

# INFO-F424 – COMBINATORIAL OPTIMIZATION PROJECT

Renaud Chicoisne  
renaud.chicoisne@ulb.be

Cristian Aguayo  
cristian.aguayo.quintana@ulb.be

In this project, we focus on the *Interpreters Scheduling Problem* (ISP). Language interpreters play a key role in the context of effective communication in multilingual/multicultural contexts. Applications of ISP range from legal proceedings or healthcare to political interactions, e.g. at the *European Parliament* or the *United Nations*. For this project we will consider a case inspired by the latter examples.

Let us consider a set  $\mathcal{I}$  of interpreters, who work for an institution in which a set  $\mathcal{L}$  of languages are spoken. Each interpreter  $i \in \mathcal{I}$  is proficient in – i.e. is qualified to interpret – a set of languages  $\mathcal{L}_i \subset \mathcal{L}$  with  $|\mathcal{L}_i| \geq 2$ . In this institution, different types of sessions are held from monday to friday: they form a set  $\mathcal{B}$  of one-hour blocks, with  $|\mathcal{B}| = 40$  (5 days of 8 working hours each). During each block  $b \in \mathcal{B}$ , a set  $\mathcal{S}_b$  of simultaneous sessions are held in different rooms. The speakers participating in a session are free to speak in their mother tongue which – we assume here – is the only language they speak. Let  $\mathcal{S} = \cup_{b \in \mathcal{B}} \mathcal{S}_b$  be the full set of sessions, and  $\mathcal{U}_s \subseteq \mathcal{L}$  the set of languages used in session  $s \in \mathcal{S}$ .

Naturally, each interpreter  $i \in \mathcal{I}$  assigned to a session  $s \in \mathcal{S}$  must perform interpretations between two languages  $l_1$  and  $l_2$  that are 1) known to her and 2) needed in the session, i.e.  $(l_1, l_2) \in (\mathcal{L}_i \cap \mathcal{U}_s)^2$ . Language interpreting is done in near real-time during the sessions: In consequence, an interpreter *is not able* to interpret for more than one pair of languages in a same session. We say that a language pair  $(l_1, l_2) \in \mathcal{L}_s^2$  is *covered* for session  $s \in \mathcal{S}$ , if  $s$  was assigned an interpreter  $i \in \mathcal{I}$  – who is proficient in both  $l_1$  and  $l_2$  – to interpret between  $l_1$  and  $l_2$ .

In the example presented in Figure 1, if we assign **Interpreter 001** and **Interpreter 002** to **Session 001**, we can cover the pair (UK, ES) and the pair (FR, ES). Note that we can obtain the same result by choosing **Interpreter 005** instead of **Interpreter 002**.

Session 00s	$\mathcal{U}_s$	Interpreter 00i	$\mathcal{L}_i$
Session 001	{UK, FR, ES}	Interpreter 001	{UK, ES}
Session 002	{ES, ES}	Interpreter 002	{FR, ES}
Session 003	{UK, ES}	Interpreter 003	{ES, DE}
		Interpreter 004	{ES, ES}
		Interpreter 005	{FR, IT, ES}

Figure 1: Example instance for the ISP problem

## 1 A simple IP

Assuming that the interpretation is *on-site*, an interpreter cannot assist to more than a single session during each block  $b \in \mathcal{B}$ .

1. Define proper sets of variables to represent the assignment of interpreters to sessions and pairs of languages, and model all the logical constraints presented within the statement. These logical constraints will be included in all the upcoming models.
2. Define proper sets of variables and constraints to represent the following objective functions:
  - (OF1) Maximize the total number of language pairs covered over all the sessions.
  - (OF2) Maximize the number of sessions in which all the language pairs are covered.

3. For each objective function, implement the IP in **Gurobi**.
4. Report objective values, solving times. Present a comparison –e.g, a graphical comparison – of the optimal assignment variables obtained with (OF1) and (OF2) for at least one instance of your choice.

## 2 Some operational constraints

So far we only considered logical constraints in our formulations, but in a real life problem there may be other operational conditions to be taken into account.







1. Model the following operational constraints using your previously defined decision variables, as well as new decision variables if you deem it convenient:
  - (OC1) Each interpreter  $i \in \mathcal{I}$  can be assigned to at most  $M_i = 15$  sessions.
  - (OC2) Each interpreter  $i \in \mathcal{I}$  cannot work during more than  $CB_i = 3$  consecutive blocks.
2. Optimize (OF1) and (OF2) subject to the logical constraints and all the new operational constraints using **Gurobi**.

## 3 Bridging

While solving the basic models for ISP, you may have noticed that some language pairs are rarely covered, or even some of them are never covered. This is because we imposed that the only way to cover a language pair in a session was by assigning an interpreter who was proficient in both languages to that session. In practice, this condition can be relaxed through *bridging*: A *bridge* between the language pair  $(l_1, l_2)$  for a session  $s$  through the language  $l'$  can be defined as follows:  $l'$  bridges the pair  $(l_1, l_2)$  if either:

- $l' \in \mathcal{L}_s$  and there is an interpreter assigned to  $(l_1, l')$  and another assigned to  $(l', l_2)$ , or
- $l' \notin \mathcal{L}_s$  and we assign an interpreter that is proficient in  $(l_1, l')$  and another that is proficient in  $(l', l_2)$ .

Defining a bridge between a pair  $(l_1, l_2)$  allows to cover it. In the situation depicted in Figure 1 we can find examples for both of the aforementioned cases:

- If **Interpreter 001** and **Interpreter 002** are assigned to **Session 1**, the pair (, ) can be covered through a bridge defined by .
- If we assign **Interpreter 003** and **Interpreter 004** to **Session 2**, we can cover the pair (, ) thanks to the bridge made by .

Remark that an interpreter cannot be part of several bridges. With this new tool at hand:

1. Define proper constraints to represent bridging. If needed, you can add new sets of variables or modify the existing ones, but please note that some logical constraints may be reformulated depending on how you define the new decision variables.
2. Optimize (OF1) and (OF2) subject to the logical constraints, the operational constraints and the bridging conditions.

## 4 Implementation and evaluation of models and algorithms

Test the proposed models, as well as the algorithms you could propose on different instance sizes:

- **example.json**: The instance depicted in Figure 1. This instance is useful to debug your code, as you should be able to solve it by hand, without using **gurobi**.
- **isp-S[#\_of\_Sessions]-I[#\_of\_Interpreter].json**: These instances have  $|\mathcal{B}| = 40$  blocks,  $|\mathcal{L}| = 24$  languages, and different set values for  $|\mathcal{S}|$  and  $|\mathcal{I}|$ .

For each instance size and each model/algorithm, report the objective value, MIP gap and the runtime. You are allowed to set a `TimeLimit` of **at least 10 minutes** in case you are experimenting high solving times. Please include this in your report, as well as all cases where your PC cannot compile a model due to lack of memory. Taking a look at the instances through an exploratory data analysis may give you interesting insights and may help you provide better discussions on the results.

## Guidelines and Evaluation

### Guidelines

- The project must be carried out by groups of no less than two people and no more than three people.
- The required models and algorithms must be implemented in `python` or `julia` along with the solver `Gurobi` when necessary. To use `Gurobi`, you have to use the `gurobipy` package for `python` and the `JuMP` package for `julia`. You can use other packages to read and manage files.
- The implementations must be done in **one and only one** of the mentioned programming languages.
- You are asked to prepare a paper-like report which may include:
  - An introduction to the Interpreters Scheduling Problem (do **NOT** copy and paste the statement) **including at least 5 appropriate bibliographic references** related to scheduling problems
  - The answers to the theoretical questions presented in this statement
  - Results and resolution times obtained for each model/algorithm, and discussion/conclusions about the different results.

### Evaluation

- You must submit the report in `.pdf` format, as well as the `.py` or `.jl` files used to solve the problem along with their respective instructions for use. Reports made on `jupyter notebook` (`.ipynb` files) are also welcome.
- The report must be written in english and it must contain the names and last names of the members of the group, and their respective ULB ID numbers.
- Please **DON'T** upload the instance files. Just clarify in which directory should the instance files must be placed to run your code.
- Verify that your codes work correctly before submitting them.
- The deadline is on June 13th, 2025 at 23:59 Brussels time.
- Any disrespect of project and/or submission guidelines will lead to penalization.