

COMPÉTITION KAGGLE

DEVELOPPEMENT D'UN ALGORITHME GÉNÉTIQUE

PROJET IML 8

SOMMAIRE

- ▶ Contexte
- ▶ La problématique et son Interprétation
- ▶ Exploration du dataset
- ▶ Modèle Algo Génétique
- ▶ Modèle Stochastic Product Search
- ▶ Modèle MIP Gurobi
- ▶ Conclusions
- ▶ Axes d'améliorations

LE CONTEXTE

- ▶ Compétition Kaggle Santa Tour 2019
 - ▶ Optimisation d'un agenda de visites par des familles
- ▶ 5000 familles de taille différente
- ▶ 10 choix par famille parmi 100 jours avant Noël
- ▶ 2 types de Pénalités à payer par le Père Noël :
 - ▶ ***preference cost*** : $f(\text{choix par ordre de préférence})$
 - ▶ ***accounting cost*** : $f(\text{différence d'affluence entre 2 jours consécutifs})$
- ▶ Contraintes d'affluence : $125 \leq \text{Affluence} \leq 300$

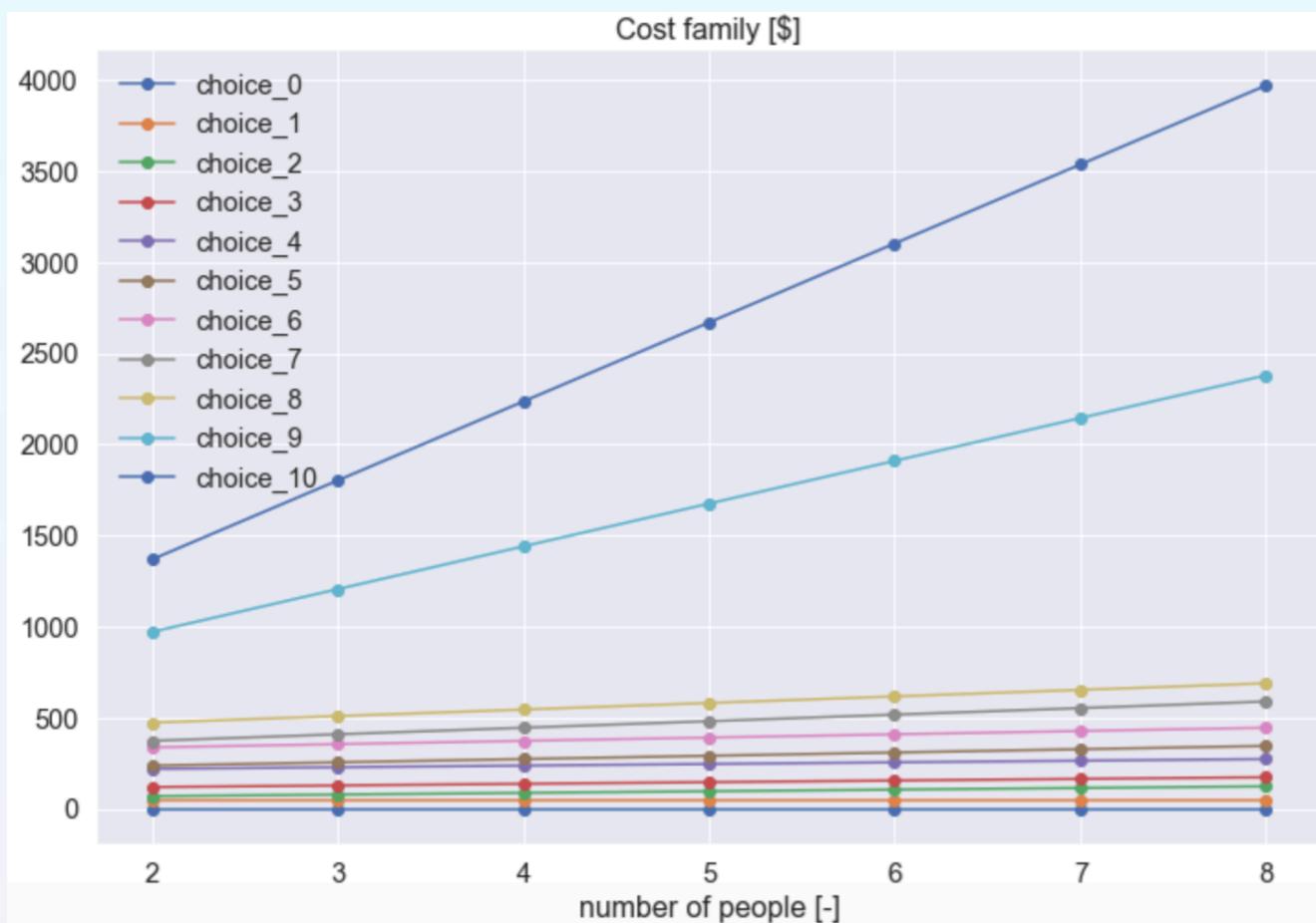
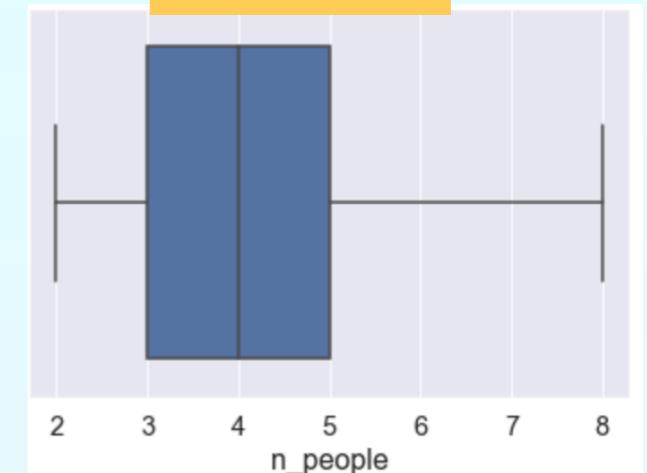
LA PROBLÉMATIQUE ET SON INTERPRÉTATION

- ▶ Différentes méthodes possibles :
 - ▶ plus simple : du type stochastique tel que algorithme génétique, ...
 - ▶ Optimizers MIP du commerce : Mixed Integer Programming
- ▶ Choix fait = méthode d'algorithme génétique
 - ▶ Entièrement développée
 - ▶ Combinée avec une autre méthode : « stochastic search ».
- ▶ Comparaison avec méthode MIP Gurobi

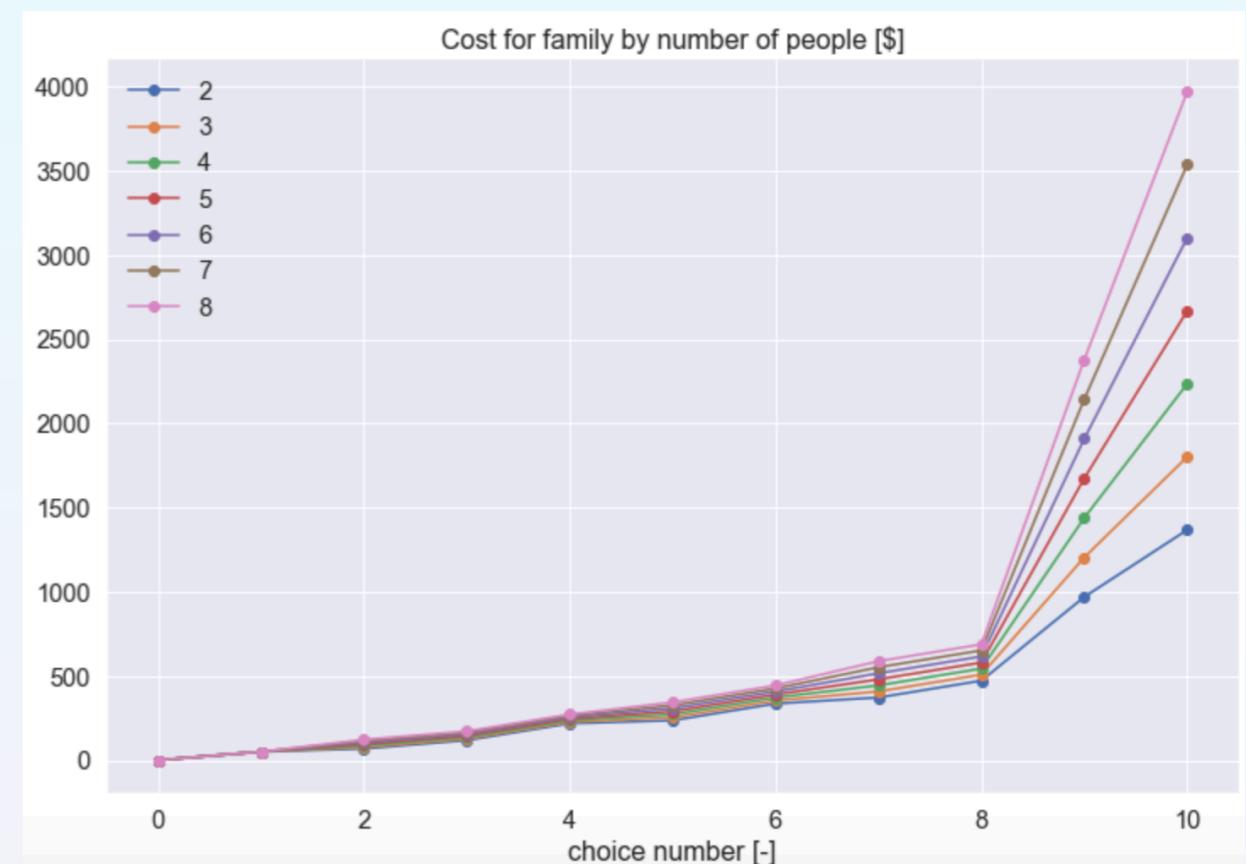
EXPLORATION

- ▶ N° Choix plus important que taille famille

MEDIANE : 4



CHOIX 9 -10 À ÉVITER!

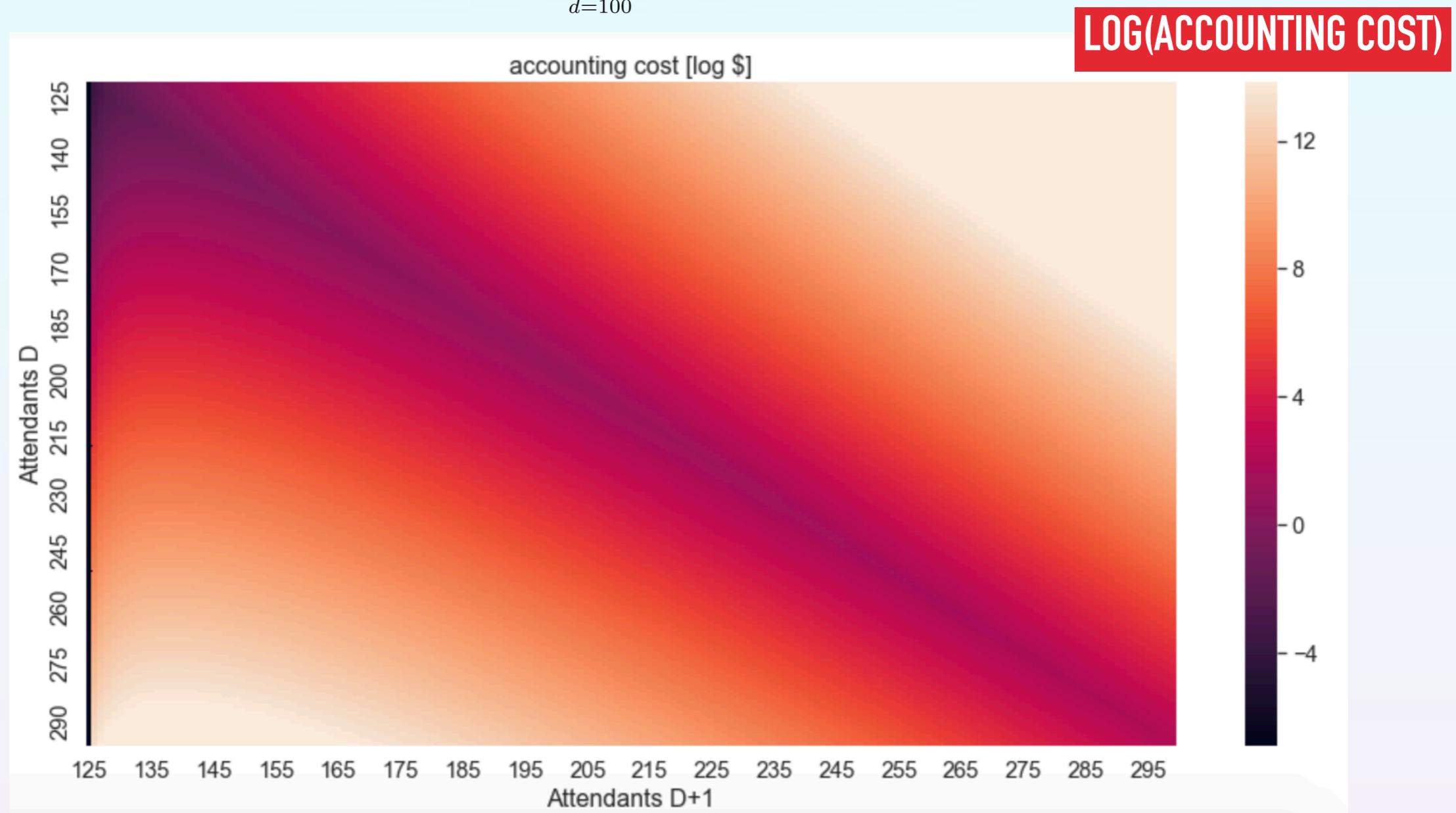


NB PEOPLE PEU IMPACTANT CHOIX ≤8

EXPLORATION

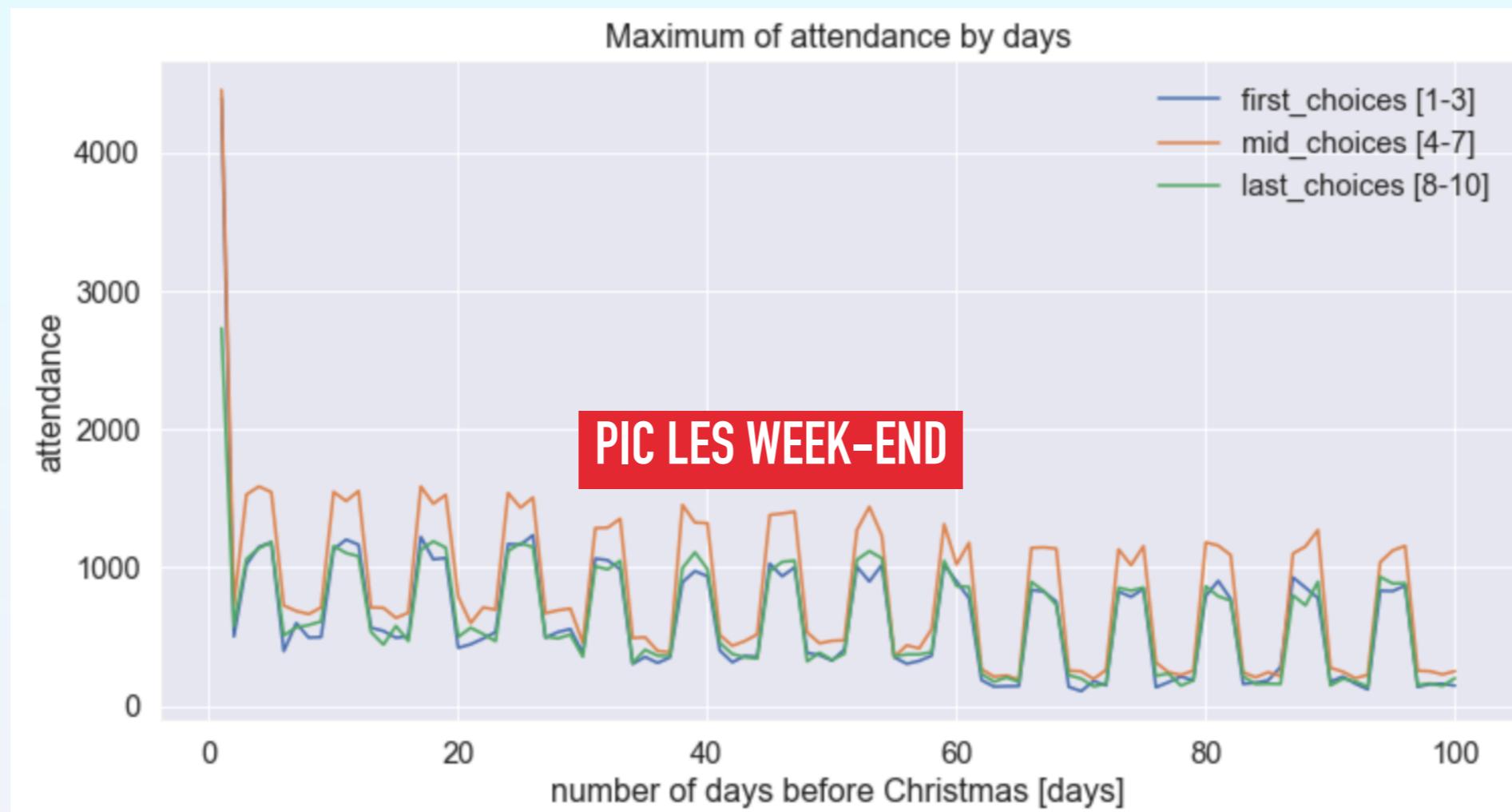
- ▶ Impact exponentiel de la différence d'affluence

$$\text{accounting penalty} = \sum_{d=100}^1 \frac{(N_d - 125)}{400} N_d^{(\frac{1}{2} + \frac{|N_d - N_{d+1}|}{50})}$$



EXPLORATION

- ▶ Très forte demande du jour avant Noël

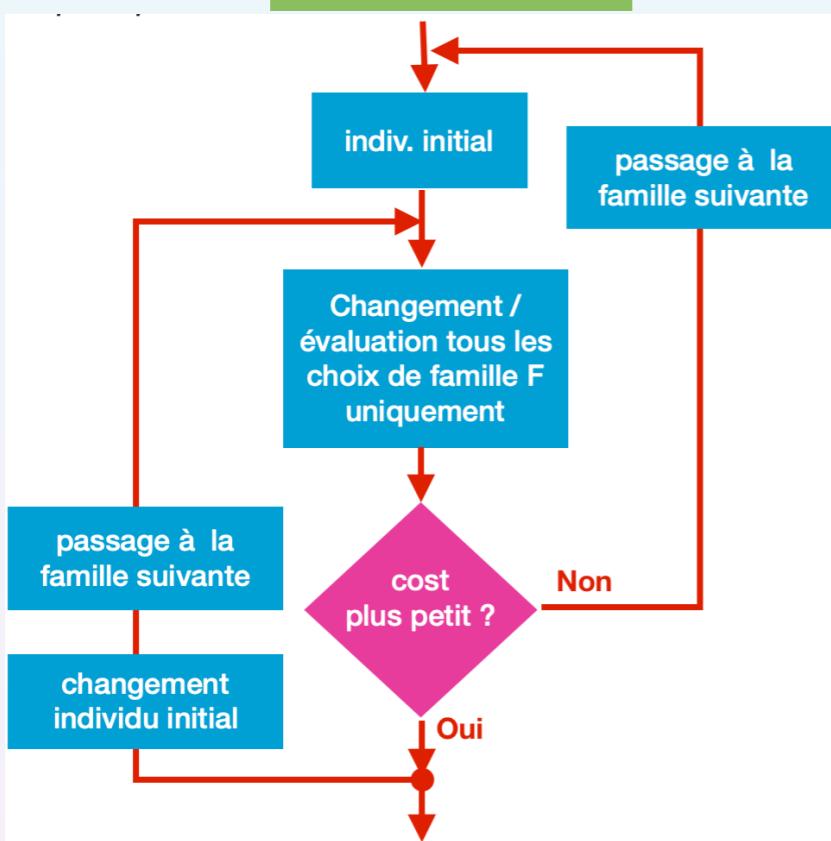


MODÈLE ALGORITHME GÉNÉTIQUE

▶ Principes similaires à l'évolution du vivant

- ▶ Pop. initiale : le sample proposé = le 1er individu
- ▶ Génération par optimisation « boost simple » suivant des chemins aléatoires

BOOST SIMPLE



EXEMPLE 10 CHEMINS

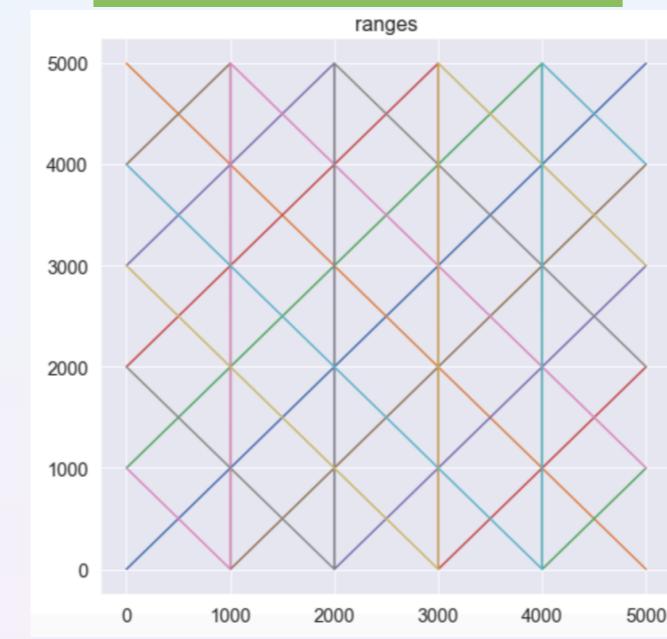
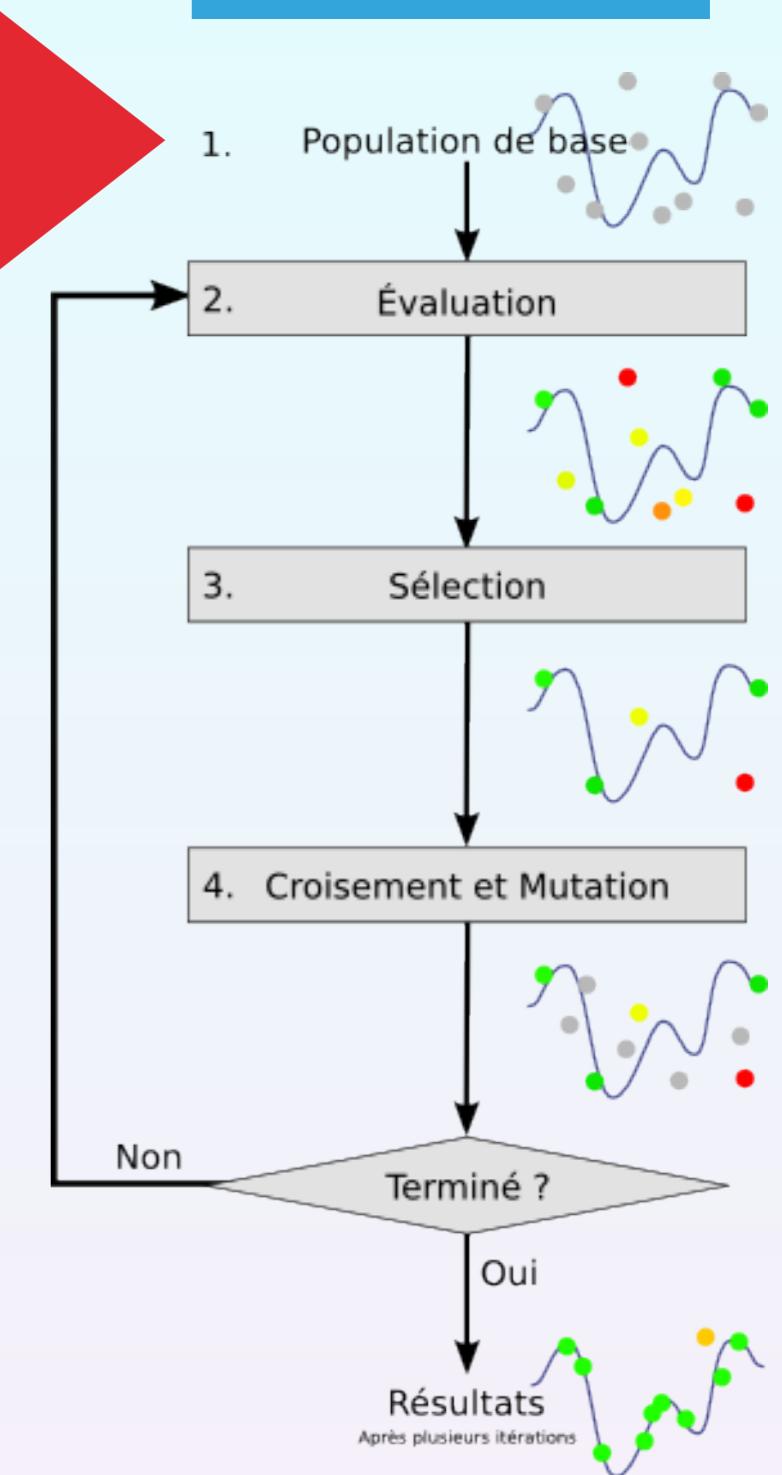


SCHÉMA DE PRINCIPE



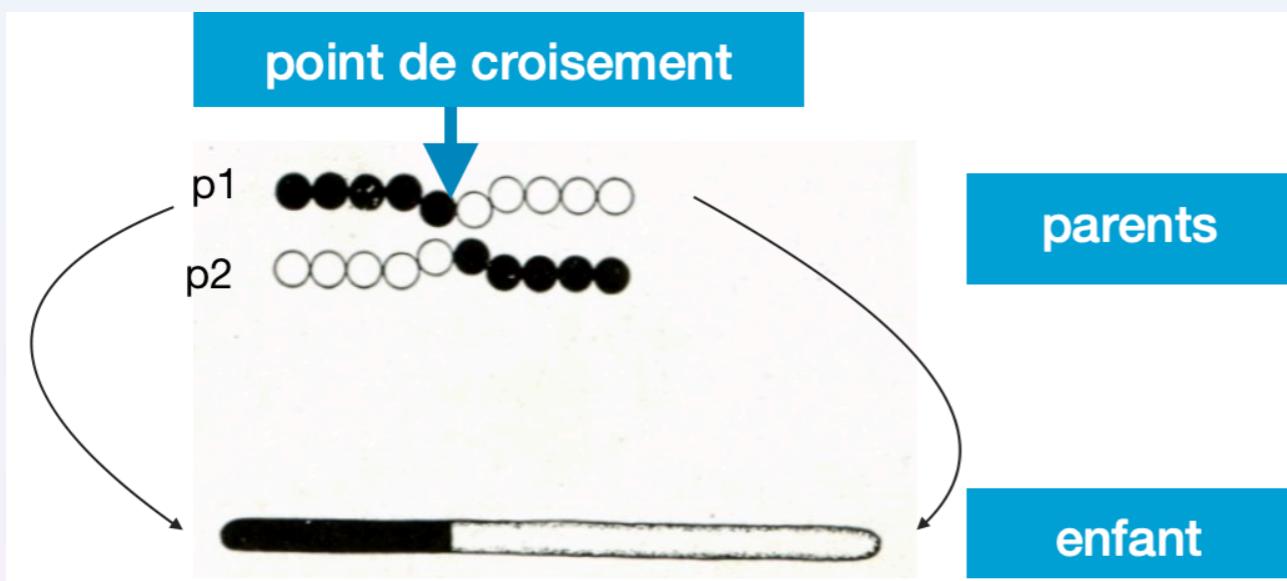
MODÈLE ALGORITHME GÉNÉTIQUE

▶ Sélection des meilleurs

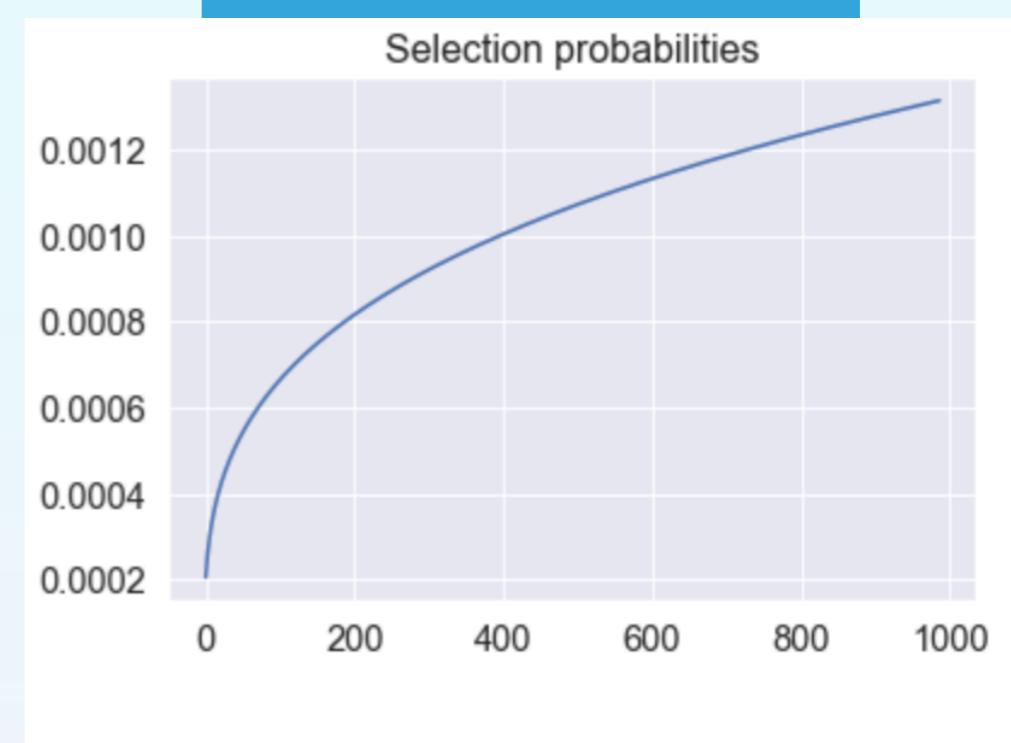
- ▶ Filtrage lors du croisement grâce à la probabilité de sélection.

▶ Totale régénération lors du ***croisement***

- ▶ Conservation uniquement des 20 meilleurs parents / 1000
- ▶ 2 parents donne 1 enfant
- ▶ 1 point de croisement aléatoire



PROBABILITÉ DE SELECTION INVERSE DU RANG



$$p_I = \frac{RANG_INVERSE_I^{pow_select}}{\sum RANG_INVERSE_I^{pow_select}}$$

MODÈLE ALGORITHME GÉNÉTIQUE

▶ Mutation partielle des individus

- ▶ % de pop mutée = 10% (R_POP_MUT)
 - ▶ 100 individus sur 1000
- ▶ % des dates changées d'un individu : 1% (R_MUT)
 - ▶ 50 dates sur 5000

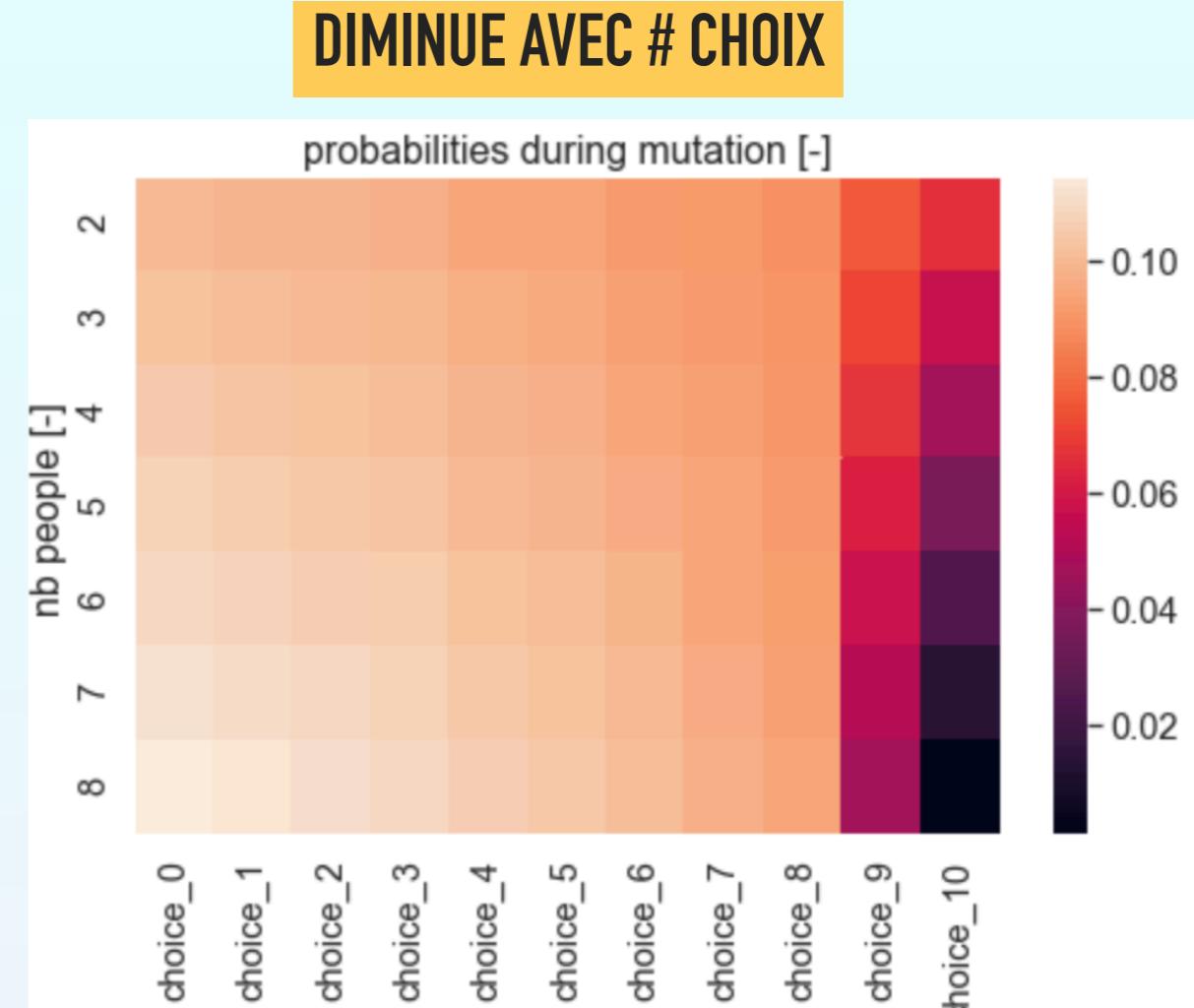
▶ Mutation sur choix des familles

- ▶ avec une fonction de probabilité

▶ Evaluation à chaque génération

▶ Optimisation du code avec numba

▶ *Boosting simple* toutes les 2000 epochs



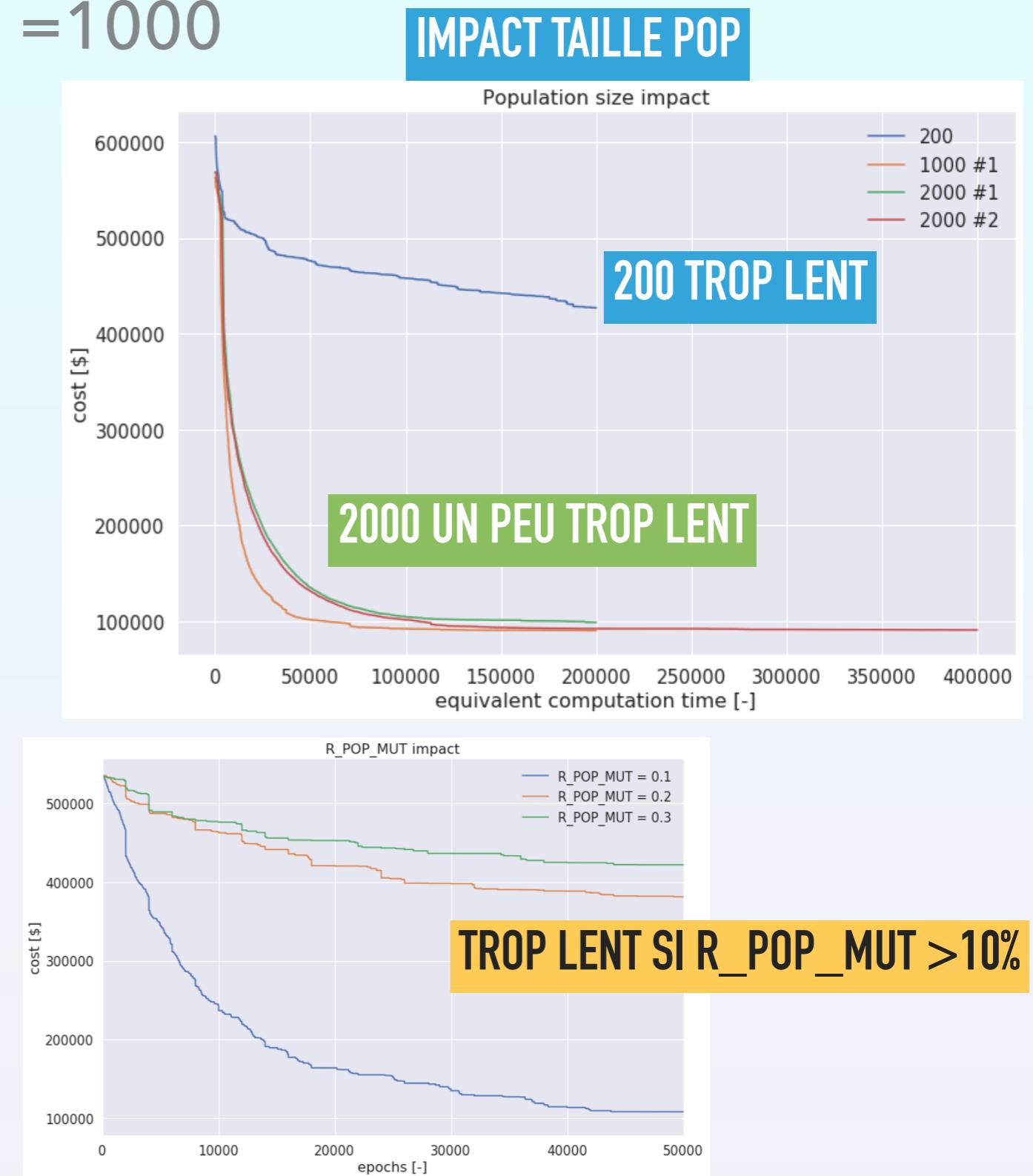
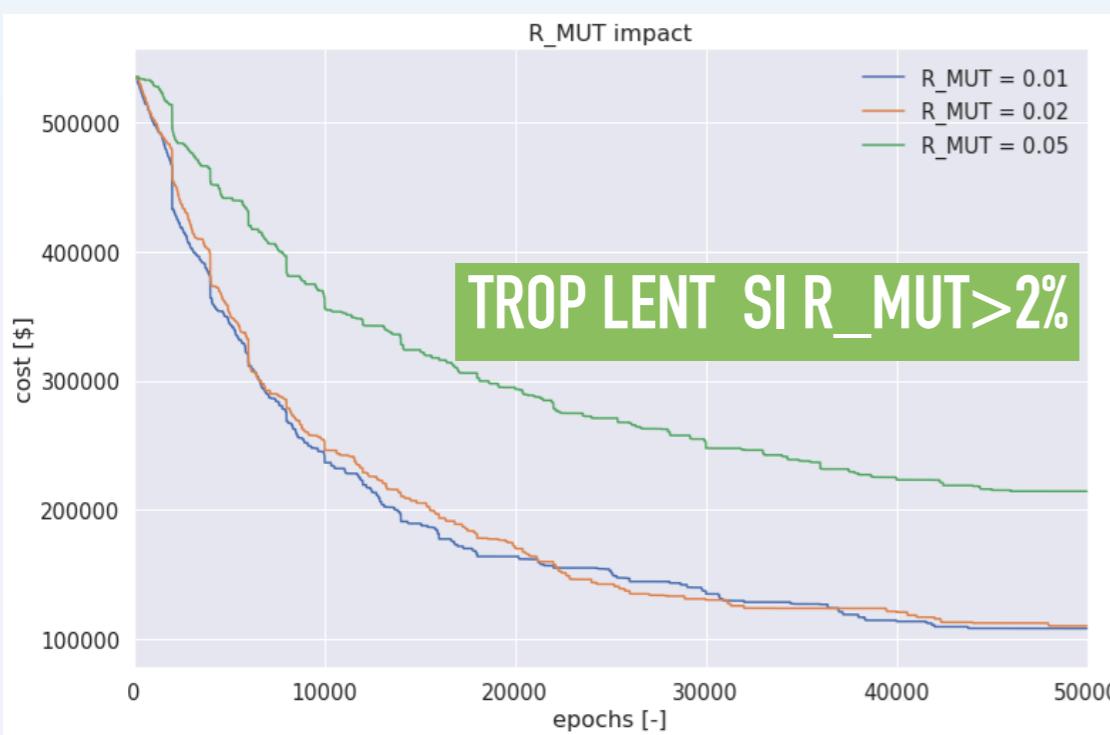
```
@njit(parallel=True, fastmath=True)
```

**OPTIMISATION EPOCH
PASSE DE 6s à 0.1s**

MODÈLE ALGORITHME GÉNÉTIQUE

- ▶ Taille population optimale = 1000

- ▶ par chemins aléatoires
- ▶ Forte Sensibilité au Taux de mutation

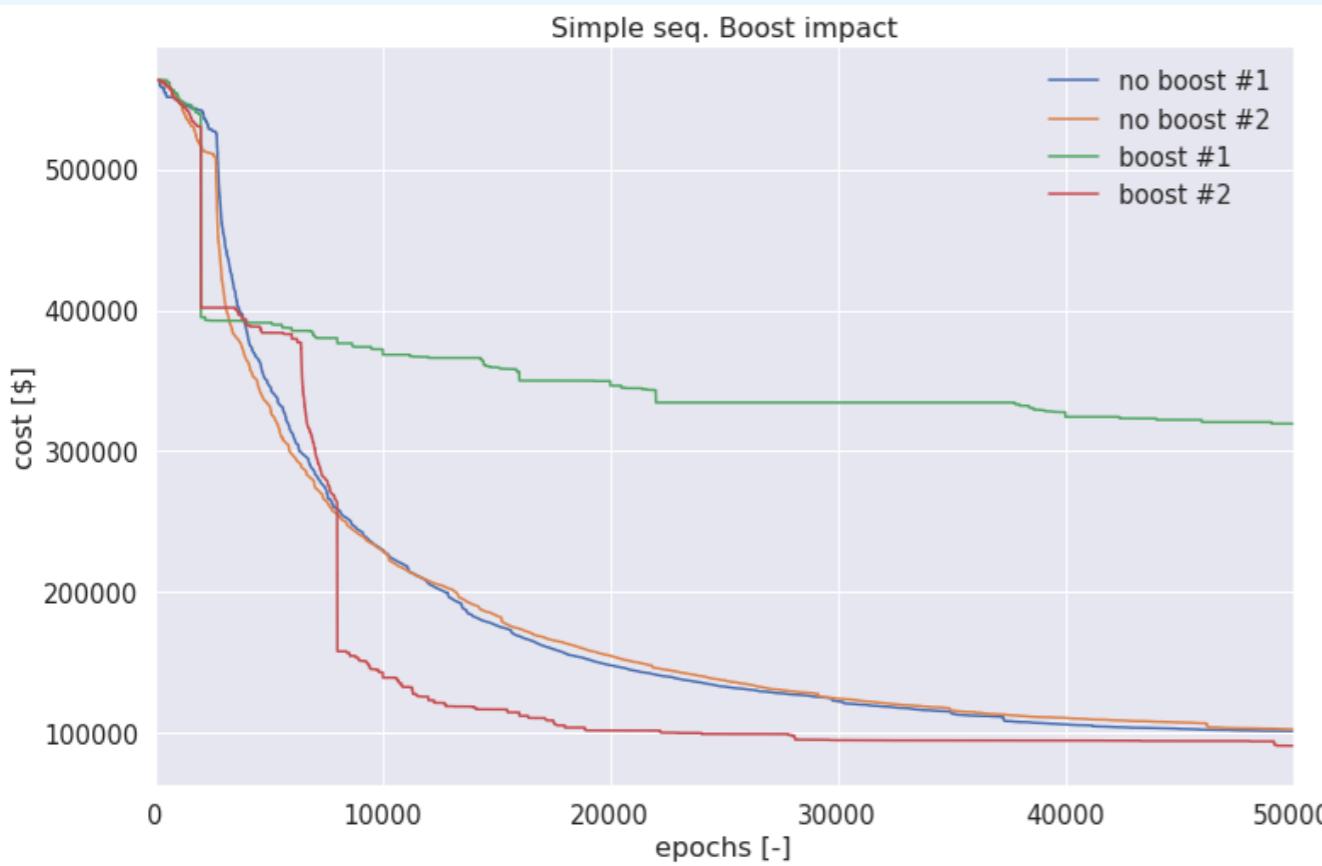


MODÈLE ALGORITHME GÉNÉTIQUE

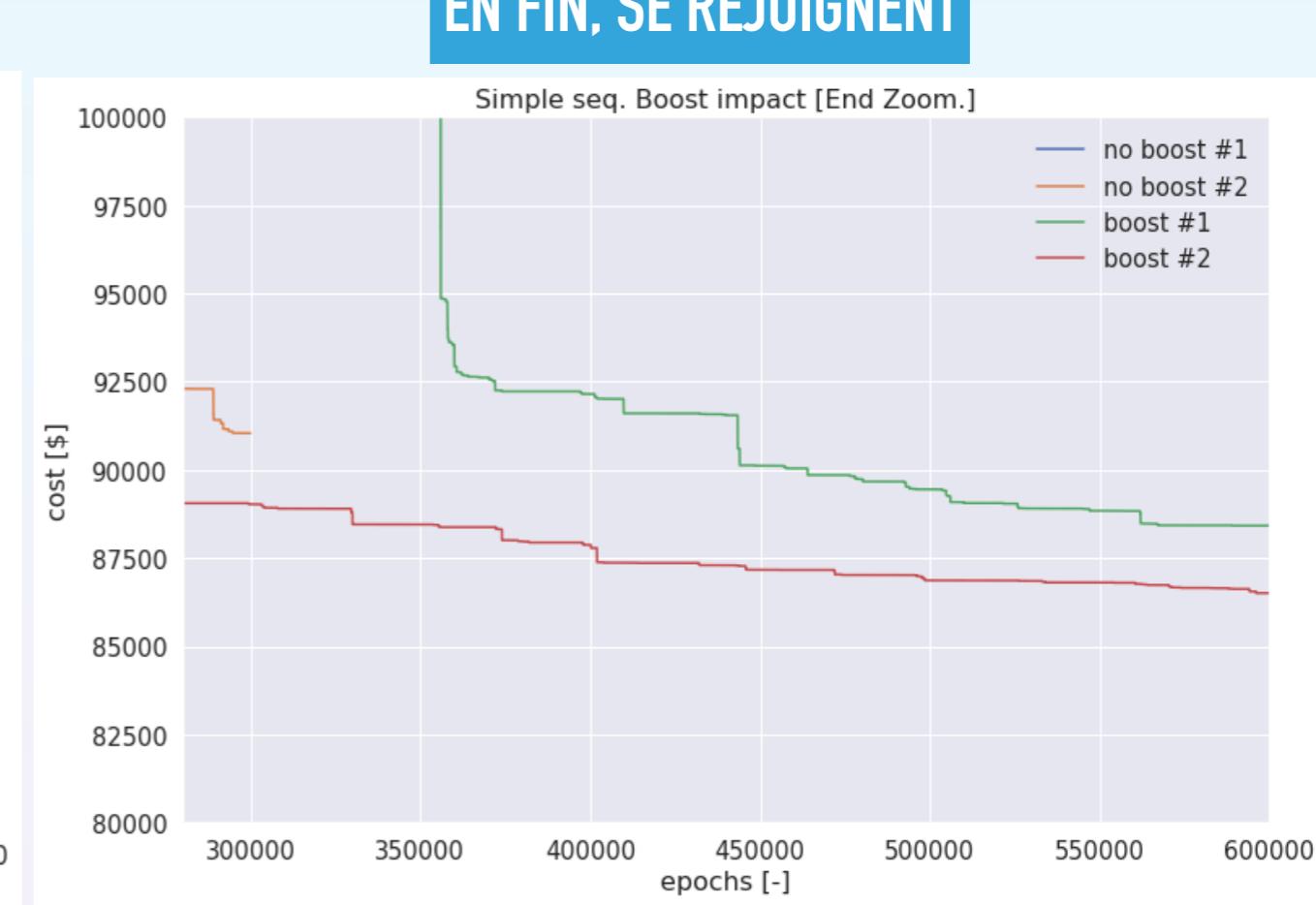
▶ ***Boost simple*** utile en début d'optimisation

- ▶ malgré blocage rare possible (1 fois sur 10)
- ▶ Moins utile en fin de convergence

MEILLEUR AU DÉBUT



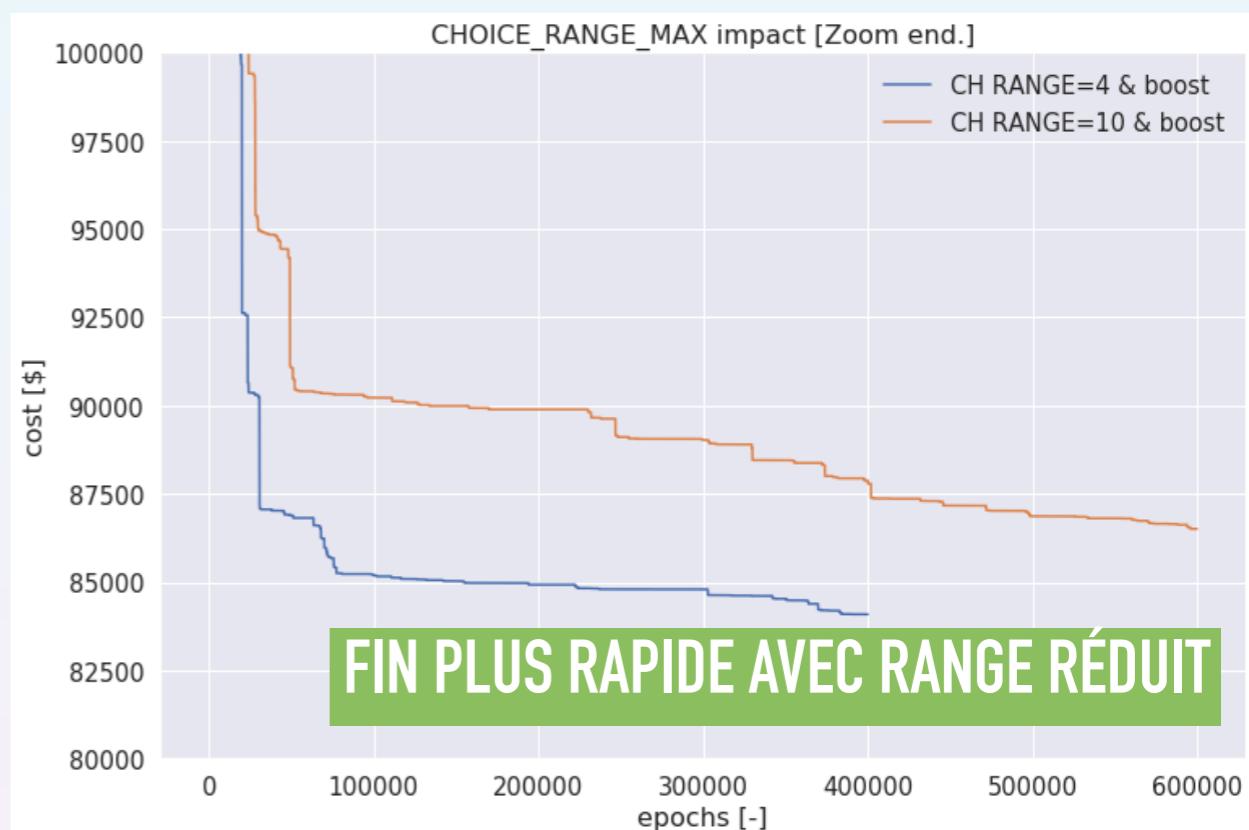
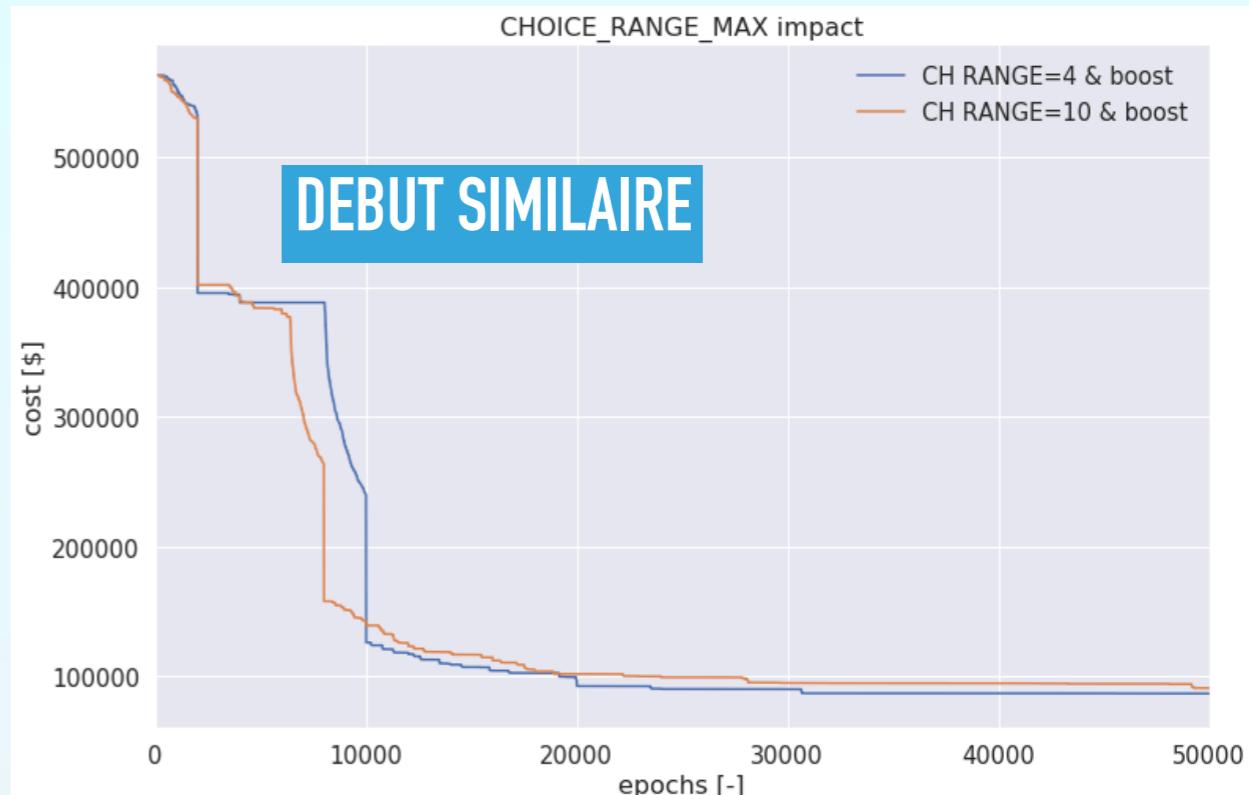
EN FIN, SE REJOIGNENT



MODÈLE ALGORITHME GÉNÉTIQUE

- ▶ Meilleur résultat = 84089
- ▶ 15h de calcul
- ▶ RANGE MAX = 5 premiers choix
- ▶ ***Boost simple*** actif
- ▶ 1000 individus
- ▶ R_MUT = 1%
- ▶ R_POP_MUT = 10%

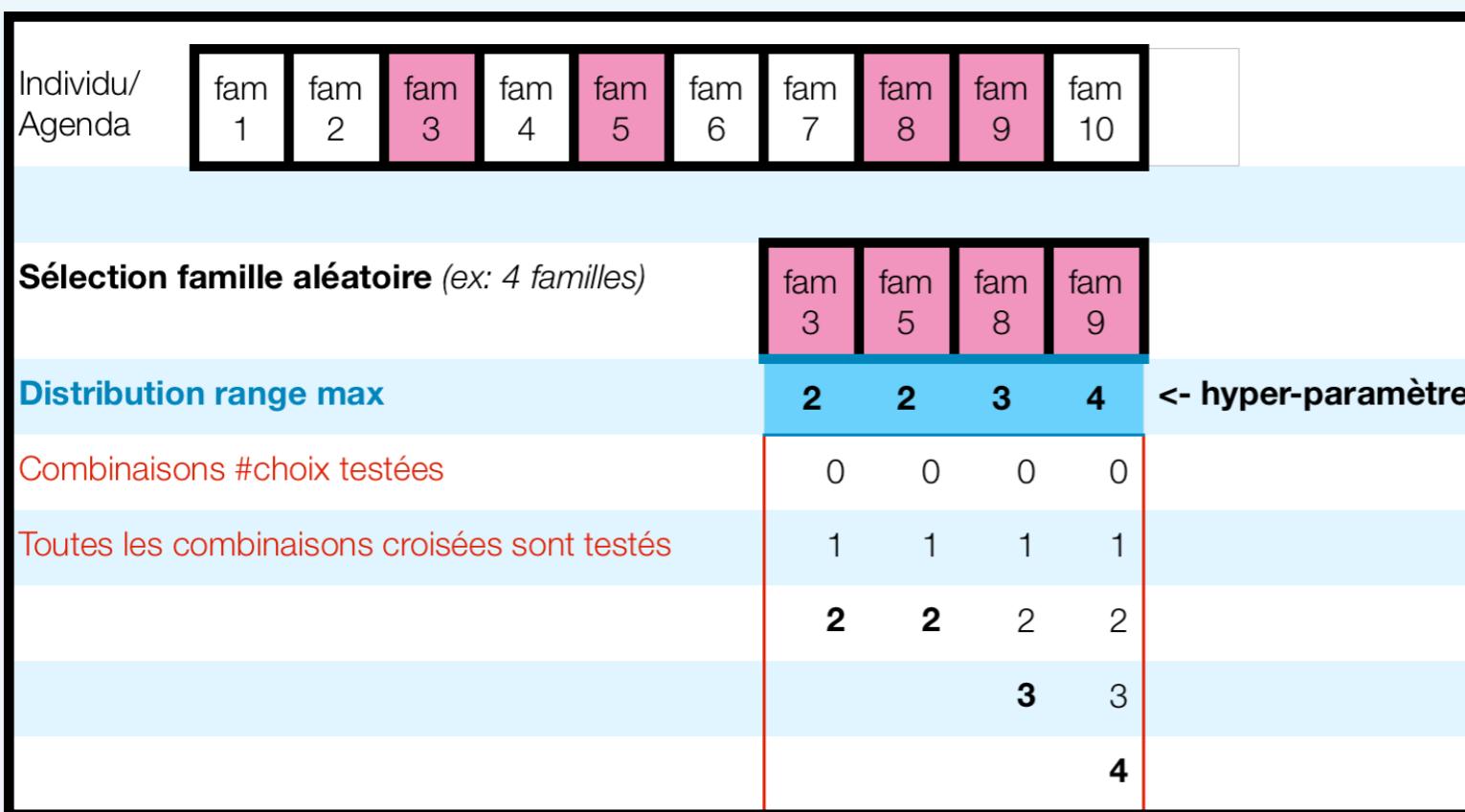
IMPACT DU RANGE MAXI



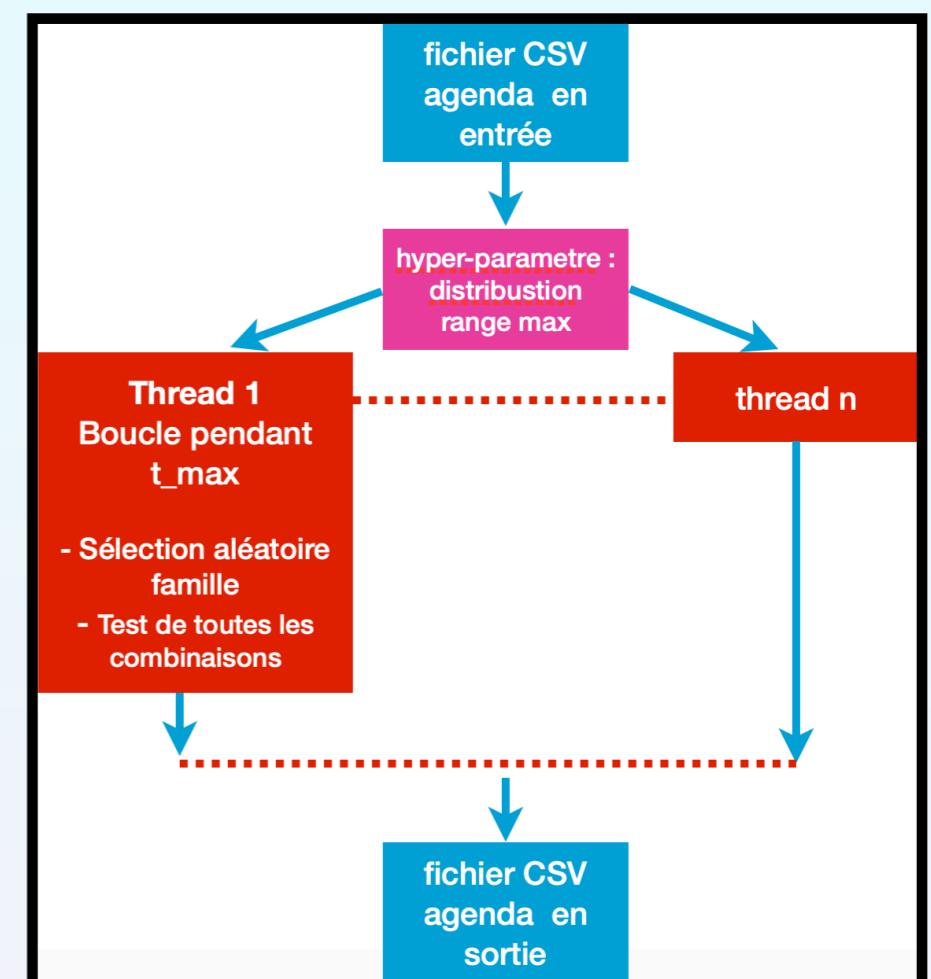
MODÈLE STOCHASTIC PRODUCT SEARCH

- ▶ Semblable à ***boost simple*** sur plusieurs familles en même temps
- ▶ Très performant car codée en C++
- ▶ **65e-9 s /calcul de coût (24e-6 s avec numba)**

PRINCIPE DU MODÈLE



PRINCIPE D'IMPLÉMENTATION EN PARALLÈLE



MODÈLE STOCHASTIC PRODUCT SEARCH

- ▶ Hyper-paramètre le plus efficace :

- ▶ 6 familles de distribution [2, 2, 2, 2, 3, 5]

NE MARCHE PAS SI INIT = SAMPLE => DEPART DEPUIS 1ERE OPTIM

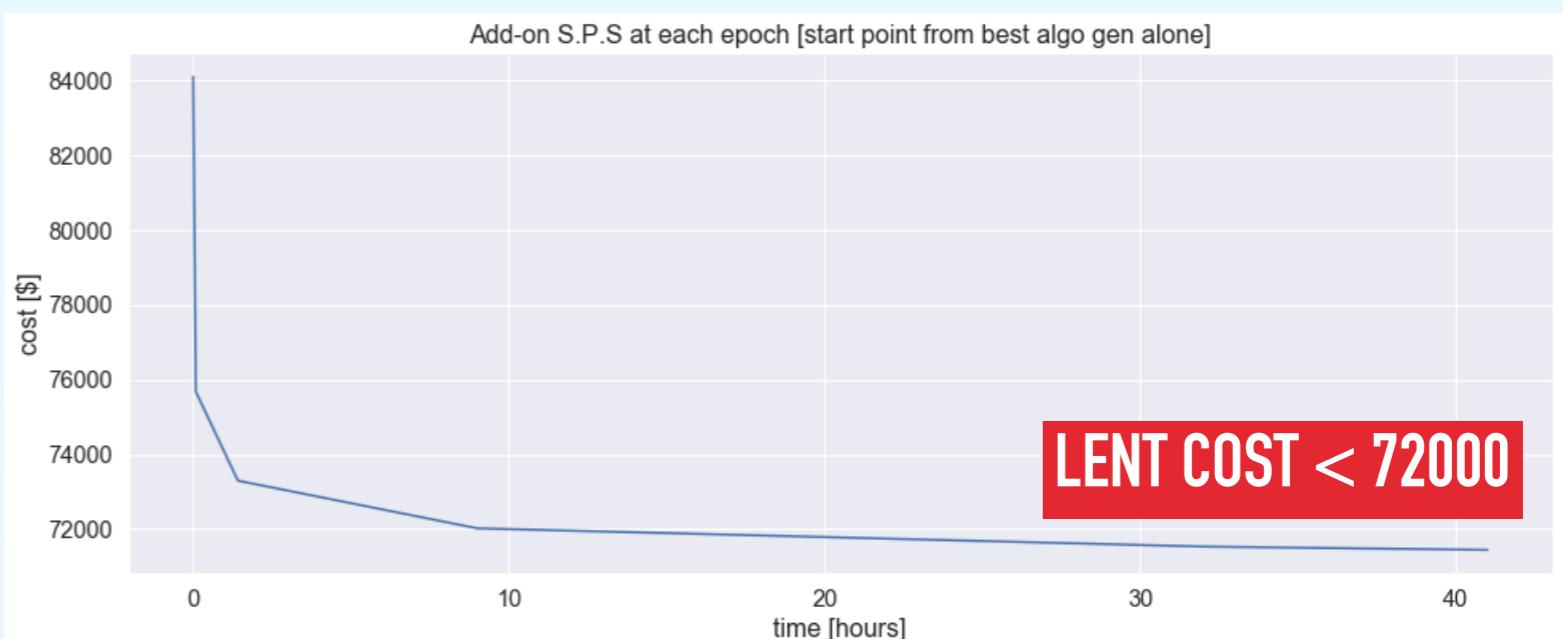
Point de départ	Distribution	Temps de calcul	cout de départ	cout final
sample fourni	pour toutes les distrib. testées	8h	5600000	5600000
meilleur avec méthode algo génétique	15 fam. {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 5}	8h	83000	73800
...	8 fam. {2, 2, 2, 2, 2, 2, 3, 5}	8h	83000	72600
...	6 fam. {2, 2, 2, 2, 3, 5}	8h	83000	72000
...	4 fam. {2, 2, 3, 5}	8h	83000	72193

> -10000\$ EN 8H

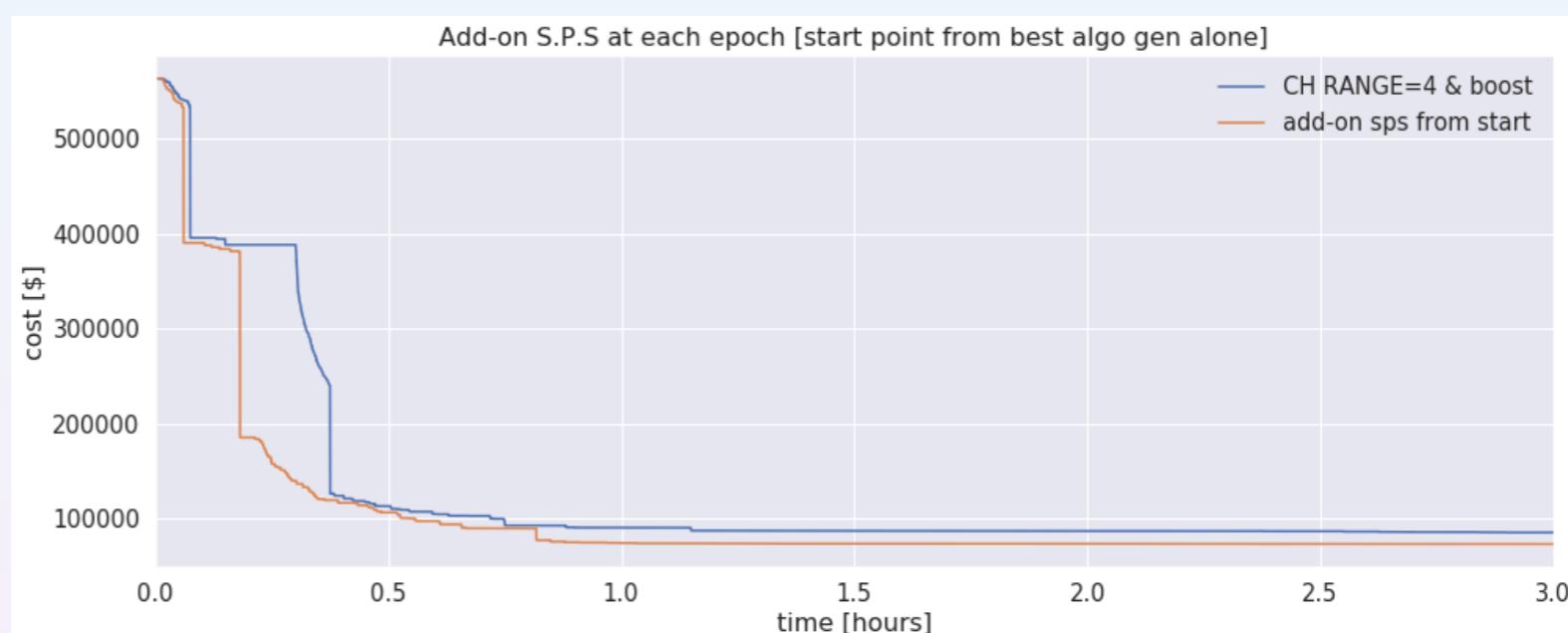
MODÈLE STOCHASTIC PRODUCT SEARCH

- ▶ Efficace aussi en add-on pour l'algo-génétique
- ▶ mais Lent si cost < 72000

epoch	time	cost
0	0	84089.677697
1500	300	75676.217504
30500	5100	73296.762613
120500	32400	72023.392795
174000	115200	71536.233524
194000	147600	71447.879463

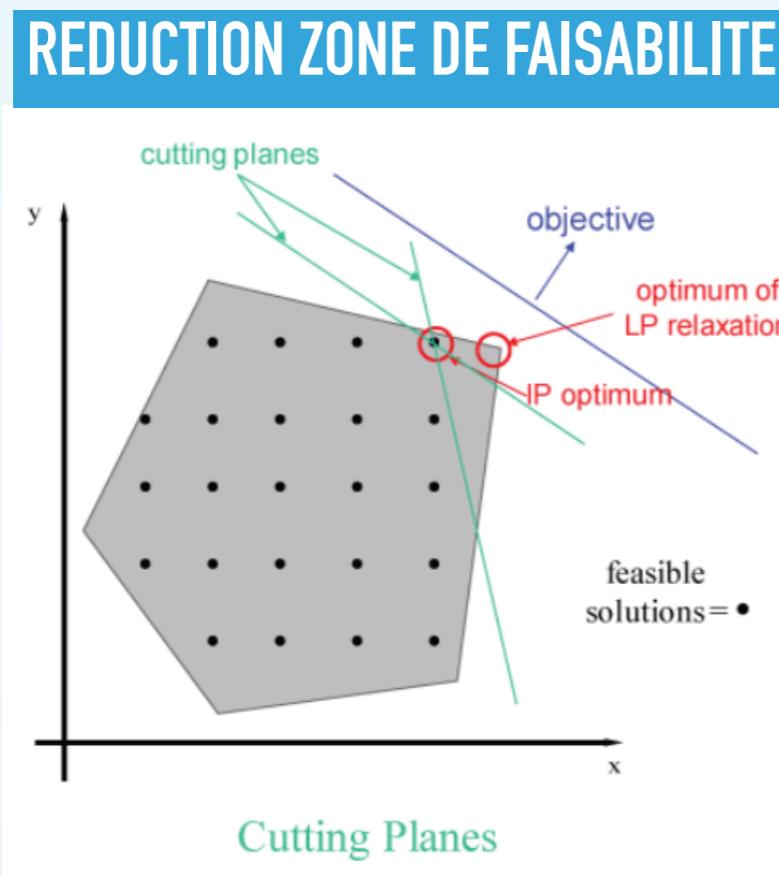


**ADD-ON DÈS LE DÉPART :
MEILLEUR DÉPART**

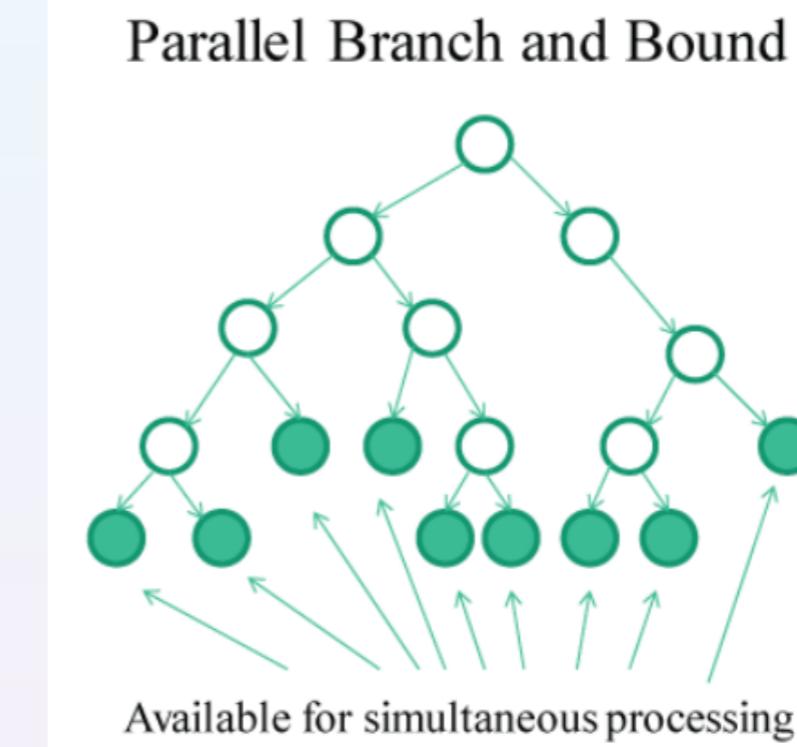


MODÈLE MIP GUROBI

- ▶ Optimiseur du commerce gagnant de la compétition
- ▶ Modélisation Mixed Integer Program
 - ▶ Utilisation des techniques : Cutting planes, Primal/Dual Simplex, Barrier et Branch&Bound



RÉSOLUTION DES SOUS-PROBLÈMES PARALÉLLISABLE



MODÈLE MIP GUROBI

- ▶ Nécessité de la réécriture du problème avec équations linéaires

NON-LINÉAIRE

preference cost + accounting penalty



FONCTION DE DÉCISION À MINIMISER

$$\sum_{f=1}^{5000} \sum_{c=1}^{10} \text{asm}_{f,c} * \text{PREF}_{f,c} + \sum_{d=1}^{100} \sum_{k=1}^{176} \sum_{l=1}^{176} \text{occ}_{d,k,l} * \text{ACC}_{k,l}$$

The objective of our MIP

$$\text{accounting penalty} = \sum_{d=100}^1 \frac{(N_d - 125)}{400} N_d^{(\frac{1}{2} + \frac{|N_d - N_{d+1}|}{50})}$$

**VARIABLES DE DÉCISION
= MATRICE BINAIRES**

asm : choix attribué = f(famille, num. choix)

occ : affluence = f(jour J, aff jour J, aff J+1)

NOMBREUSES CONTRAINTES REFORMULER

$$125 \leq N_d \leq 300$$



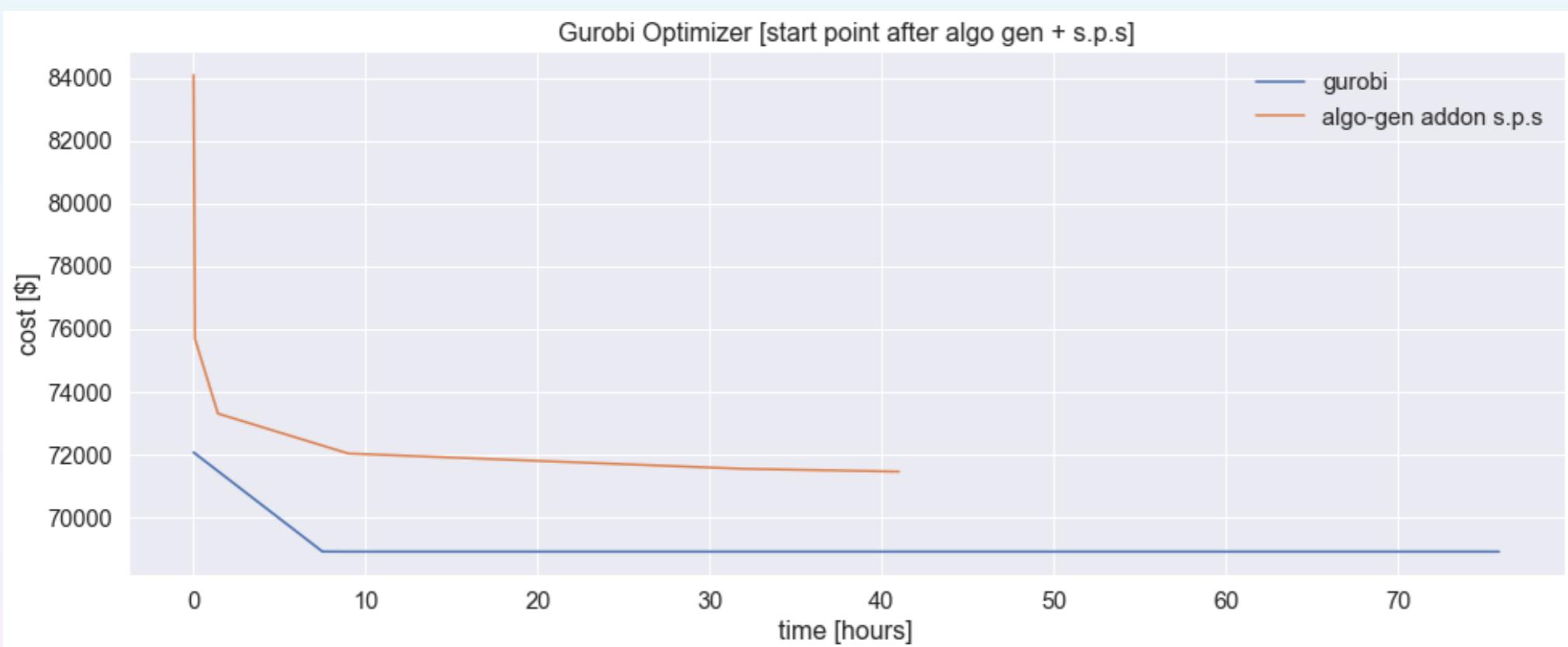
$$\text{occupancy}(d) = \sum_{f=1}^{5000} \sum_{c=1}^{10} \text{asm}_{f,c} * \text{family_size}_f * \mathbb{1}(\text{choices}_{f,c} = d)$$

Calculating the occupancy from the assignment matrix

MODÈLE MIP GUROBI

- ▶ Meilleur convergence au départ
- ▶ Mais lent par la suite vers l'optimum

Point de départ	Temps de calcul	coût de départ	coût final (avant optimal)	time	cost
meilleur résultat avec aglo gen + S.P.S	76h	72060	68895	0.000000	72060.800000
				7.491944	68899.094911
				8.796389	68895.761275
				75.886944	68895.761275



CONCLUSIONS

- ▶ Algo. génétique efficace en 1ère approche
- ▶ Performance de calcul accrue avec Numba
- ▶ Meilleur avec add-on Stochastique Product Search
- ▶ Gurobi : plus rapide pour trouver un meilleur minimum
- ▶ Mais très lent pour l'optimum
- ▶ Tout de même : Gurobi = modèle à recommander

AXES D'AMELIORATION

- ▶ Améliorer l'implémentation S.P.S dans l'algorithme génétique
- ▶ Si coder en C++ => plus rapide
- ▶ Pour S.P.S : étude hyper-paramètres peut être améliorée
- ▶ Optimiser les bornes pour Gurobi