1-

Good evening …

In this talk I will briefly present **CP Optimizer** which a generic optimization engine with a particular focus on solving industrial scheduling problems.

2-

We started developing **CP Optimizer** in 2007 ... some years before our company, ILOG, became acquired by IBM.

At that time we already had a lot of experience with our previous generation tools (ILOG Solver and Scheduler) to apply Constraint Programming technology for solving the real-life optimization problems of our customers, and in particular scheduling problems.

But these tools were complex to use and that’s why we decided to develop a new tool, CP Optimizer, to make it easier to people with **no particular background** on AI, or CP or Operations Research to solve complex scheduling problems.

3-

So, CP Optimizer follows a Model & run approach:

The user focuses on providing a **declarative model** of the optimization problem, using the classical ingredients of combinatorial optimization: variables, constraints, expressions, objective function …

And the resolution is performed by an automated search algorithm so that the user do not have any resolution code to write.

The search is an exact algorithm that can provide optimality proofs,

It is deterministic (solving twice the same problem will result in the same solution)

It is usually anytime (in the sense that finding an initial feasible solution is in general fast)

And it is continuously improving from version to version ...

4-

So … you will say, there are existing optimization frameworks that work this way like typically **Mixed Integer Linear Programming**, or **Constraint Programming** tools.

From our point of view, these tools have some limitations when dealing with scheduling problem in that they mostly deal with **numerical decision variables** (integer or floating point).

When you are dealing with scheduling problems, you are dealing with a particular dimension that is the "temporal dimension" and in this context there are some other mathematical concepts that naturally emerge like:

Intervals for representing an interval of time, an activity for instance

Functions to represent the evolution of some quantity or state variable over time

Permutations of intervals if you are dealing with unary resources

Optional intervals if you are considering some events that may or may not occur in your schedule.

5-

So the idea of CP Optimizer is to integrate these simple mathematical concept in the model …

While keeping all the good ideas of other well established optimization frameworks like MILP, in particular:

- The **Model & run** approach of course
- The **completeness** of the search
- Some **input/ouput format** to understand and exchange the problem instances
- The possibility to use you favorite **programming language** to define the model (C++, Python, Java, OPL)
- Some tools to **help developing models** like **warnings** or e**xplanations for infeasibility** of a model
- The possibility to inject an existing solution as a **starting point** to the search,
- Etc …

6-

So let' see how a very classical academical scheduling problem can be formulated in CP Optimizer … I will take the example of the **Resource-Constrained Project Scheduling** that you probably already know.

The problem consists in scheduling a set of **tasks** with p**recedence constraints** between the task and **resource requirements**. Each task requires some quantity of **cumulative resources** that have a **limited capacity.** The problem is to minimize the **makespan** of the schedule.

7-

Here is the mathematical formulation of the problem in CP Optimizer.

You see that the model is using **interval variables** a_i to represent the **tasks**.

It posts some **precedence constraints** between these decision variables.

And creates a **function** that describes the evolution of the resource usage over time and post a **constraint** on the maximum value of this function.

It is difficult to think of a more compact model …

8-

We can see this model formulated in OPL and a solution for a particular instance of this problem.

The **OPL** formulation is a **one-to-one** mapping of the **mathematical formulation**. And you can easily write the same model in **Python**, in **Java** or in C++.

9-

Of course, I said that the main target of CP Optimizer are i**ndustrial problems**, **not academical ones**. But nevertheless, CP Optimizer is **very good** at solving these classical scheduling problems.

Some yours ago we published a paper summarizing some results were we **closed** and **improved** many classical instances of scheduling problems.

I have to say that one of the reasons we do not particularly focus on these instances is that they are usually **small** compared to the real industrial problems we face.

For instance the largest size of problems in classical RCPSP benchmark is **120 or 300 tasks**. In real problems we once had to solve a real world RCPSP-like problem with 1.000.000 tasks. That was in the **aircraft assembly** domain. Most of the existing approaches developed for small problems won't scale to this size. Whereas CP Optimizer can and, for this type of simple scheduling problem, can produce solutions in **almost-linear time** with respect to problem size.

10-

But in the real world, scheduling problems are often much more complex that academic ones

In particular:

- **Activities** and **resources** are much more **complex**. Often there are different ways for executing an activity. Also, resources are complex machines that for instance may process the activities by batches of limited capacity, with setup times, etc.

- The **objective function** is never as simple as a makespan. Most of the times several criterions need to be considered involving resource costs, tardiness costs, etc.

- Problems are often **over-constrained**, not all activities can be executed and part of the problem is to decide which subset of activities to execute

- Decision variables are **heterogenous** and involve at the same time the decision to perform or not an activity, the start/end time of activities, the allocation of resources, the grouping of activities into batches, etc…

- Problems are **large** ...

- … And require **fast solving time** (for instance for the operational scheduling of a **fabrication plant in the semiconductor industry**, a rescheduling occurs every **5 minutes** …)

11-

The **modeling concepts** of CP Optimizer make it easier to formulate **complex problems**. For instance the following things are easy to model :

- **Variable** activity **duration**

- **Oversubscribed** problems that involve optional activities

- **Hierarchical** problems with a **work breakdown structure** and alternative ways to perform some tasks in the hierarchy

- Different types of cumulative resources like **inventories**, **reservoirs**

- Activities that can be **batched** and processed together by a resource

- Unary resources with sequence-dependent **setup times** and **costs**

12-

Here is the complete model (in OPL) of a pretty realistic problem in the context of semi-conductor manufacturing.

I will not enter into the details but the problem involves **scheduling** a certain number of **lots**. Each lot is composed of several **steps** that need to be **allocated on a machine**. Machines can **group** steps of the same family (this is the color of the step on the picture) into **batches** and require some **setup time** between batches. There are some **max waiting time** between some consecutive steps that must be satisfied as much as possible and a second optimization criterion is a **total tardiness cost**.

You see **on the right side** the CP Optimizer model … Each of the components of the problem naturally maps into some modeling concept.

13-

We **improve** the automated search algorithm of CP Optimizer f**rom version to version**. On this plot, you can have an idea of the average speed of the different versions across a **large number of different types of scheduling problems** that we keep in our benchmark. And you can also see the main ideas we introduced in the search algorithm that helps improving the performance … Some are described in papers (like **Failure-Directed Search** or **Objective Landscapes**).

14 -

If you are **curious** to see what happens **under the hood**, most of it is also described in papers, I will give the references in my last slide. Our main philosophy is to consider everything that can be useful both in Operations Research and Artificial Intelligence …

15-

Here are some papers about CP Optimizer …

16-

So, thanks for listening … I just want to add that the **full version** of **CP Optimizer** can be downloaded **free of charge** if you are an academic and use it for teaching or for research.