On warm-starting constraint generation methods via machine learning tools for solving mixed-integer programs

Asunción Jiménez-Cordero

asuncionjc@uma.es

Joint work with:
Juan Miguel Morales González
Salvador Pineda Morente

oasys.uma.es

XXXIX Congreso Nacional de Estadística e Investigación Operativa (SEIO 2022)

June 8th, 2022

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 755705)

Outline

Outline

Mixed Integer
Linear Programs
(MILP)

Machine
Learning
(ML)

Combine knowledge from both worlds

Recent reviews: *Bengio et al. [2021]; Gambella et al. [2021]*

Outline

MILP

$$(P_{\boldsymbol{\theta}}[\mathcal{J}]) \qquad \begin{cases} \min\limits_{\boldsymbol{z} \in \mathbb{R}^n \times \mathbb{Z}^q} \boldsymbol{c}^T \boldsymbol{z} \\ \quad \text{s.t. } \boldsymbol{a}_j^T \boldsymbol{z} \leq b_j, \quad \forall j \in \mathcal{J} \end{cases}$$

- $\boldsymbol{\theta} = \{\boldsymbol{c}, \boldsymbol{a}_j, b_j, \forall j \in \mathcal{J}\}$.
- $P_{\boldsymbol{\theta}}[\mathcal{J}]$ bounded and feasible.
- Optimal solution $\boldsymbol{z}^*_{\boldsymbol{\theta}}[\mathcal{J}]$ is a singleton.

MILP

$$(P_{\boldsymbol{\theta}}[\mathcal{J}]) \qquad \begin{cases} \min\limits_{\boldsymbol{z}\in\mathbb{R}^n\times\mathbb{Z}^q} \boldsymbol{c}^T\boldsymbol{z} \\ \quad \text{s.t. } \boldsymbol{a}_j^T\boldsymbol{z} \leq b_j, \quad \forall j \in \mathcal{J} \end{cases}$$

- NP-hard. Very difficult to solve.
- Pure optimization-based strategy: Constraint generation (CG), *Minoux [1989]*.
- Iterative process. Unaffordable in online applications.

MILP

$$(P_{\boldsymbol{\theta}}[\mathcal{J}]) \qquad \begin{cases} \min\limits_{\boldsymbol{z} \in \mathbb{R}^n \times \mathbb{Z}^q} \boldsymbol{c}^T \boldsymbol{z} \\ \quad \text{s.t. } \boldsymbol{a}_j^T \boldsymbol{z} \leq b_j, \quad \forall j \in \mathcal{J} \end{cases}$$

- Apply ML to alleviate computational burden in CG.
- Enhacing performance in MILPs.
- Learning from the information of previously solved instances.
- Providing a *good* warm-start.
- Reducing number of iterations in CG.
- Reducing computational time in MILPs.

MILP

$$(P_{\boldsymbol{\theta}}[\mathcal{J}]) \qquad \begin{cases} \min\limits_{\boldsymbol{z} \in \mathbb{R}^n \times \mathbb{Z}^q} \boldsymbol{c}^T \boldsymbol{z} \\ \quad \text{s.t.} \ \boldsymbol{a}_j^T \boldsymbol{z} \leq b_j, \quad \forall j \in \mathcal{J} \end{cases}$$

- Literature: only use information from the binding constraints, i.e., $\boldsymbol{a}_j^T \boldsymbol{z}^* = b_j$.
- Not enough when integer variables appear.
- Some non-binding constraints should be included. Need to define *invariant constraint set*.

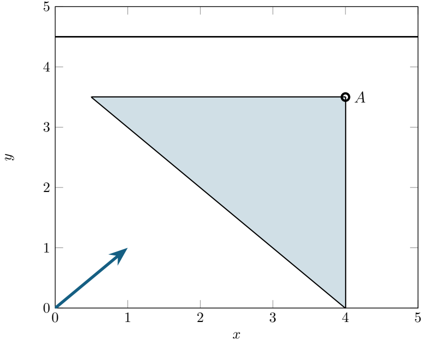
Invariant Constraint Set, $\mathcal{S}$

According to *Calafiore [2010]*:

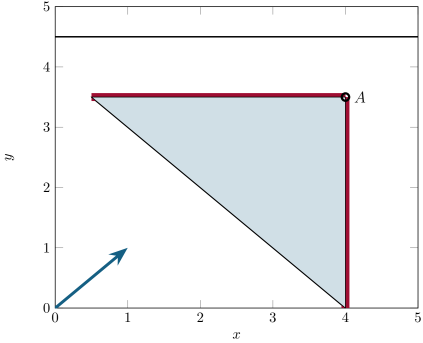
$\mathcal{S} \subset \mathcal{J}$ s.t. $\boldsymbol{c}^T \boldsymbol{z}^*_{\boldsymbol{\theta}}[\mathcal{S}] = \boldsymbol{c}^T \boldsymbol{z}^*_{\boldsymbol{\theta}}[\mathcal{J}]$

The integrality of the decision variables is crucial to find out which constraints belong to $\mathcal{S}$.

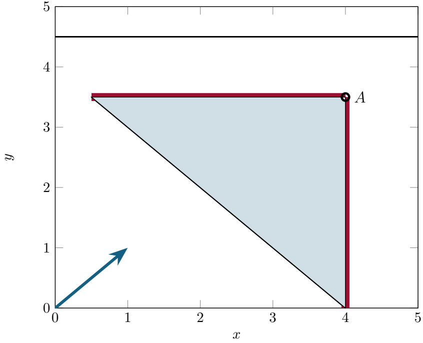
LP vs MILP (Example taken from *Pineda et al. [2020]*)



LP vs MILP (Example taken from *Pineda et al. [2020]*)



LP vs MILP (Example taken from *Pineda et al. [2020]*)

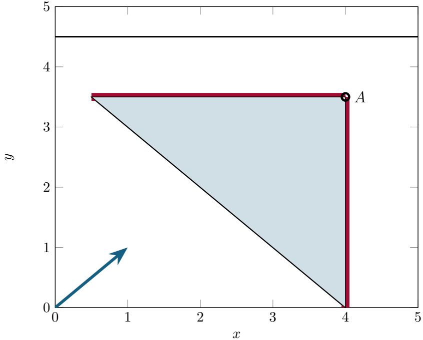


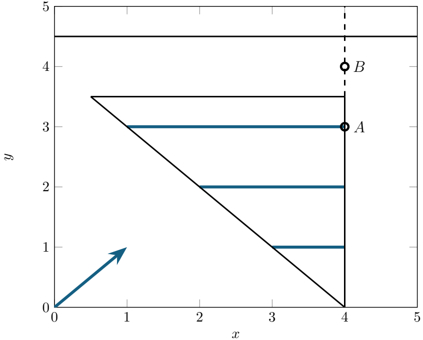
Binding constraints

$$\mathcal{B} = \{ j \in \mathcal{J} : \boldsymbol{a}_j^T \boldsymbol{z}_\theta^*[\mathcal{J}] = b_j \}$$

$$\mathcal{S} = \mathcal{B}$$

LP vs MILP (Example taken from *Pineda et al. [2020]*)



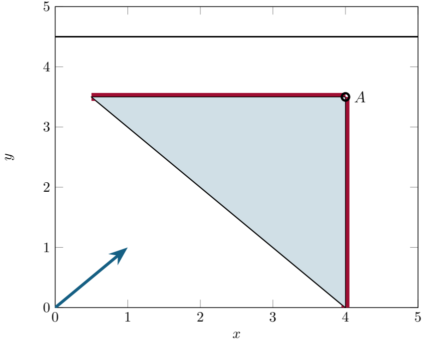


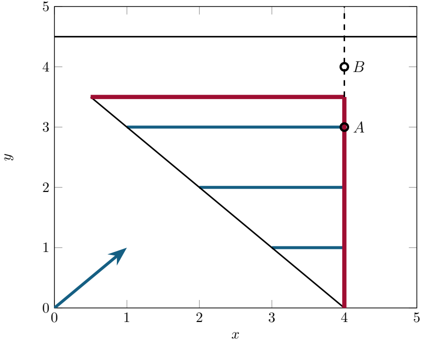
Binding constraints

$$\mathcal{B} = \{j \in \mathcal{J} : \boldsymbol{a}_j^T \boldsymbol{z}_\theta^*[\mathcal{J}] = b_j\}$$

$$\mathcal{S} = \mathcal{B}$$

LP vs MILP (Example taken from *Pineda et al. [2020]*)



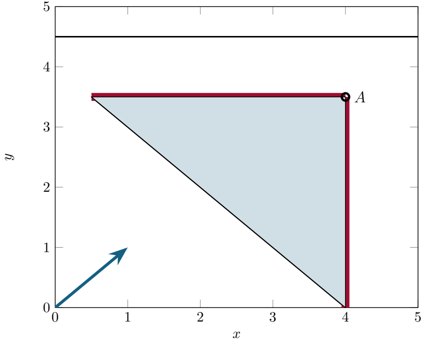


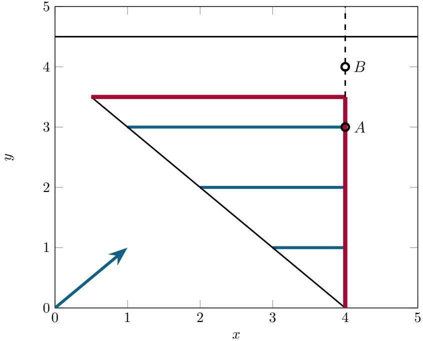
Binding constraints

$$\mathcal{B} = \{j \in \mathcal{J} : \boldsymbol{a}_j^T \boldsymbol{z}_\theta^*[\mathcal{J}] = b_j\}$$

$$\mathcal{S} = \mathcal{B}$$

LP vs MILP (Example taken from *Pineda et al. [2020]*)





Binding constraints

$$\mathcal{B} = \{j \in \mathcal{J} : \boldsymbol{a}_j^T \boldsymbol{z}_\theta^*[\mathcal{J}] = b_j\}$$

Some non-binding constraints

also belong to $\mathcal{S}$.

$$\mathcal{S} = \mathcal{B}$$

$$\mathcal{S} \supset \mathcal{B}$$

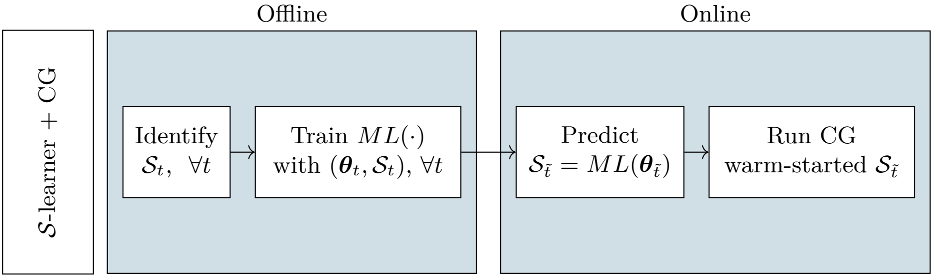
- Finding $\mathcal{S}$ is challenging in MILPs.
- For each train instance t, we look for $\mathcal{S}_t$, including some of the non-binding constraints.
- And reduced MILP $P_{\theta_t}[\mathcal{S}_t]$ is solved.

How to find $\mathcal{S}_t$?

Algorithm Identifying an invariant constraint set for each instance t

0) Initialize $\mathcal{S}_t = \mathcal{B}_t$.
1) Solve $P_{\boldsymbol{\theta_t}}[\mathcal{S}_t]$ with solution $\boldsymbol{z}^*_{\boldsymbol{\theta}_t}[\mathcal{S}_t]$.
2) If $\boldsymbol{z}^*_{\boldsymbol{\theta}_t}[\mathcal{S}_t]$ is infeasible for $P_{\boldsymbol{\theta_t}}[\mathcal{J}]$, go to step 3). Otherwise, stop.
3) $\mathcal{S}_t := \mathcal{S}_t \cup \{j \in \mathcal{J} \setminus \mathcal{S}_t : j \text{ is the most violated constraint}\}$, go to step 1).

Recap


$\mathcal{S}$-learner + CG
Offline
Identify $\mathcal{S}_t, \ \forall t$
Train $ML(\cdot)$ with $(\boldsymbol{\theta}_t, \mathcal{S}_t), \forall t$
Online
Predict $\mathcal{S}_{\tilde{t}} = ML(\boldsymbol{\theta}_{\tilde{t}})$
Run CG warm-started $\mathcal{S}_{\tilde{t}}$

Advantages

- Based on CG. Crucial constraints are included.
- Optimality and feasibility guarantees of CG are preserved.
- Reducing number of iterations of CG.
- Maybe only one iteration.
- Independent on the ML method used.
- Identifying $\mathcal{S}$ and training ML is performed offline.

Outline

Experimental Setup

- Binary classification problem. $k$nn.
- Label $s_j^t = \pm 1$ depending on inclusion on $\mathcal{S}_t$.
- We choose to be conservative when including constraints.
- Synthetic and real-world applications.

Two comparative approaches

	Offline	Online
CG		Run CG no warm-start
$\mathcal{B}$-learner + CG	Identify $\mathcal{B}_t, \ \forall t$ → Train $ML(\cdot)$ with $(\boldsymbol{\theta}_t, \mathcal{B}_t), \forall t$ →	Predict $\mathcal{B}_{\tilde{t}} = ML(\boldsymbol{\theta}_{\tilde{t}})$ → Run CG warm-started $\mathcal{B}_{\tilde{t}}$
$\mathcal{S}$-learner + CG	Identify $\mathcal{S}_t, \ \forall t$ → Train $ML(\cdot)$ with $(\boldsymbol{\theta}_t, \mathcal{S}_t), \forall t$ →	Predict $\mathcal{S}_{\tilde{t}} = ML(\boldsymbol{\theta}_{\tilde{t}})$ → Run CG warm-started $\mathcal{S}_{\tilde{t}}$

Unit Commitment problem

$$\begin{cases} \min\limits_{\boldsymbol{x}\in\mathbb{R}^n,\, \boldsymbol{y}\in\{0,1\}^n} \sum_{i=1}^{n} c_i x_i \\ \quad \text{s.t.} \ \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} d_i, \\ \quad -f_j \leq \sum_{i=1}^{n} a_{ij}(x_i - d_i) \leq f_j, \quad j = 1, \ldots, m \\ \quad l_i y_i \leq x_i \leq u_i y_i, \quad i = 1, \ldots, n \end{cases}$$

- $\boldsymbol{\theta} = \mathbf{d}$.
- $n = 96$.
- $m = 120$ (240 constraints).
- $T = 8640$ (Leave-one-out).

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

Pure optimization-based tools worse than MILP solver

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

Larger values of k imply more constraints (more conservative).

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

Computational gains in both approaches.

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

Few extra constraints in S-learner. Large time improvements.

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

Adding constraints is not enough ($k = 20$ vs $k = 5$)

Results

	k	$[C_{min}, C_{max}]$	$[I_{min}, I_{max}]$	$P_1(\%)$	$\Delta(\%)$
CG	-	[0, 22]	[1, 23]	9.16	188.38
$\mathcal{B}$+CG	5	[0, 23]	[1, 8]	54.40	74.09
	20	[0, 26]	[1, 5]	68.28	62.35
	100	[0, 29]	[1, 5]	83.70	54.62
$\mathcal{S}$+CG	5	[0, 26]	[1, 5]	92.66	44.84
	20	[0, 29]	[1, 4]	98.81	42.83
	100	[0, 32]	[1, 3]	99.71	44.41

• C : constr. • I: iterat. • P_1: one iteration. • Δ: time wrt MILP solver.

$\approx$ 100% of the instances are optimally solved with one iteration (P_1).

More details

Machine-learning-aided warm-start of constraint generation methods for online mixed-integer optimization

Asunción Jiménez-Cordero*, Juan Miguel Morales, Salvador Pineda

OASYS Group. Ada Byron Research Building, Arquitecto Francisco Peñalosa St., 18, 29010, University of Málaga, Málaga, Spain

Abstract

Mixed Integer Linear Programs (MILP) are well known to be NP-hard problems in general. Even though pure optimization-based methods, such as constraint generation, are guaranteed to provide an optimal solution if enough time is given, their use in online applications is still a great challenge due to their usual excessive time requirements. To alleviate their computational burden, some machine learning techniques have been proposed in the literature, using the information provided by previously solved MILP instances. Unfortunately, these techniques report a non-negligible percentage of infeasible or suboptimal instances.

By linking mathematical optimization and machine learning, this paper proposes a novel approach that speeds up the traditional constraint generation method, preserving feasibility and optimality guarantees. In particular, we first identify offline the so-called invariant constraint set of past MILP instances. We then train (also offline) a machine learning method to learn an invariant constraint set as a function of the problem parameters of each instance. Next, we predict online an invariant constraint set of the new unseen MILP application and use it to initialize the constraint generation method. This warm-started strategy significantly reduces the number of iterations to reach optimality, and therefore, the computational burden to solve online each MILP problem is significantly reduced. Very importantly, the proposed methodology inherits the feasibility and optimality guarantees of the traditional constraint generation method. The computational performance of the proposed approach is quantified through synthetic and real-life MILP applications.

Keywords: Mixed integer linear programming, machine learning, constraint generation, warm-start, feasibility and optimality guarantees

Available at:

https://www.researchgate.net/publication/350371853_Machine-learning-aided_warm-start_of_constraint_generation_methods_for_online_mixed-integer_optimization

SCAN ME

Outline

Conclusions

- Approach which combines MILPs and ML.
- Warm-start in CG algorithm.
- Keeping optimality and feasibility guarantees of CG.
- Reduce computational burden.
- Tested on synthetic and real-world applications.

Conclusions

- Approach which combines MILPs and ML.
- Warm-start in CG algorithm.
- Keeping optimality and feasibility guarantees of CG.
- Reduce computational burden.
- Tested on synthetic and real-world applications.

Further research

- Other input parameters.
- Introduce expert-knowledge information.

References

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, doi:10.1016/j.ejor.2020.07.063.

Calafiore, G. C. (2010). Random convex programs. *SIAM Journal on Optimization*, 20(6):3427–3464, doi:10.1137/090773490.

Gambella, C., Ghaddar, B., and Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, doi:10.1016/j.ejor.2020.08.045.

Karapetyan, D., Punnen, A. P., and Parkes, A. J. (2017). Markov chain methods for the bipartite boolean quadratic programming problem. *European Journal of Operational Research*, 260(2):494–506, doi:10.1016/j.ejor.2017.01.001.

Kool, W., Van Hoof, H., and Welling, M. (2019). Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Kruber, M., Lübbecke, M. E., and Parmentier, A. (2017). Learning when to use a decomposition. In Salvagnin, D. and Lombardi, M., eds., *Integration of AI and OR Techniques in Constraint Programming*, pages 202–210. Springer International Publishing.

Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., and Lodi, A. (2018). Predicting tactical solutions to operational planning problems under imperfect information. *arXiv preprint arXiv:1807.11876*.

Liberto, G. D., Kadioglu, S., Leo, K., and Malitsky, Y. (2016). DASH: Dynamic approach for switching heuristics. *European Journal of Operational Research*, 248(3):943–953, doi:10.1016/j.ejor.2015.08.018.

Lodi, A. and Zarpellon, G. (2017). On learning and branching: A survey. *TOP*, 25(2):207–236, doi:10.1007/s11750-017-0451-6.

Minoux, M. (1989). Networks synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, 19(3):313–360, doi:10.1002/net.3230190305.

Pineda, S., Morales, J. M., and Jiménez-Cordero, A. (2020). Data-driven screening of network constraints for unit commitment. *IEEE Transactions on Power Systems*, 35(5):3695–3705, doi:10.1109/TPWRS.2020.2980212.

On warm-starting constraint generation methods via machine learning tools for solving mixed-integer programs

SCAN ME

Asunción Jiménez-Cordero

asuncionjc@uma.es

JOINT WORK WITH:

Juan Miguel Morales González

Salvador Pineda Morente

Thank you very much for your attention!

UNIVERSIDAD DE MÁLAGA

OASYS Optimization and Analytics for Sustainable energY Systems

erc European Research Council

oasys.uma.es

XXXIX Congreso Nacional de Estadística e Investigación Operativa (SEIO 2022)

June 8th, 2021

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 755705)