Università degli Studi di Padova

Wind Farm Cable Problem

Advanced combinatorial optimization algorithms
Ricerca Operativa 2

Benini Michele **1139089**
Piona Davide **1149616**

CONTENTS

1 DEVELOPMENT ENVIRONMENT

To allow replicability of our experiments we add some informations about the system and the environment that we used for the following research. We used the following development environment:

- CPLEX : version 0.1-2018
- Linux Ubuntu : version 18.04 LTS (Operative System)
- C : (language)
- Sublime text: (IDE / text editor)
- Github : (versioning)
- Gnuplot : (library plot)

1.1 REAL WORLD INFO

As in Fischetti and Pisinger, 2018 research, we used a library of real-life instances that are made publicly available for benchmarking. For benchmark purposes we collected the data of five different real wind farms in operation in United Kingdom (wf02, wf03, wf05) and Denmark (wf01, wf04). Different types of cable with different costs, capacities and electrical resistances are available on the market; we considered 5 different sets of real cables, named cb01, cb02, cb03, cb04, and cb05; the turbines in each wind park are of the same type, so we can express the cable capacities as the maximum number of turbines that it can support. Finally we estimated the maximum number of connections to the substation, namely the input parameter C. The ?? table contains all the informations.

name	site	turbine type	no. of turbines	C	allowed cables
wf01	Horn Rev 1	Vestas 80-2MW	80	10	cb01-cb02-cb05
wf02	Kentis Flats	Vestas 90-3MW	30	∞	cb01-cb02-cb04-cb05
wf03	Ormonde	Senvion 5MW	30	4	cb03-cb04
wf04	DanTysk	Simens 3.6MV	80	10	cb01
wf05	Thanet	Vestas 90-3MW	100	10	cb04-cb05

Table 1: Basic information on the real-world wind farms we used for tests.

1.2 INSTANCE TEST BED

The set of tests that we had the possibility to use are the combinations between the 5 instances of wind farms and the 5 sets of real cables. For simplicity and in order to decrease the possibility to make mistakes we adopted the same a numeration as the Fischetti and Pisinger, 2018 article. This solution allows us to easily compare our results with the results in that research. Table ?? reports the resulting numbers of the different instances. For example we renamed the 01 wind farm as data_01.turb and the corresponding cables as data_01.cbl.

number	wind farm	cable set
01	wf01	wf01_cb01_capex
02		wf01_cb01
03		wf01_cb02_capex
04		wf01_cb02
05		wf01_cb05_capex
06		wf01_cb05
07	wf02	wf02_cb01_capex
08		wf02_cb01
09		wf02_cb02_capex
10		wf02_cb02
12		wf02_cb04_capex
13		wf02_cb04
14		wf02_cb05_capex
15		wf02_cb05
16	wf03	wf03_cb03_capex
17		wf03_cb03
18		wf03_cb04_capex
19		wf03_cb04
20	wf04	wf04_cb01_capex
21		wf04_cb01
26	wf05	wf05_cb04_capex
27		wf05_cb04
28		wf05_cb05_capex
29		wf05_cb05

Table 2: Test Bed instance numbers.

2 MATHEMATICAL MODEL

2.1 WIND FARM CABLE PROBLEM INTRODUCTION

We have studied the Wind Farms Cable Problem, which is represented by a number of wind farms in the sea that produces energy; the power production of needs to be routed by some cables to the substation and then directed to the coast. To do that, each turbine must be connected through a cable to another turbine, and eventually to a substation.

This problem complexity if the cable routing problem is strongly NP-hard according to [cite pdf].They have proved that the problem is NP-hard in two formulation. First, in the case where all turbines have the same power production and the nodes are not associated with points in the plane. Second, in the case where the turbines can have different power production and are associated with point in the plane.

When designing a feasible cable routing it's necessarly to take in account a number of constraints. Here we list some of them and then we'll describe the mathematical model that realizes those constraints. Our model is based on the following requirements:

- since the energy flow is unsplittable, the energy flow leaving a turbine must be supported by a single cable;
- power losses should be avoided beacuse it will cause revenue losses in the future; [?]
- different cables, with different capacities and costs, are available; this means that it is important to choose the right cable to minimize the costs without affecting the revenues
- the energy flow on each connection cannot exceed the capacity of the installed cable;
- due to the substation physical layout, a given maximum number of cables, say C, can be connected to each substation;
- cable crossing should be avoided. (we will discuss this problem in the next subsection)

[è corretta questa introduzione di K e n?]

Let K denote the number of different types of cables that can be used and let n be the number turbines.

Definition of $y_{i,j}$:

$$y_{i,j} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is constructed} \\ 0 & \text{otherwise.} \end{cases} \qquad \forall\, i,j = 1,...,n$$

$$y_{i,i} = 0, \quad \forall i, i = 1, ...n$$

Definition of $x^k_{i,j}$:

$$x^k_{i,j} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is constructed with cable type k} \\ 0 & \text{otherwise.} \end{cases} \qquad \forall\, i, j = 1, ..., n \,\forall\, k = 1, ...K$$

Definition of $f_{i,j}$:

$$f_{i,j} \geqslant 0, \quad \forall i, j = 1, ...n$$

Objective function:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k}^{K} \mathtt{cost}(k) \cdot \mathtt{dist}(i,j) \cdot x^k_{i,j} \tag{1}$$

Constraints:

$$\sum_{j=1}^{n} y_{h,j} = \begin{cases} 1 & \text{if } P_h \geqslant 0, \quad \forall h = 1, ..., n \\ 0 & \text{if } P_h = -1 \end{cases} \tag{2}$$

$$\sum_{i=1}^{n} y_{i,h} \leqslant C, \quad \forall h \mid P_h = -1 \tag{3}$$

$$\sum_{j=1}^{n} f_{hj} = \sum_{i=1}^{n} f_{ih} + P_h, \quad \forall h \mid P_h \geqslant 0 \tag{4}$$

$$y_{i,j} = \sum_{k=1}^{K} x^k_{i,j}, \quad i, j = 1, ..., n \tag{5}$$

$$\sum_{k=1}^{K} \mathtt{cap}(k) x^k_{ij} \geqslant f_{ij}, \quad \forall i, j = 1, ..., n \tag{6}$$

[magari cambiare ordine qui sotto]
The objective function 1 minimizes the total cable layout cost, where $\mathtt{dist}(i,j)$ is the Euclidean distance between nodes i and j and $\mathtt{cost}(k)$ is the unit cost for the cable k. Constraints 5 impose that only one type of cable can be selected for each build arc. Constraints 4 are flow conservation constraints: the energy exiting each node h is equal to the energy entering h plus the power production of the node. Constraints 6 ensure that the flow does not exceed the capacity of the installed cable. Constraints 2 impose that only one cable can exit a turbine and that no one cable can exit from the substation. Constraint 3 imposes the maximum number of cables (C) that can enter in a substation, depending on the data of the instance.

2.2 CROSSING CABLES

According to Fischetti and Pisinger, 2018, an important constraint is that cable crossings should be avoided. In principle, cable crossing is not impossible, but is strongly discouraged in practice as building one cable on top of another is more expensive and increases the risk of cable damages.

2.2.1 Crossing check

In order to evaluate if two arches cross: given the arc a between P_1 and P_2 and the arc b between P_3 and P_4. We define the coordinates of a general point as: $P_j = (X_i, y_i)$.
Using the Cramer method we have:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \lambda \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} \quad \lambda \in]0, 1[$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} + \mu \begin{pmatrix} x_4 - x_3 \\ y_4 - y_3 \end{pmatrix} \quad \mu \in]0, 1[$$

Then we evaluate if the determinant is equal to zero value; to do this in a calculator environment avoiding mistakes we will check if the determinant is smaller than a constant epsilon with value $\simeq 10^{-9}$. We can have two situations:

1. if $\det = 0 \quad \Rightarrow$ no crossing
2. if $\det \neq 0 \quad \Rightarrow (\lambda, \mu)$ $\mathtt{if}$ $(\lambda \in]0, 1[)$ && $(\mu \in]0, 1[) \Rightarrow$ crossing

2.2.2 Crossing condition

[non ho ben capito qui cosa rappresentino le funzioni, intanto le ho scritte poi basta aggiungere una descrizione]
1) condizione base

$$y(a, b) + y(c, d) \leqslant 1, \quad \forall (a, b, c, d) : [P_a, P_b]/\mathrm{cross}/[P_c, P_d]$$

2)

$$y(a, b) + y(c, d) + y(b, a) + y(d, c) \leqslant 1, \quad \forall (a, b, c, d) : [P_a, P_b]/\mathrm{cross}/[P_c, P_d]$$

In this case the integer solution is equal to the previous one, but in case of fractional solution the performances are much better. However the number of checks is $O(n^4)$ which is still too much.

3)
Given:

$$Q(a, b, c) = \{(a, b), (b, a)\} \vee \{(c, d) \in \delta^+(c) : [P_c, P_d]/\mathrm{cross}/[P_a, P_b]$$

$$\sum y_{ij} \leqslant 1, \quad (i, j) \in \ Q(a, b, c)$$

This rule is more stronger and creates less constraints. It exploits the fact that from a node C can exit only one cable, no more.

2.2.3 Lazy Constraints

The generated constraints are often too much and risks to block the solution for a long time. Instead of add sistematically all the constraints in the model at the beginning, we generates them "on the fly" when they are violated by the Branch and Bound process, and we add the new constraints before to update the incumbent. The CPLEX command to add a set of constraints is CPXAddLazyConstraints.
This tecnique decreases the efficiency of the CPLEX pre-processing, but generally gives good results.

!!!!!!!!!!!FIN QUI!!!!!!!!!!

2.3 ...

3 METHODS

3.1 PLAIN EXECUTION

We simply create the linear programming model and then we pass it to CPLEX for the optimization. The performances, as we will notice for all the methods, depends on the instance: we noticed the power of CPLEX that find the optimal solution in few seconds, and in the meantime we discovered that some instances takes many hours to be solved by our machines. The main steps of our code in this phase are: to read the input files and parse it ...

3.2 RELAXED MODE

RELAX: this method tries to 'relax' some constraints in order to make faster the process of searching the first solution (so that RINS can start working). We add a slack variable >=0 in the model. Then we add this variable also in the objective function multiplied for a constant reasonably large. In this way, even if CPLEX could find a wrong initial solution, it's probable that this solution will rapidly get better and became correct.

3.3 CPLEX HEURISTICS-PARAMS

We enable some CPLEX heuristic methods adapting them to our specific case and instances. We use: RINS: tries to improve the incumbent; in the initial part the process don't change, but as soon as a solution is found the RINS heuristic tryes to improve it with more frequency. It is possible to infer that the RINS method has been used in a CPLEX step when in the logs there is a '*' near to the number (?) POLISHING: this heuristic tries to modify some variables of a (good) solution to improve the solution; it is possible to set a condition that enables this method, in order to avoid a too erply usage of this method that can lead to a waste of time an performances.

3.4 NO-CROSSING CONDITION

In order to avoid that the cables crosses each other it's necessary to add a function that checks that no cable crosses another. This check could be done using the Cramer Method:

. . . formule e immagine... So the most intuitive constraints is the following: . . . But, using the constraints that from a vertex the outgoing edges must be <=1 , we could use this constraint: . . . formula e immagine . . .

3.5 LAZY CONSTRAINTS METHOD

Adding all the "no-crossing constraints" statically to the model will probably block it for a really long time. So we add them to the CPLEX pool of constraints and it will check those constraints only when a solution is created. In the case that some constraint is violated CPLEX will add the corresponding constraint before the incumbent update. (?) In the end we use CPXAddLazyConstraints instead of CPXAddRows. The laxy constraints decrease the power of the CPLEX pre-processing.

We used a condition (?) that helps the process avoiding some duplicated constraints .. (?) We noticed that, even if we add the constraints using CPXAddLazyConstraints, the computation time of the solution is sometimes really high.

3.6 LOOP METHOD

3.7 CALLBACK METHOD

3.8 ...

4 HEURISTICS

4.1 MATH HEURISTIC – HARD FIXING

4.2 TABOO SEARCH

4.3 ANT ALGORITHM

4.4 ...

Ciao

BIBLIOGRAPHY

Fischetti, Martina and David Pisinger

2018 "Optimizing wind farm cable routing considering power losses", *European Journal of Operational Research*, 270, 3, pp. 917-930, ISSN: 0377-2217, DOI: 10.1016/j.ejor.2017.07.061.