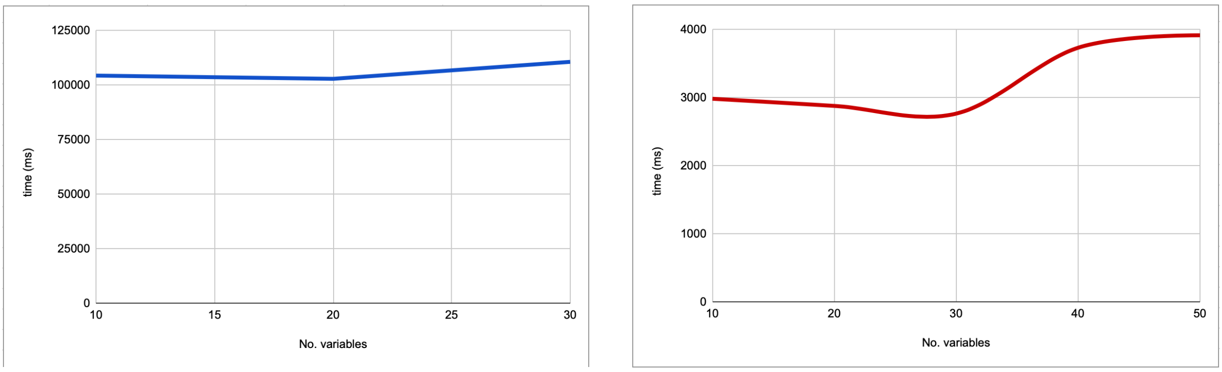

Fig. 3. Representation of number of messages sent with the increase of variables (left-blue graph is DPOP and right-red is DSA).

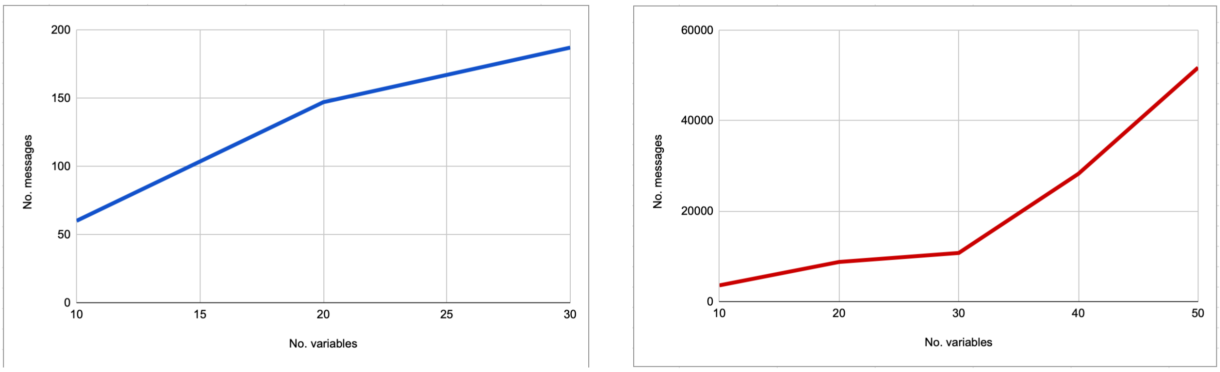

Fig. 4. Representation of computational time with the increase of variables (left-blue graph is DPOP and right-red is DSA).

6 Conclusion

In this work, the Radio Link Frequency Assignment Problem was studied. The problem is mapped to distributed constraint optimization problem (DCOP) and can be solved using 19 complete and incomplete solution methods in the FRODO open source framework. The real-life samples are taken from French CELAR which contains thousands of variables and constraints.

At the first step of investigation, the XCSP generator is constructed in the python environment. This parser receives 4 .txt files to create a .xml file. The latter is then used as input data in FRODO and several algorithms can be tested.

In my case, I have studied two DCOP solutions: DPOP and DSA. After computations, I can summarize that the DSA is preferable, if there is a need for a quick solution to the problem. But in this case, we will be content with a suboptimal solution. And if your network is strongly limited, then the DPOP method will be better.

References

1. Lau, T.L. & Tsang, Edward & Sq, Colchester. (1998). Solving the Radio Link Frequency Assignment Problem with The Guided Genetic Algorithm.
2. https://www7.inra.fr/mia/T/schiex/Doc/rlfap.shtml#var