Multi Agent System Project

Ahmad ALIBRAHIM

1 CSP, DCSP, and DCOP

1.1 Constraint Satisfaction Problem

A Constraint Satisfaction Problem (CSP) is a triplet <X, D, C> where:

- X = $x_1, ..., x_n$ is the set of variables to instantiate.
- D = $D_1, ..., D_m$ is the set of domains. Each variable x_i has a domain D_j
- C = $C_1, ..., C_k$ is the set of constraints, which are relations between some variables from X.

1.2 Distributed Constraint Satisfaction Problem

A Distributed Constraint Satisfaction Problem (DCSP) is a 5-uplet <A, X, D, C, $\emptyset$> where <X, D, C> is a CSP and:

- A = $a_1, ..., a_t$ is the set of agents.
- $\emptyset$: X $\rightarrow$ A : Function assigning variables from X to agents in A.

1.3 Distributed Constraint Optimization Problem

A Distributed Constraint Optimization Problem (DCOP) is a DCSP <A, X, D, C, $\emptyset$> with:

- A cost function $f_{ij} : D_i \times D_j \rightarrow \mathbb{N} \cup \infty$
- An objective function F: $D \rightarrow \mathbb{N} \cup \infty$. For example: F=$\sum_{ij} f_{ij}(d_i, d_j)$

2 Model FAP as DCOP

FAP problem can be mapped into DCOP <A, X, D, C, $\emptyset$, f_{ij}, F> by the following criteria:

- X: Set of radio links
- A: set of agents

- D: Set of available frequencies (finite set)
- C: set of binary constraints between two links from X, in our special case, the only constraint between two link is that the difference of their frequencies should be greater than a given constant K:
  - Let D_1 and D_2 be the set of frequencies corresponding to agent A_1 and A_2 respectively and f_{1_i} and f_{2_j} be the chosen frequencies by agents A_1 and A_2 respectively, then given the constant K_{12}, the constraint between these 2 links is : $|f_{1_i} - f_{2_j}| > K_{12}$
- $\emptyset : X_i \rightarrow A_i$. In this case, for simplicity, each agent from A is responsible for a link from X.
- f_{XY}: Let C_{XY}: $|d_X - d_Y| > K_{XY}$ be the constraint between variables X and Y

$$f_{XY} := \begin{cases} \alpha & \text{if } C_{XY} \text{ is violated: } |d_X - d_Y| \leq K_{XY} \ (\alpha > 0 \text{ is the weight of the violated constraint}) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

- F: Let f_{XY} be the cost function between two variables X and Y, then the objective function F is the sum of all the cost functions between each 2 variables variables: $\sum_{XY} f_{XY}$

3 DPOP on simple instance of FAP

In this problem, we will have the following options:

- Radio links: $link_1$, $link_2$, $link_3$

Link	Domain
$link_1$	{1,3,5}
$link_2$	{1,3}
$link_3$	{1,3,5}

- The constraints are as following:

First Link	Second Link	Constraint	Violation Weight
$link_1$	$link_2$	$\|f_1 - f_2\| > 2$	200
$link_2$	$link_3$	$\|f_2 - f_3\| > 2$	100

The following diagram illustrates the problem:

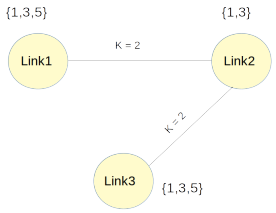


Figure 1: FAP problem

3.1 DFS Tree Construction

The tree construction phase have the following steps:

1. By convention, start from $link_1$.
2. $link_1$ adds its id to the token and send it to its only neighbor $link_2$
3. $link_2$ receives the token, it marks $link_1$ as its parent: $\text{parent}(link_2) = link_1$
4. $link_2$ adds its own id to the token and send it to its only neighbor $link_3$
5. $link_3$ receives the token, it marks $link_2$ as its parent: $\text{parent}(link_3) = link_2$
6. $link_3$ has no neighbors, it sends the token back to its parent $link_2$
7. All neighbors of $link_2$ are visited, it removes its id from the token and sends it back to its parent $link_1$
8. All neighbors of $link_1$ are visited, the algorithm terminates

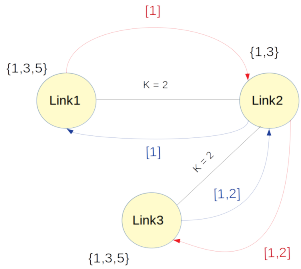


Figure 2: DFS Messages

The constructed DFS tree is the following:

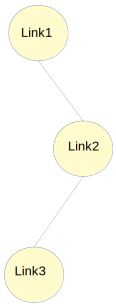


Figure 3: DFS Tree

3.2 Utility Phase

In this phase, the node of the tree adds and joins all the cost functions of its children and its parent, then it projects itself out of the resulting function and sends it to its parent. First, $link_3$ will send its cost function to its parent $link_2$.

The cost function of $link_3$ is demonstrated in the following diagram:

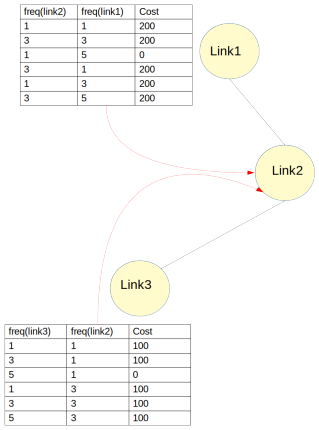
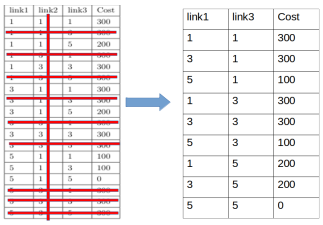


Figure 4: Util Phase Tables

After this step, $link_2$ adds and joins the cost functions (tables) and projects itself out of the resulting table, with deleting the redundant rows, taking into consideration the rows having less cost. The following table shows the addition of the two preceding cost functions:

link1	link2	link3	Cost
1	1	1	300
1	1	3	300
1	1	5	200
1	3	1	300
1	3	3	300
1	3	5	300
3	1	1	300
3	1	3	300
3	1	5	200
3	3	1	300
3	3	3	300
3	3	5	300
5	1	1	100
5	1	3	100
5	1	5	0
5	3	1	300
5	3	3	300
5	3	5	300

After projecting $link_2$ out of the tables, and deleting the duplicate rows keeping the minimum cost, the table will look like the following:



link1	link3	Cost
1	1	300
3	1	300
5	1	100
1	3	300
3	3	300
5	3	100
1	5	200
3	5	200
5	5	0

Figure 5: Result function

3.3 Value Phase

The root $link_1$ will search for the value that minimizes the cost, which is 5, and informs $link_2$ about this value. $link_2$ will find the value that minimizes the received cost function, which is 1. After that, $link_3$ will choose the frequency 5 in order to minimize the cost (cost = 0).

4 Analysis and Comments

The result for the above problem is the following:

- freq($link_1$)=5
- freq($link_2$)=1
- freq($link_3$)=5

We can see that this will be the best solution in order to minimize the cost. Because by this assignment, all the constraints are satisfied and therefore no constraints are violated and the costs will all be 0 for each of the 2 links. Going back to the FAP problem, the result can be translated as there is no links that will interfere which each others and all the links satisfy the condition that says that the

absolute value of the difference between 2 links should be greater than the given constant K corresponding to these 2 links.

5 Translator

A translator is done in order to parse the CELAR, GRAPH, and SUBCELAR6 instances, then convert them to XCSP files. The programming language used to do this work is Java. The translator is a command line tool that accepts one argument which is the path of the directory containing the 3 files: **var.txt**, **dom.txt**, and **ctr.txt**. After that, we use this tool to convert all the instances of CELAR, GRAPH, and SUBCELAR6. The generated XCSP files can be found in the folder named "xcspInstances".

6 FRODO Expriments

In order to compare the performance of several DCOP algorithms, we run these algorithms on different problem instances using FRODO. Then we gather some statistics about the performance of each algorithms.

6.1 Generation of Random Problems

In FRODO, one can interact with the user interface to upload his problem and configuration file in order to solve a specific instance. Also, one can write a python script that generates random instances of a specific problem, and then gather some statistics about the different algorithms that ran on the generated problem.
In this case, we generate several random problems, by changing the number of variables, number of constraints, and the complexity of the constraints. After that, we gather some statistics about 5 algorithms we saw during the lectures:

- ADOPT
- DPOP
- DSA
- MaxSum
- MGM

6.1.1 DSA-MaxSum-MGM

The first experiment is done between 3 algorithms (DSA, Max-Sum, and MGM). All the 3 algorithms ran on the same problems, each problem having different complexity.

In the following graph, we can see the effect of the variation of number of messages with respect to number of variables for the three algorithms.

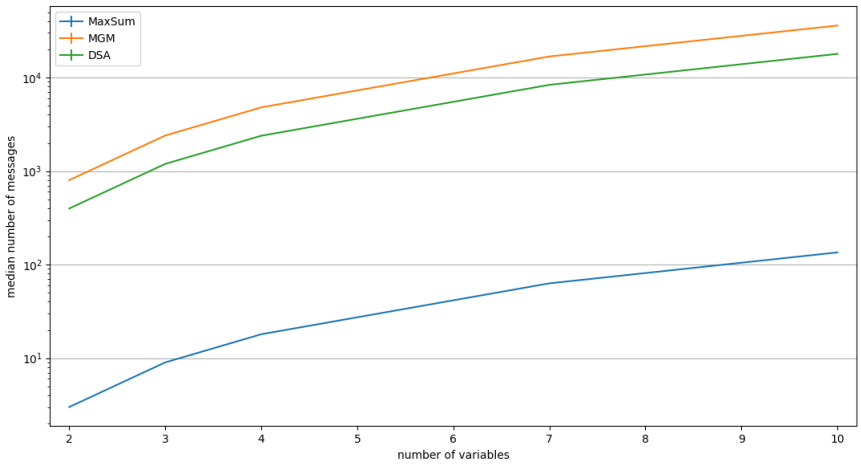


Figure 6: Variation of number of messages with respect to number of variables

Analysis: We can see from the above graph that when the number of variables is small the MaxSum algorithm needs less number of messages than MGM and DSA algorithms. As the number of variables increase, the comparison stays the same where the number of messages of each algorithm increases, but MaxSum algorithm will always have less number of messages than the 2 other algorithms. **We can conclude that MaxSum algorithm is more efficient than MGM and DSA regarding the number of messages sent.**

In the same experiment, we draw another graph that shows the variation of the time needed to complete the algorithm with respect to the complexity of the constraints of the problem (p2 value).

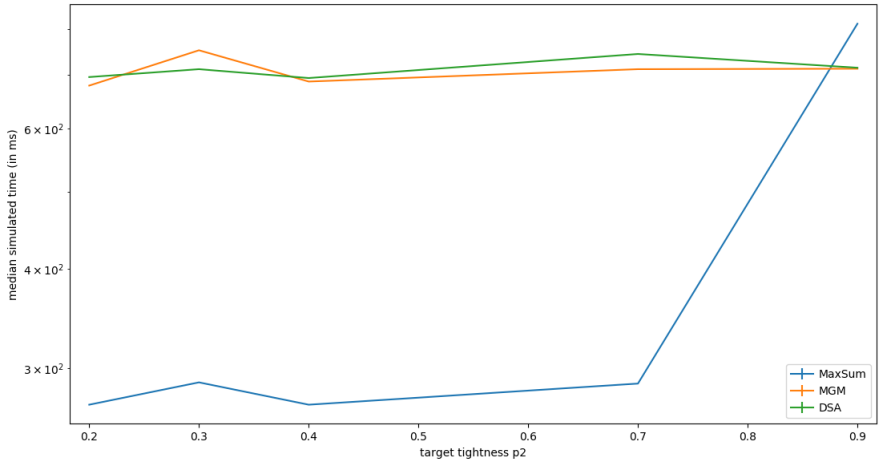


Figure 7: Variation of the time to solve the problem with respect to the complexity of the constraints

Analysis: From the above graph, we can see that at the beginning of the experiment, where the complexity was between 0.2 and 0.7, MaxSum Algorithm needed less time to complete the algorithm than the other 2. As the complexity of the constraints increases from 0.7 to 1,the time of MaxSum algorithm increases exponentially where the other algorithms completed the problem with their normal time. **We can conclude that MaxSum is more efficient regarding the simulation time when the problem is simple, but as the problem gets more complex, MGM and DSA perform better.**

The last statistics gathered are the efficiency of each algorithm regarding the cost. Here is a graph showing us how does the cost of the algorithms varies according to the number of variables.

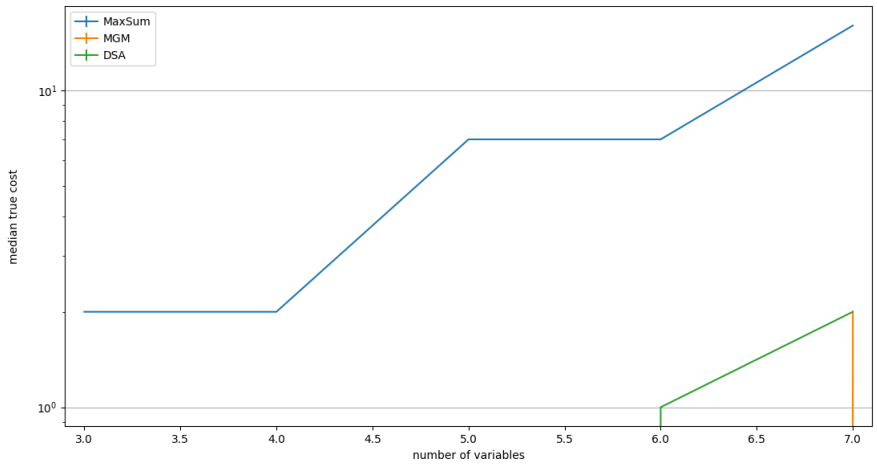


Figure 8: Variation of cost with respect to the number of variables

From the above graph, we can realize that MaxSum algorithm doesn't give good solutions if we

compare it with MGM and DSA, and that explains the fact that it doesn't take so much time and doesn't have a lot of messages as we see in the above 2 graphs. The other two algorithms gives a low cost comparing it to the cost of the MaxSum algorithm. **We can conclude that MGM and DSA algorithms gives in general better solutions than MaxSum algorithm**.

6.1.2 ADOPT-DPOP

The second experiment is done between ADOPT and DPOP algorithms. The reason why 2 experiments are done is that ADOPT and DPOP algorithms are slow for big instances where they usually cause a timeout, while DSA, MaxSum, and MGM are fast and can handle instances larger than that of ADOPT and DPOP. In this experiment, the problems are less complex than before, and statistics are gathered to compare between the efficiency of ADOPT and that of DPOP.

The following graph shows us the variation of the number of messages of DPOP and ADOPT with respect to the complexity of the problem.

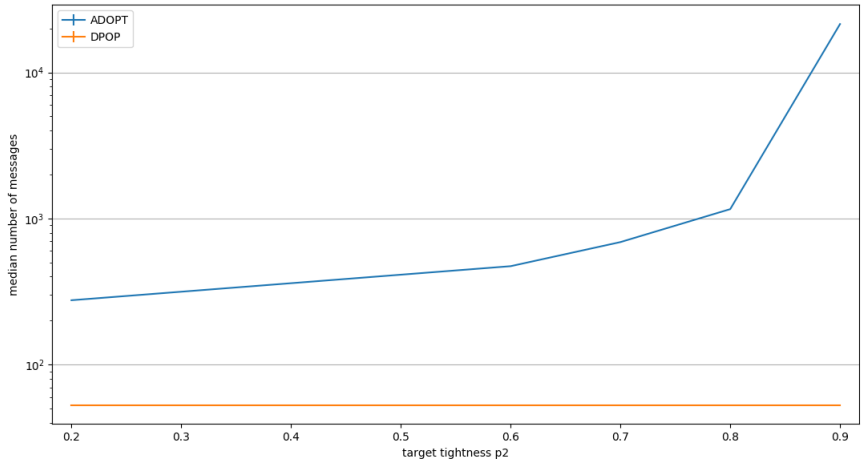


Figure 9: Variation of cost with respect to the complexity of the problem

As we can see in the above graph, the number of messages sent in the DPOP algorithm is always less than that of ADOPT algorithm. As the complexity increase, the number of messages of both algorithms will increase, but the interesting point is that the number of messages of ADOPT will increase exponentially, while that of DPOP increases linearly.

The next graph shows the variation of simulated time with respect to the complexity of the problem of the 2 algorithms ADOPT and DPOP.

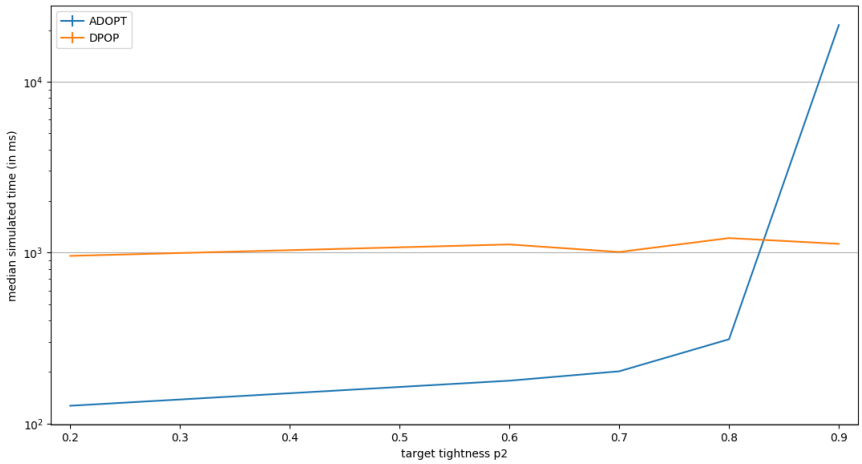
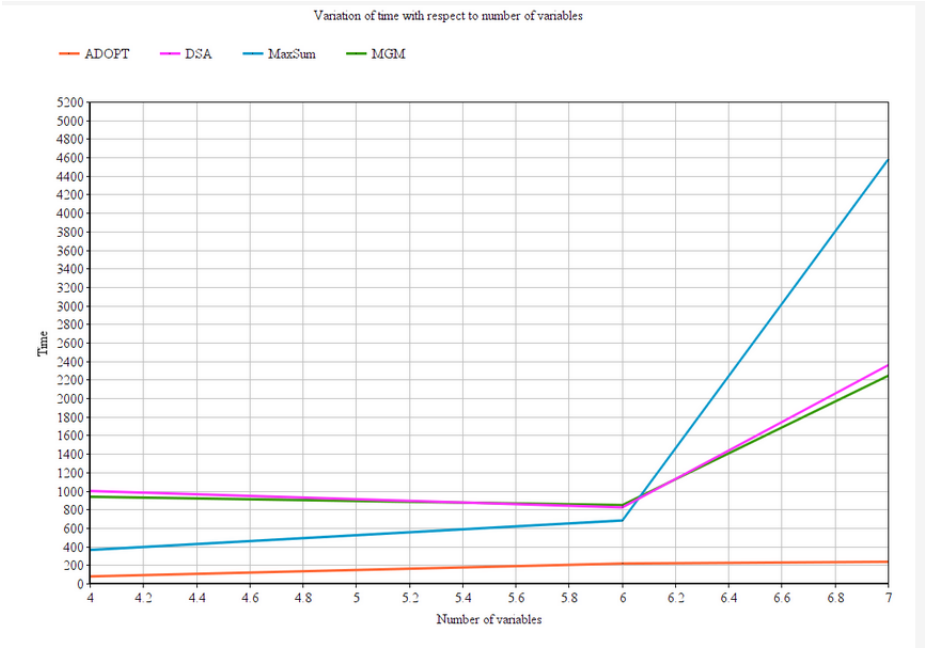


Figure 10: Variation of simulated time with respect to the complexity of the problem

In the above graph we can see that at the beginning of the experiment, the time of ADOPT was less that that of DPOP. As the complexity of the problem is increases, the time of ADOPT increases exponentially while that of DPOP increase manually.

6.2 Frequency Assignment Problem

There are some FAP instances given by Thomas Schiex. These are very large instances where the number of variables varies between 200 and 1000. Addition to that, they contain hard complex constraints. Solving these large instances using FRODO would need some days or even months for any algorithm. In this section, we edited these large instances in order to decrease the number of variables and make the problems less complex. Some FAP instances are generated, and we tested the performance of some algorithms on these problems. We came up with the following graph:



In the above graph, we can see that the as the number of variables of hard constrained problem increase, the time of ADOPT increases linearly, while the time of the other algorithms increases exponentially.

7 References

https://manual.frodo-ai.tech/FRODO_User_Manual.html#x1-66000B

http://www7.inra.fr/mia/T/schiex/Doc/CELAR.shtml#hints

https://www.onlinecharttool.com

https://www.emse.fr/~picard/cours/mas/MAS-project-2019.pdf