Proximity-driven MIP heuristics
with an application to
wind farm layout optimization

Matteo Fischetti, University of Padova, Italy

Joint work with Martina Fischetti and Michele Monaci
MIP technology

- **Mixed-Integer (Linear) Programming** is a powerful technique that recently became a feasible and appealing tool to solve complex/huge real problems.

<table>
<thead>
<tr>
<th>Assuming (a modest) 1000x machine speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>It took</strong></td>
</tr>
<tr>
<td>&gt;4 months (early 90's)</td>
</tr>
<tr>
<td>&gt;7 years (early 90's)</td>
</tr>
</tbody>
</table>
Advantages of the MIP approach

• Many industrial problems can be modeled as MIPs

• Different constraints can easily be added → what if analysis

• In many cases, off-the-shelf MIP software is able to produce a proven-optimal solution

• In the hardest cases, MIP-based heuristics yield very good solutions within acceptable computing times

• MIP-based heuristics can be easier to design and implement than ad-hoc heuristics
A case study: wind farm layout

Given

• a site (*offshore* or onshore)
• characteristics of the turbines to build
• measurements of the wind in the site

Determine a turbine allocation that **maximizes power production**

Taking into account:

• proximity constraints (no collisions)
• minimum/maximum number of turbines
• wake effects
The problem

• Define a grid of **sites** (candidate points for turbine allocation)

• For each site pair \((i,j)\), let \(I_{ij}\) denote the average interference (power loss) experienced at point \(j\) if a turbine is built on site \(i\). It depends on average wind speed and direction, nonlinear turbine power curve, etc.

• Assume overall interference is **cumulative** (sum of pairwise interf.s)
Basic (quadratic) model

Let $V$ be the site set, $P_i$ be the max. power production at point $i$, $E_i$ denote incompatible site pairs, and $N_{MIN}$ and $N_{MAX}$ be input limits on the n. of built turbines

$$\begin{align*}
\max & \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V} I_{ij} x_i x_j \\
\text{s.t.} & \quad N_{MIN} \leq \sum_{i \in V} x_i \leq N_{MAX} \\
& \quad x_i + x_j \leq 1 \quad \forall [i, j] \in E_I \\
& \quad x_i \in \{0, 1\} \quad \forall i \in V
\end{align*}$$
Designing a simple wind-farm heuristic

- Our first heuristic is **not** based on the MIP model → hopefully easy to implement…

- **Basic move:** Given a feasible solution $x$, we want to see if we can improve it by flipping a single variable $x_j \rightarrow 1$-opt exchange

- **Simple heuristic:** for each $j$, compute the objective improvement $\delta_j$ when flipping $x_j$ (alone), and find $\max \{ \delta_i \}$

  $\sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V} I_{ij} x_i x_j$

  where $I_{ij} = +\infty$ for incompatible pairs $[i,j] \in E$

- **Complexity:** $O(|V|^2)$ for each max computation
Improving the basic heuristic

- Complexity can be reduced from $O(|V|^2)$ to $O(|V|)$ by using parametric techniques:

1. Initialize in $O(|V|^2)$

$$\delta_j = \begin{cases} 
    P_j - \sum_{i \in V: x_i = 1} (I_{ij} + I_{ji}) & \text{if } x_j = 0 \\
    -P_j + \sum_{i \in V: x_i = 1} (I_{ij} + I_{ji}) & \text{if } x_j = 1 
\end{cases}$$

2. When a certain $x_{j^*}$ is going to be flipped, incrementally update all $\delta_j$'s in $O(|V|)$ time

$$\delta_j = \begin{cases} 
    -\delta_j & \text{if } j = j^* \\
    \delta_j - (I_{jj^*} + I_{j^*j}) & \text{if } j \neq j^* \text{ and } x_j = x_{j^*} \\
    \delta_j + (I_{jj^*} + I_{j^*j}) & \text{if } j \neq j^* \text{ and } x_j \neq x_{j^*} 
\end{cases}$$
... and improving

• **2-opt exchanges** can be implemented as well → time consuming, but we can apply it only from time to time, etc.

• Start with a **better initial** solution?

  1. Start with the null solution $x = 0$  
     
     #toobad

  2. Greedy heuristic  
     
     #better

  3. Randomized greedy  
     
     #grasp

  4. Smart solutions tend to put more turbines on the border of offshore area  
     
     #smart

  5. … more and more ideas pop out and require to be implemented, debugged and tested  
     
     #curseofbeingtoosmart
... and improving ...

- Escaping **local optimal solutions** by using
  1. Random restarts
  2. Tabu Search
  3. Variable Neighborhood Search (VNS)
  4. Simulated Annealing
  5. Genetic Algorithms
  6. Evolutionary Heuristics
  7. 

- **... our first heuristic is not based on the MIP model** → hopefully *easy to implement*... →*are we sure?*
MIP heuristics

• Consider a generic Mixed-Integer convex 0-1 Problem (0-1 MIP)

\[
\begin{align*}
\min f(x) \\
g(x) &\leq 0 \\
x_j &\in \{0, 1\} \quad \forall j \in J
\end{align*}
\]

where \( f \) and \( g \) are convex functions and removing integrality leads to an easy-solvable continuous relaxation

• A **black-box** (exact or heuristic) MIP solver is available

• **How to use the MIP solver to quickly provide a sequence of improved heuristic solutions (time vs quality tradeoff)?**
Large Neighborhood Search

• **Large Neighborhood Search** (LNS) paradigm:
  
  1. introduce **invalid constraints** into the MIP model to create a nontrivial sub-MIP “centered” at a given heuristic sol. \( \tilde{x} \) (say)
  2. Apply the MIP solver to the sub-MIP for a while…

• Possible implementations:
  
  – **Local branching**: add the following linear cut to the MIP

\[
\Delta(x, \tilde{x}) = \sum_{j \in J: \tilde{x}_j = 0} x_j + \sum_{j \in J: \tilde{x}_j = 1} (1 - x_j) \leq k
\]

  – **RINS**: find an optimal solution \( x^* \) of the continuous relaxation, and fix all binary variables such that \( x^*_j = \tilde{x}_j \)

  – **Polish**: evolve a population of heuristic sol.s by using RINS to create offsprings, plus mutation etc.
Proximity search

• We want to work with a modified objective function that hopefully allows the black-box solver to quickly improve the incumbent solution $\tilde{x}$

• “Stay close” principle: we bet on the fact that improved solutions live near the incumbent, hence we attract the search within a neighborhood of $\tilde{x}$ (without imposing any artificial neighborhood constraints)

• **Step 1.** Add an explicit cutoff constraint $f(x) \leq f(\tilde{x}) - \theta$

• **Step 2.** Replace the objective $f(x)$ by the proximity function

$$\Delta(x, \tilde{x}) = \sum_{j \in J: \hat{x}_j = 0} x_j + \sum_{j \in J: \hat{x}_j = 1} (1 - x_j) = \|x - \tilde{x}\|_J^2$$
Proximity search heuristic

1. run a black-box solver on the original 0-1 MIP, until a “reasonably good” feasible solution $\tilde{x}$ is found;
   \begin{verbatim}
   repeat
   2. explicitly add the cutoff constraint $f(x) \leq f(\tilde{x}) - \theta$ to the MIP model, where $\theta > 0$ is a given parameter;
   3. replace the objective function $f(x)$ by a new “proximity” one, say $\Delta(x, \tilde{x})$;
   4. run the MIP solver on the new model until a termination condition is reached, and let $x^*$ be the best feasible solution found;
   5. if $J \subset N$ then refine $x^*$ by solving the convex program $x^* := \text{argmin}\{f(x) : g(x) \leq 0, x_j = x_j^* \forall j \in J\}$;
   6. recenter $\Delta(x, \cdot)$ by setting $\tilde{x} := x^*$, and/or update $\theta$
   \end{verbatim}

until an overall termination condition is reached;

return $\tilde{x}$
A MIP-based heuristic for wind farm

- What you need here is

1. A robust MIP solver

2. An idea of the **size** and difficulty of that practical instances that we want to solve (100 sites? or 1,000? or 10,000?)

3. A sound MIP model that **“does not die”** for the instances of interest → for heuristics, speed is sometimes more important than polyhedral tightness…

4. An idea about **“how to drive the MIP solver”** to deliver the solution you want → LNS, local branching, polish, proximity search…
A standard MIP linearization

- Introduce a quadratic n. of var.s $z_{ij} = x_i \cdot x_j$

$$\begin{align*}
\text{max} & \quad \sum_{i \in V} P_i x_i - \sum_{i \in V} \sum_{j \in V, i < j} (I_{ij} + I_{ji}) z_{ij} \\
\text{s.t.} & \quad N_{MIN} \leq \sum_{i \in V} x_i \leq N_{MAX} \\
& \quad x_i + x_j \leq 1 \quad \forall [i, j] \in E_I \\
& \quad x_i + x_j - 1 \leq z_{ij} \quad \forall i, j \in V, i < j \\
& \quad x_i \in \{0, 1\} \quad \forall i \in V \\
& \quad z_{ij} \in \{0, 1\} \quad \forall i, j \in V, i < j
\end{align*}$$
An alternative MIP linearization

Glover’s trick: the objective function

\[ \sum_{i \in V} P_i x_i - \sum_{i \in V} (\sum_{j \in V} I_{ij} x_j) x_i \]  

is restated as

\[ \sum_{i \in V} (P_i x_i - w_i) \]  

where

\[ w_i := (\sum_{j \in V} I_{ij} x_j) x_i = \begin{cases} \sum_{j \in V} I_{ij} x_j & \text{if } x_i = 1; \\ 0 & \text{if } x_i = 0. \end{cases} \]

→ the new continuous variable \( w_i \) is the product between a continuous term (\( \sum \ldots \)) and a binary variable \( x_i \) → McCormick linearization
An alternative MIP linearization

A linearized model with linear n. of additional var.s $w_i$ and BIGM constr.s

\[
\begin{align*}
\text{max } z &= \sum_{i \in V} (P_i x_i - w_i) \\
\text{s.t. } N_{\text{MIN}} &\leq \sum_{i \in V} x_i \leq N_{\text{MAX}} \\
&\quad x_i + x_j \leq 1 \quad \forall [i, j] \in E_I \\
&\quad \sum_{j \in V} I_{ij} x_j \leq w_i + M_i (1 - x_i) \quad \forall i \in V \\
&\quad x_i \in \{0, 1\} \quad \forall i \in V \\
&\quad w_i \geq 0 \quad \forall i \in V
\end{align*}
\]
Which linearization is better?

<table>
<thead>
<tr>
<th></th>
<th>n = 100</th>
<th>n = 150</th>
<th>n = 200</th>
<th>n = 300</th>
<th>n = 500</th>
<th>n = 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mod 2</td>
<td>Mod 4</td>
<td>Mod 2</td>
<td>Mod 4</td>
<td>Mod 2</td>
<td>Mod 4</td>
</tr>
<tr>
<td>root LP time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.07</td>
</tr>
<tr>
<td>root time</td>
<td>78.6</td>
<td>22.79</td>
<td>66.05</td>
<td>66.96</td>
<td>87.55</td>
<td>88.69</td>
</tr>
<tr>
<td>root bound</td>
<td>54.38</td>
<td>53.1</td>
<td>53.46</td>
<td>53.96</td>
<td>63.50</td>
<td>63.84</td>
</tr>
<tr>
<td>root heu.sol.</td>
<td>9600</td>
<td>70.12</td>
<td>66.02</td>
<td>66.96</td>
<td>87.55</td>
<td>88.69</td>
</tr>
<tr>
<td>root cuts</td>
<td>54.38</td>
<td>53.1</td>
<td>53.46</td>
<td>53.96</td>
<td>63.50</td>
<td>63.84</td>
</tr>
<tr>
<td>final n. nodes</td>
<td>16K</td>
<td>16M</td>
<td>178K</td>
<td>27M</td>
<td>33K</td>
<td>12M</td>
</tr>
<tr>
<td>final best bound</td>
<td>54.49</td>
<td>54.49</td>
<td>54.15</td>
<td>54.36</td>
<td>65.09</td>
<td>66.28</td>
</tr>
<tr>
<td>final best heu.sol</td>
<td>54.49</td>
<td>54.49</td>
<td>54.15</td>
<td>54.36</td>
<td>65.09</td>
<td>66.28</td>
</tr>
<tr>
<td>final time</td>
<td>2006.05</td>
<td>1698.70</td>
<td>3600</td>
<td>3600</td>
<td>3600</td>
<td>3600</td>
</tr>
</tbody>
</table>

Comparison between the linearizations with quadratic n. of var.s/constr.s (Mod 2) and with linear n. of var.s/constr.s (Mod 4) time limit of 3600 sec.s on a PC

→ Mod 4 (linear n. of var.s/constr.s) much better for heuristics
Our overall MIP-based heuristic

- **Step 0.** read input data and compute the overall interference matrix \((I_{ij})\);

- **Step 1.** (optional) apply **ad-hoc heuristics** (1-opt) to get a first incumbent \(\tilde{x}\);

- **Step 2.** (optional) apply quick **ad-hoc refinement heuristics** (few iterations of iterated 1- and 2-opt) to possibly improve \(\tilde{x}\);

- **Step 3.** if \(n > 2000\), randomly **remove points** \(i\) with \(\tilde{x}_{i} = 0\) so as to reduce the number of candidate sites to 2000;

- **Step 4.** build a MIP model for the resulting subproblem and apply **proximity search** to refine \(\tilde{x}\) until the very first improved solution is found (or time limit is reached);

- **Step 5.** if time limit permits, repeat from Step 2.
Computational results

Alternative heuristics implemented in C and run on a quad-core PC (16GB RAM)

a) proxy: our MIP-based proximity-search heuristic built on top of Cplex 12.5.1

b) cpx_def: Cplex 12.5.1 in its default setting, starting from the same heuristic solution $\tilde{x}$ used by proxy

c) cpx_heu: same as cpx_def, with an internal tuning intended to improve heuristic performance (aggressive RINS & Polish)

d) loc_sea: an ad-hoc local-search procedure not based on any MIP solver

Testbed (real offshore site: Horns Rev 1 in Denmark)

- offshore 3,000 x 3,000 (m) square with 400m minimum turbine separation
- no limit on the number of turbines to build
- Siemens SWT-2.3-93 turbines (rotor diameter 93m)
- pairwise interference computed using Jensen's model, by averaging 250,000+ real-world wind samples from Horns Rev 1 (Denmark)
Computational results

Fig. 4 Solution profit over time for 4 sample instances with $n = 1,000$ (top left and top right), $n = 5,000$ (bottom left), and $n = 10,000$ (bottom right); the higher the profit the better.
Thanks for your attention

Papers


and slides available at www.dei.unipd.it/~fisch