Quantum algorithms for solving hard combinatorial optimization problems in the field of “smart-charging” of electrical vehicles

M. Porcheron, EDF-R&D division

Plan

1. Smart-charging of electrical vehicles
2. From smart-charging problems to graph-theory problems
   thanks to an old and fruitful field of Operational Research : *scheduling*
3. From graph-theory problems to quantum algorithms

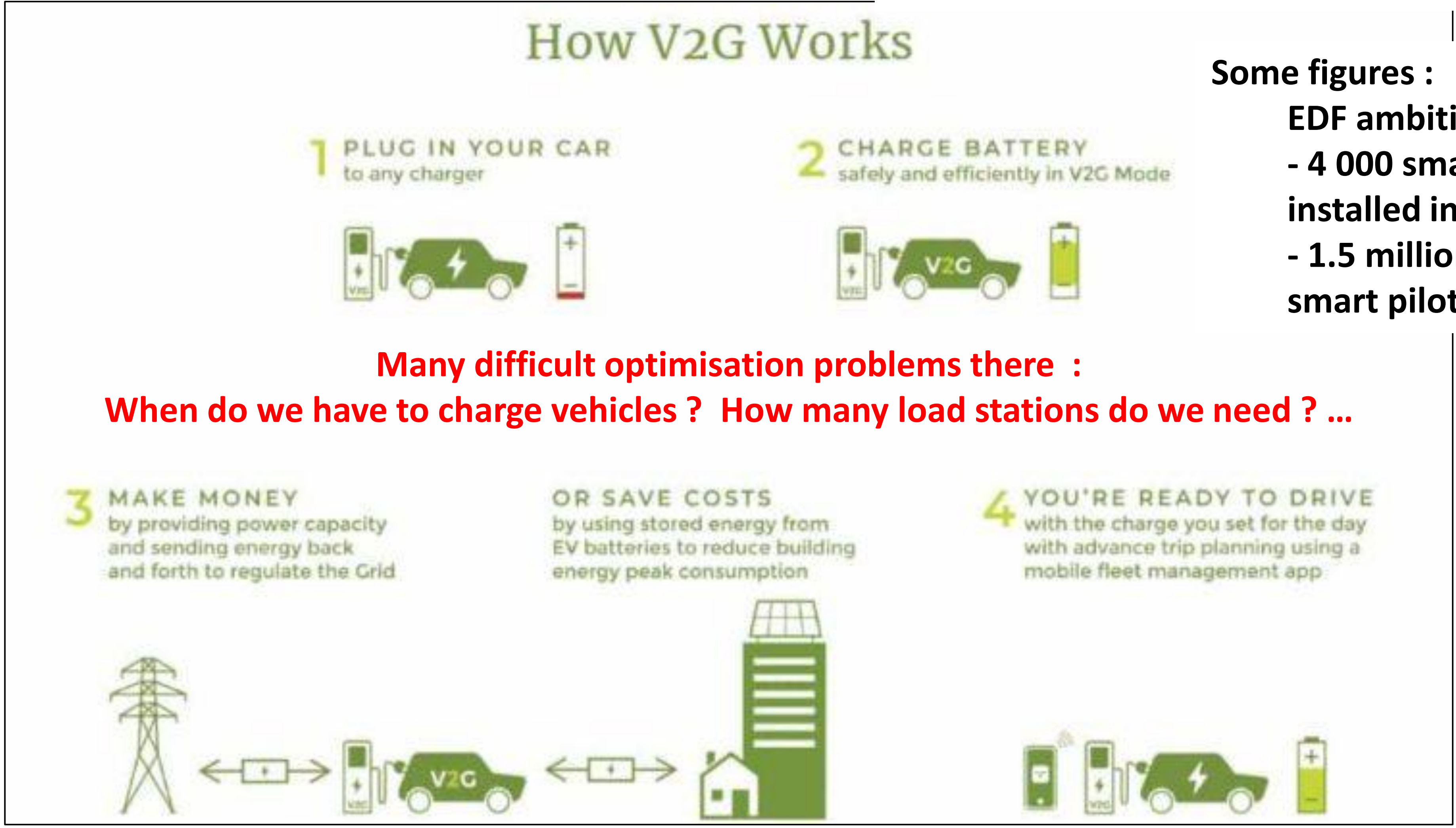
Smart-Charging of Electrical Vehicles

Smart-Charging

- Technologies that aim at optimizing charges/discharges of electrical vehicles
- « V1G » : form grid to vehicle
  - Power is delivered in an unidirectional manner from grid to vehicle, to charge its battery
- **Vehicle To Grid (« V2G ») : in both ways**
  - Energy stored in accumulators can also serve to power a building, or to regulate the grid
- Many constraints here !
  - The high level of power required to load electrical vehicles, especially on fast load stations, compels to optimally modulate the load demand in time
  - While satisfying needs of users, charging/discharging cycles of batteries, limits on available power delivered by the grid, reserves required to guaranty frequency stability, etc.

Smart-Charging

https://les-smartgrids.fr/dreev-edf-smart-charging-v2g/



Some figures :

EDF ambition in Europe :

- 4 000 smart load-stations installed in 2020

- 1.5 million of vehicles with a smart piloted load in 2035

Many difficult optimisation problems there :

When do we have to charge vehicles ? How many load stations do we need ? …

From smart-charging problems to graph-theory problems

thanks to an old and fruitful field of Operational Research : *scheduling*

Scheduling

- **J = {1 … n } *jobs* to execute on I = {1…m} *machines***
  - At a given time step, one job performs on a single machine and a machine can only execute a single job
- **A scheduling problem is described by a triplet : $\alpha|\beta|\gamma$**
  (Graham and Lawler classification [Graham Lawler et al 79])
  - α : machine environment : single/multiple, parallel, uniform …
  - β : job characteristics : splitting (pre-emption) allowed or not, resource or precedence constraints, due dates …
  - γ : criteria to be minimized : total completion time, global makespan, lateness …
- **Examples**
  - **1|prec|Lmax :** ***minimise maximum lateness on a single machine, subject to precedence constraints on the jobs***
  - **R|pmtn|$\sum C_j$:** ***minimise the total completion time on a variable number of unrelated machines, allowing pre-emption***
- **A huge bunch of applications …**
  - Manufacturing industry (job shop scheduling), logistics (timetables, project scheduling), transport (fleet and crew management), computing (jobs scheduling on parallel machines, cloud management …)
- **… and around sixty years of researches on the subject !**

Scheduling

- **Complexity**
- P : problems solvable in polynomial « time » (number of instructions) in the size of their data
- « Easy » tractable problems

604 E Complexity Classification of Deterministic Scheduling Problems

SINGLE MACHINE	PARALLEL MACHINES	SHOPS
$1 \mid r_j, p_j = 1, prec \mid \sum C_j$	$P2 \mid p_j = 1, prec \mid L_{\max}$	$O2 \mid\mid C_{\max}$
$1 \mid r_j, prmp \mid \sum C_j$	$P2 \mid p_j = 1, prec \mid \sum C_j$	
$1 \mid tree \mid \sum w_j C_j$		$Om \mid r_j, prmp \mid L_{\max}$
	$Pm \mid p_j = 1, tree \mid C_{\max}$	
$1 \mid prec \mid L_{\max}$	$Pm \mid prmp, tree \mid C_{\max}$	$F2 \mid block \mid C_{\max}$
$1 \mid r_j, prmp, prec \mid L_{\max}$	$Pm \mid p_j = 1, outtree \mid \sum C_j$	$F2 \mid nwt \mid C_{\max}$
	$Pm \mid p_j = 1, intree \mid L_{\max}$	
$1 \mid\mid \sum U_j$	$Pm \mid prmp, intree \mid L_{\max}$	$Fm \mid p_{ij} = p_j \mid \sum C_j$
$1 \mid r_j, prmp \mid \sum U_j$		$Fm \mid p_{ij} = p_j \mid L_{\max}$
$1 \mid r_j, p_j = 1 \mid \sum w_j U_j$	$Q2 \mid prmp, prec \mid C_{\max}$	$Fm \mid p_{ij} = p_j \mid \sum U_j$
	$Q2 \mid r_j, prmp, prec \mid L_{\max}$	
$1 \mid r_j, p_j = 1 \mid \sum w_j T_j$		$J2 \mid\mid C_{\max}$
	$Qm \mid r_j, p_j = 1 \mid C_{\max}$	
	$Qm \mid p_j = 1, M_j \mid C_{\max}$	
	$Qm \mid r_j, p_j = 1 \mid \sum C_j$	
	$Qm \mid prmp \mid \sum C_j$	
	$Qm \mid p_j = 1 \mid \sum w_j C_j$	
	$Qm \mid p_j = 1 \mid L_{\max}$	
	$Qm \mid prmp \mid \sum U_j$	
	$Qm \mid p_j = 1 \mid \sum w_j U_j$	
	$Qm \mid p_j = 1 \mid \sum w_j T_j$	
	$Rm \mid\mid \sum C_j$	
	$Rm \mid r_j, prmp \mid L_{\max}$	

Table E.1 Polynomial Time Solvable Problems [Pineddo 2008]

Scheduling

- **Complexity**
- *NP* : problems for which no polynomial algorithm is known, but such that a solution can be *verified* in polynomial time
- *NP-Hard* : problems to which any problem in *NP* can be reduced in polynomial time
- *NP-Complete* : *NP-Hard* problems in *NP*

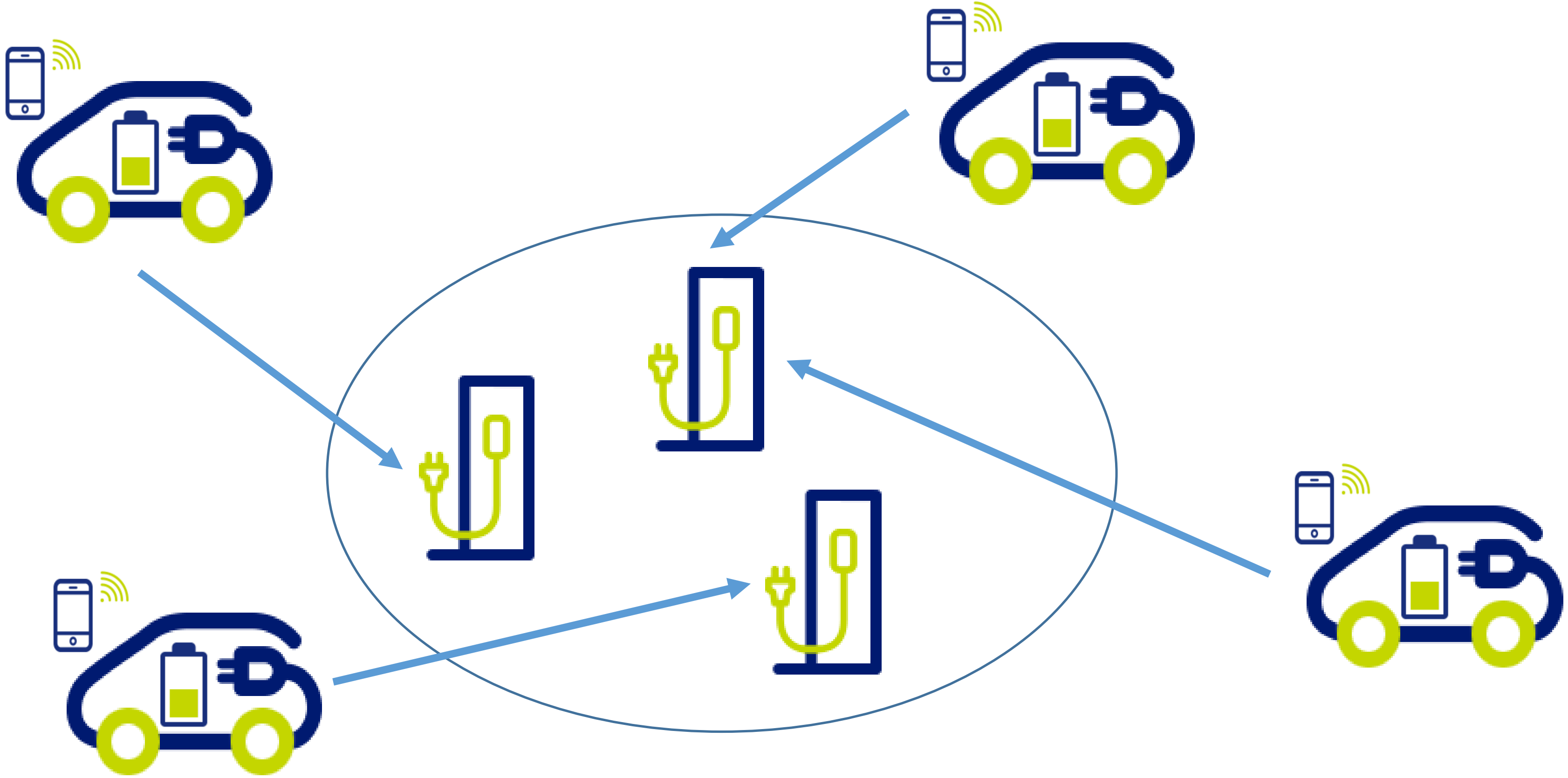
SINGLE MACHINE	PARALLEL MACHINES	SHOPS
$1 \mid\mid \sum w_j U_j \quad (*)$ $1 \mid r_j, prmp \mid \sum w_j U_j \quad (*)$ $1 \mid\mid \sum T_j \quad (*)$	$P2 \mid\mid C_{\max} \quad (*)$ $P2 \mid r_j, prmp \mid \sum C_j$ $P2 \mid\mid \sum w_j C_j \quad (*)$ $P2 \mid r_j, prmp \mid \sum U_j$ $Pm \mid prmp \mid \sum w_j C_j$ $Qm \mid\mid \sum w_j C_j \quad (*)$ $Rm \mid r_j \mid C_{\max} \quad (*)$ $Rm \mid\mid \sum w_j U_j \quad (*)$ $Rm \mid prmp \mid \sum w_j U_j$	$O2 \mid prmp \mid \sum C_j$ $O3 \mid\mid C_{\max}$ $O3 \mid prmp \mid \sum w_j U_j$

Table E.2 NP-Hard Problems in the Ordinary Sense

[Pineddo 2008]

Scheduling

- A very (very) large number of conventional algorithms are available
  - *Exact* in the (pseudo-)polynomial case (e.g. dynamic programming), or for reduced instances in strong NP (e.g. Branch&Bound for linear formulations)
  - *Approximate* : based on linear or semi-definite positive relaxations
  - *Probabilistic,* in general in *BPP (Bounded-error Probabilistic Polynomial time*) : probability of success $\geq 2/3$, probability of fail $\leq 1/3$
  - *Heuristic* : greedy algorithms, genetic algorithms, local search, constraint programming…
- What about quantum algorithms?
  - **Well, to begin, they'll have to challenge the above dream team of conventional algorithms !**
  - *Grover* : quadratic speedup on any problem in NP with respect to a "brute force" exhaustive search
  - Many scheduling problems can be formulated as Binary Quadratic Optimisation Problems (QUBO) ➔ *Quantum Annealing (QA), Quantum Adiabatic Computing (QAA)* and *Quantum Approximate Optimisation Algorithm (QAOA)* are good candidates
- Scheduling is often a matter of graphs …



From smart-charging problems to graph-theory problems

Minimization of total charging time ➔ Max-Cut

Minimization of the number of charging stations➔ Max Independent Set

Modelling

- J = {1 … n } jobs of charge of *n* electrical vehicles, on a set I = {1…m} of « parallel » charge stations
- The completion time of a load *j* is noted C_j. We try to minimize the total time of completion of the charges $\sum_{j \in J} w_j C_j$, where w_j represents a non-negative integer *weight* associated with job *j* measuring its *importance/priority*
  - *For example, we want to prioritize the charge of safety-related intervention vehicles.*
- In standard scheduling notation, this is : $P_m || \sum_{j \in J} w_j C_j$

Hypotheses

- Each load must be run on a station and can be on any of them, and a station can only perform one charge at a time
- Stations are considered *identical :* **the charging duration p_{ij} of vehicle *j* on station *i* is the same whatever the station is, i.e. $p_{ij} = p_j$**
- We neglect possible *resource constraints* (maximum number of charging stations operating in parallel, maximum number of loads performed by a station, for example) and *"early or late date" constraints* on the completion of the load jobs
- Load tasks are considered *non-preemptive*, i.e. can not be interrupted to be resumed later.
  - That is to say that a charge is entirely performed on the same station, without being interrupted.
  - Note that problems without preemption are generally more difficult than with (less degrees of freedom)

Complexity

- $\mathbf{1||\sum_{j\in J} w_j C_j}$: problem with one machine (and its derivatives) can be solved in *n.log(n)*
  - *Smith's Rule : schedule jobs in non-increasing order of w_j/p_j. Intuitively, this amounts to postponing the longest jobs at the latest (weighting the duration by the priority w_j); this avoids accumulating their durations in the sum of the completion times of the others*
- $\mathbf{P_m||\sum_{j\in J} C_j}$: problem with *m* identical parallel machines and $w_j = 1$, i.e. no "priority" on the jobs, can also be solved in *n.log(n)* by a generalization of the Smith's rule above
- $\mathbf{P_m||\sum_{j\in J} w_j C_j}$: NP-Hard !
  - Numerous classical approximation algorithms based on relaxations of the IP or SDP formulations, and on various (meta-)heuristics

Reduction to *Max-Cut* problems

- We notice that once the jobs are assigned to the machines, the optimal scheduling consists of scheduling the jobs on each machine according to the non-decreasing order given by p_j/w_j
- Thus, the optimal order to apply in any solution may be predetermined :
  $k \prec j$ iff $k \neq j$ and ${}^{p_k}/_{w_k} \leq {}^{p_j}/_{w_j}$
  - If $k \prec j$ *and* *k* and *j* are assigned to the same machine, then *k* will necessarily be processed before *j*.
- One can thus see any problem with *m* machines like the search for an optimum *m-partition* of all the jobs, taking into account this order
- In the 2-machine case, we search for an optimal partition in two subsets of the set of jobs

Reduction to *Max-Cut* problems

- In the 2-machine case : $\mathbf{P_2||\sum_{j \in J} w_j C_j}$
- Let *G=(V,E)* be the complete graph whose the *n* vertices in *V* correspond to the *n* jobs in *J*.
- We define a weight on each edge *(i,j)* by :

$$w_{ij} = \min\{w_i p_j ; w_j p_i\}$$

- This implements a total order relation on the jobs :

$k \prec j$ si $k \neq j$ et ${}^{p_k}/_{w_k} \leq {}^{p_j}/_{w_j}$

Reduction to *Max-Cut* problems

- We show that for every partition of *V* into two subsets (S, V\S) :

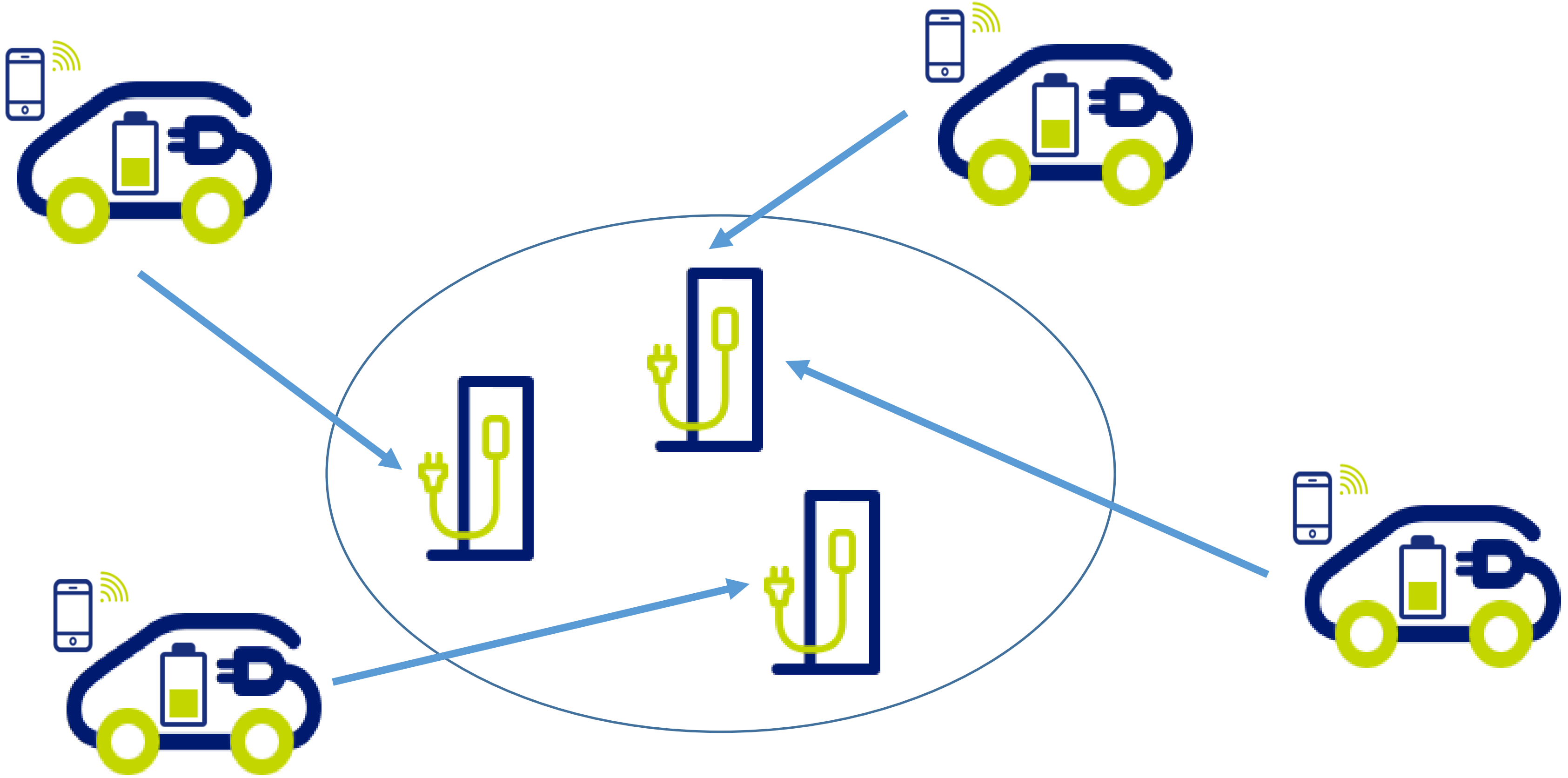
$$\sum_{1\leq i\leq j\leq n} w_{ij} + \sum_{j=1}^{n} w_j p_j = \sum_{j=1}^{n} w_j C_j + \sum_{i\in S, j\in V\setminus S} w_{ij}$$

  - $\sum_{1\leq i\leq j\leq n} w_{ij}$ is a *constant* term representing the sum of the weights of all the edges of *E*
  - $\sum_{j=1}^{n} w_j p_j$ is a *constant* term representing the sum of the weighted durations of all jobs in *V*
  - $\sum_{j=1}^{n} w_j C_j$ is the *total completion time* that we want to minimize
  - $\sum_{i\in S, j\in V\setminus S} w_{ij}$ is the weight of the edges of which one vertex is in S and the other in V\S, i.e. *the weight* of the *cut* associated with the partition/assignment (S, V\S)
- ➔ Minimising $\sum_{j=1}^{n} w_j C_j$ is thus equivalent to finding the *cut* (S, V\S) such that $\sum_{i\in S, j\in V\setminus S} w_{ij}$ is maximal

Reduction to *Max-Cut* problems

- $\mathbf{P_2 || \sum_{j \in J} w_j C_j}$ ⇔ Max-Cut !

- **The approach generalizes to *m* machines :**
- $\mathbf{P_m || \sum_{j \in J} w_j C_j}$ ⇔ Max-m-Cut !



From smart-charging problems to graph-theory problems

Minimization of total charging time → Max-Cut

Minimization of the number of charging stations → Max Independent Set

Interval Scheduling Problems

- A set of intervals representing tasks to be performed whose start dates are known in addition to their durations
  - Two intervals of tasks overlap if their intersection is not empty.
- A set of machines. Each machine can only perform one task at a time and is always available.
- A task runs only on one machine, and can not be interrupted to be resumed later, possibly on another machine (*no preemption*)
- The problem is to perform all the tasks using a minimum of machines, i.e. to find a task assignment to the machines such that no pair of tasks assigned to the same machine overlaps, while minimizing the number of machines used
  - basic version, many variants
- ~ Facility location / covering problems

Reduction to MIS problems

- Consider an *Interval graph* whose vertices are the tasks and such that there is an edge between two vertices if the intervals associated with their tasks overlap
- The basic version of the *interval scheduling problem* is to find a *coloring* of this graph, its *chromatic number* corresponding to the *minimum number of machines needed to schedule all the tasks*.
- Finding the *maximum stable (MIS)* of this graph is equivalent to finding the maximum set of tasks that can be executed on the *same* machine (no *overlapping*)
- Note that there are *approximate algorithms determining a coloring from an enumeration of MIS*

Modelling

- We consider a time horizon *T*
- We associate with each EV *v* a task defined as a *load interval* on *T:* $[sc_v, ec_v]$
- We build an *interval graph* whose nodes are the load tasks of the EVs and that there is an edge between two nodes iff their load intervals overlap
- The **MIS** of this graph then gives the maximum set of loads achievable on a given station
- A **coloring** of this graph provides the minimum number of stations required for all the loads

Modelling

- We need not only *load durations*, but also *load starting dates*.
- These could be calculated on a price criterion by a very simple Linear Integer Program, as below :
- **Data**
  - A time horizon **[1, T]** discretised in time steps ***t*** of equal duration ***dt*** *(*hours)
  - A price λ^t for each time steps (€/kWh)
  - A load power ***PC*** (kW) assumed to be identical for all stations
  - A charge $\boldsymbol{EC_v}$ (kwh) to be delivered to each *v* on the horizon
  - From ***PC*** and $\boldsymbol{EC_v}$ we deduce a *load duration* expressed in time steps, for each *v* : $\boldsymbol{DC_v} = \frac{EC_v}{PC \times dt}$

Modelling

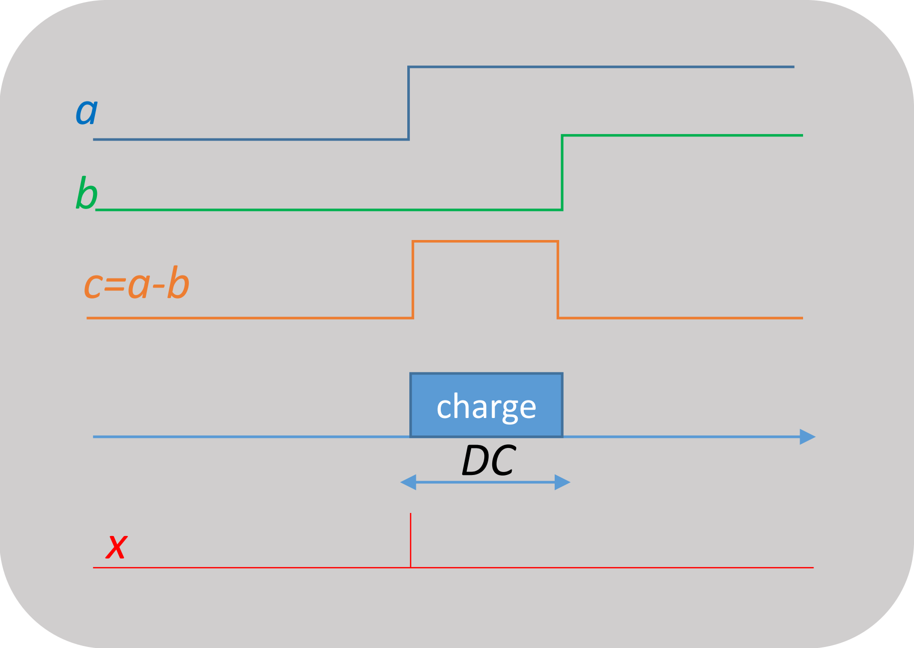
- **Variables**
- x_v^t = 1 if *v* load begins in *t*, 0 if not

- $\forall\,(v, t \in [1,T])\ a_v^t = \sum_{\tau=1}^{t} x_v^\tau$
  = 1 if *t* ≥ starting time of *v* load, 0 if not

- $\forall\,(v, t \in [1,T])\ b_v^t = \begin{cases} \sum_{\tau=1}^{t-DC_v} x_v^\tau \text{ si } t > DC_v \\ 0 \text{ si } t \le DC_v \end{cases}$
  = 1 if *t* > end time of *v* load, 0 if not

$\Longrightarrow \forall\,(v, t \in [1,T])\ c_v^t = a_v^t - b_v^t =$

- $\begin{cases} \boldsymbol{\sum_{\tau=1}^{t} x_v^\tau \text{ si } t \le DC_v} \\ \sum_{\tau=1}^{t} x_v^\tau - \sum_{\tau=1}^{t-DC_v} x_v^\tau = \sum_{\tau=1}^{t-DC_v} x_v^\tau + \sum_{\tau=t-DC_v+1}^{t} x_v^\tau - \sum_{\tau=1}^{t-DC_v} x_v^\tau = \boldsymbol{\sum_{\tau=t-DC_v+1}^{t} x_v^\tau \text{ si } t > DC_v} \end{cases}$
  = 1 if *t* belongs to the load interval of *v*, 0 if not



Modelling

- **Constraints** :
  - For each vehicle *v,* a single charge on the horizon : $\forall\, v, \sum_{t=1}^{T} x_v^t = 1$
  - Charges must be complete on the horizon :
    $\forall\, v, \sum_{t=1}^{T} c_v^t PCdt = EC_v \Leftrightarrow \sum_{t=1}^{T} c_v^t = EC_v / PCdt \Leftrightarrow \sum_{t=1}^{T} c_v^t = DC_v$
- **Objective :** $\boldsymbol{Min_{(x_v^t, c_v^t)} \left\{ \sum_v \sum_{t=1}^{T} \lambda^t c_v^t \, dt \right\}}$
- **Solving this problem provides the *optimal load start dates* according to the cost of each load over the time horizon considered.**
  - Note that we can easily add constraints to this problem, such as:
    - *Load date at the earliest*, for example to take into account a minimum travel time to the stations;
    - *Load date at the latest*, for example to take into account expected future vehicle engagements;
    - *Maximum number of parallel loads at a time* : to limit the number of overlapping intervals with respect to the number of available stations
- **Associated with the *charge durations*, this makes it possible to build a *graph of intervals* on which the specific techniques of *graph coloration/MIS* can apply**

From graph-theory problems to quantum algorithms

Ongoing works in collaboration with :

- **Margarita Veshchzerova's PhD thesis co-advised with E. Jeandel and Simon Perdrix, Loria/Mocqua Université de Nancy**
- **Institut d'Optique/Atos/European Project PASQuanS (Programmable Atomic Large-Scale Quantum Simulation)**
- **The start up Pasqal (spin off from Institut d'Optique)**

Minimization of total charging time

- Max-cut : a “classical” application of QAOA [Fahri et al 14]

- Our current research topic : QAOA from Max-cut to Max-**m**-cut

Minimization of the number of charging stations

- MIS : QA, QAA, QAOA
- Very promising results obtained for **Unit-Disk Graphs** on quantum devices using Rydberg atoms as qubits [Pichler et al 18]
  - Results recently reproduced by the Atos team on the QLM
  - A Unit-Disk Graph is such that two vertices are connected iif the distance between them is < r
- Our current research topic: from graphs of overlapping load intervals to Unit-Disk Graphs of Rydberg atoms arrays

References

- Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106, January 2000.
- Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to instances of an NP-complete problem. Science, 292:5516, 2001
- Edward Farhi, Jeffrey Goldstone, Sam Gutmann. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028v1. 2014
- R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics, 5:287 – 326, 1979
- Hannes Pichler, Sheng-Tao Wang, Leo Zhou, Soonwon Choi, and Mikhail D. Lukin. Quantum Optimization for Maximum Independent Set Using Rydberg Atom Arrays. arXiv:1808.10816v1 [quant-ph] 31 Aug 2018.
- Pinedo. Scheduling_Theory, Algorithms, and Systems Scheduling\Scheduling_Theory, Algorithms, and Systems(Pinedo,2008).pdf
- Martin Skutella. Semidefinite relaxations for parallel machine scheduling. In Proceedings 39th Annual Symposium on Foundations of Computer Science. 1998. Scheduling\Semidefinite Relaxations for Parallel Machine Scheduling.pdf
- Heng Yang , Yinyu Ye , Jiawei Zhang. An approximation algorithm for scheduling two parallel machines with capacity constraints. Discrete Applied Mathematics 130 (2003) 449 – 46. Scheduling\An approximation algorithm for scheduling two parallel machines with capacity constraints.pdf

Thank you !