Projet Systèmes de décision

Présenté par : Baddou Othmane / Bakoury Ayoub / Benaija Ismail

Plan de présentation

01 Contexte

02 Présentation du générateur de données

03 NCS SAT et MAXSAT

04 Single peaked

05 Nos résultats NCS

Contexte

Modélisation et l’implémentation de plusieurs algorithmes de décision.

Programme d'enseignement supérieur doit décider de l'admission d'étudiants sur la base de leurs évaluations dans 4 cours. Basé sur résultats à la "majorité" de cours, sinon, l'étudiant est refusé. Du point de vue de la commission, tous les cours (critères) n'ont pas la même importance.

intéressons particulièrement au cas d’apprentissage des paramètres de ce modèle décisionnel à partir d’un jeu de données

Les algorithmes proposés

01

MR–SORT

02

NCS SAT

03

NCS MAXSAT

04

Single Peaked version

MR–SORT

Premier algorithme du projet

MR-Sort est un système de décision qui trie les éléments en classes en fonction de leur évaluation sur chaque critère en utilisant certains paramètres. L'objectif ici est d'apprendre ces paramètres à partir des décisions qui ont été prises (dans le jeu de données).

$$
\begin{cases}
max\ \alpha & \\
\qquad \sum_{i \in N} c_{ij} + x_j + \varepsilon = \lambda & \forall a_j \in A^{*1} \\
\qquad \sum_{i \in N} c_{ij} = \lambda + y_j & \forall a_j \in A^{*2} \\
\qquad \alpha \leq x_j,\ \alpha \leq y_j & \forall a_j \in A^{*} \\
\qquad c_{ij} \leq w_i & \forall a_j \in A^{*}, \forall i \in N \\
\qquad c_{ij} \leq \delta_{ij} & \forall a_j \in A^{*}, \forall i \in N \\
\qquad c_{ij} \geq \delta_{ij} - 1 + w_i & \forall a_j \in A^{*}, \forall i \in N \\
\qquad M\delta_{ij} + \varepsilon \geq g_i(a_j) - b_i & \forall a_j \in A^{*},\ \forall i \in N \\
\qquad M(\delta_{ij} - 1) \leq g_i(a_j) - b_i & \forall a_j \in A^{*},\ \forall i \in N \\
\qquad \sum_{i \in N} w_i = 1,\ \lambda \in [0.5, 1] & \\
\qquad w_i \in [0, 1] & \forall i \in N \\
\qquad c_{ij} \in [0, 1],\ \delta_{ij} \in \{0, 1\} & \forall a_j \in A^{*},\ \forall i \in N \\
\qquad x_j, y_j \in \mathbb{R} & \forall a_j \in A^{*} \\
\qquad \alpha \in \mathbb{R} &
\end{cases}
$$

Programme mathématique MR–Sort dans le cas de 2 catégories

Nos résultats

Le générateur de données

Nous utilisons un script pour générer aléatoirement la donnée pour tous nos algorithmes présentés. Il est possible d'ajouter du bruit. La figure à droite présente le format données sous forme csv

Les inputs sont:
-Nombre de matières
-Nombre d'instances
-Nombre de classes
-Bruit en pourcentage

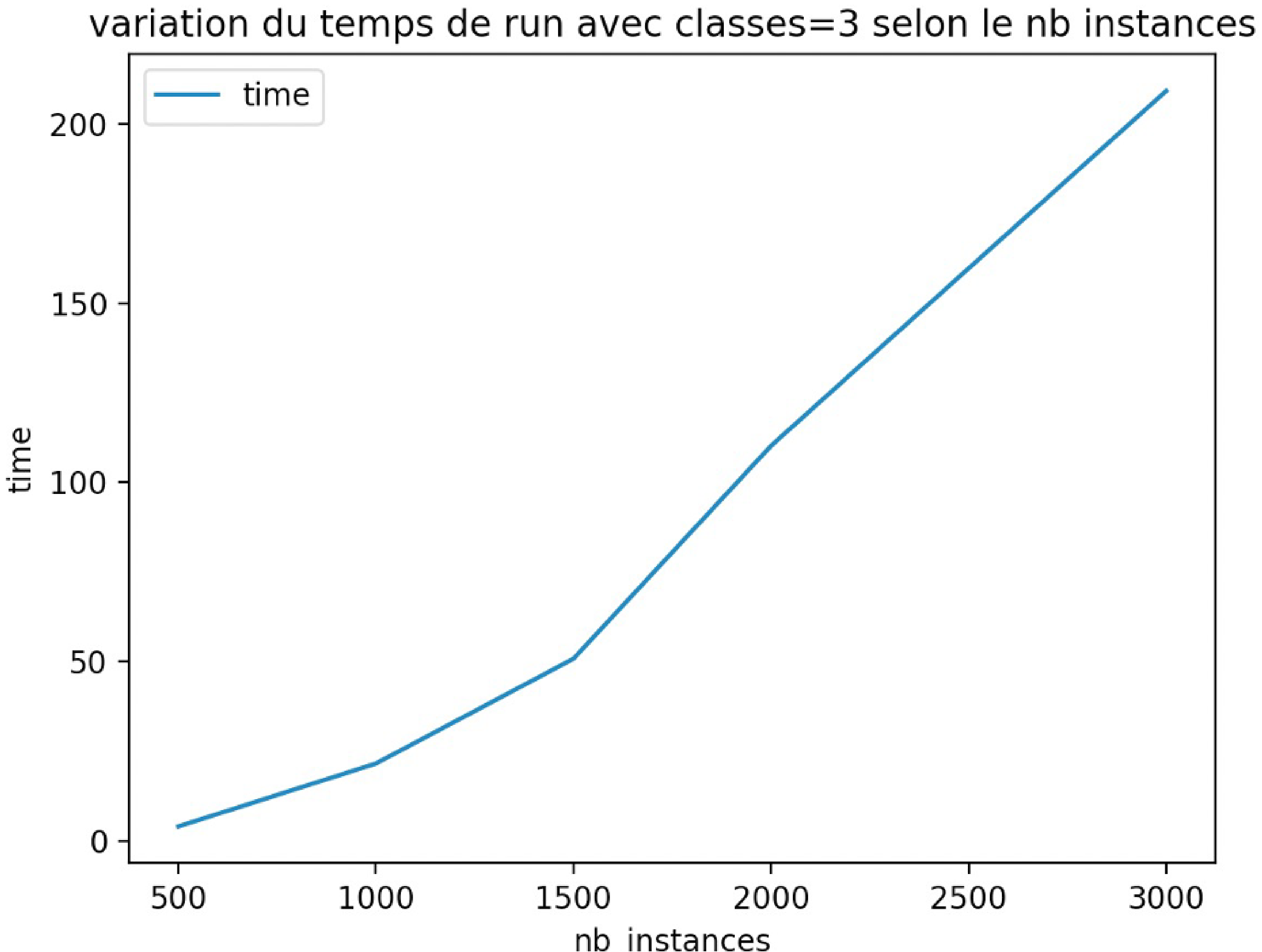
	0	1	2	3	4	accepted
0	15	12	16	9	17	0
1	0	14	11	14	10	0
2	3	14	10	7	14	0
3	2	19	16	13	6	1
4	16	9	6	18	9	0
5	19	15	14	19	0	1
6	20	0	9	11	14	0
7	11	16	17	3	12	0
8	18	13	4	16	17	0
9	14	15	15	8	6	1
10	8	12	20	3	13	0

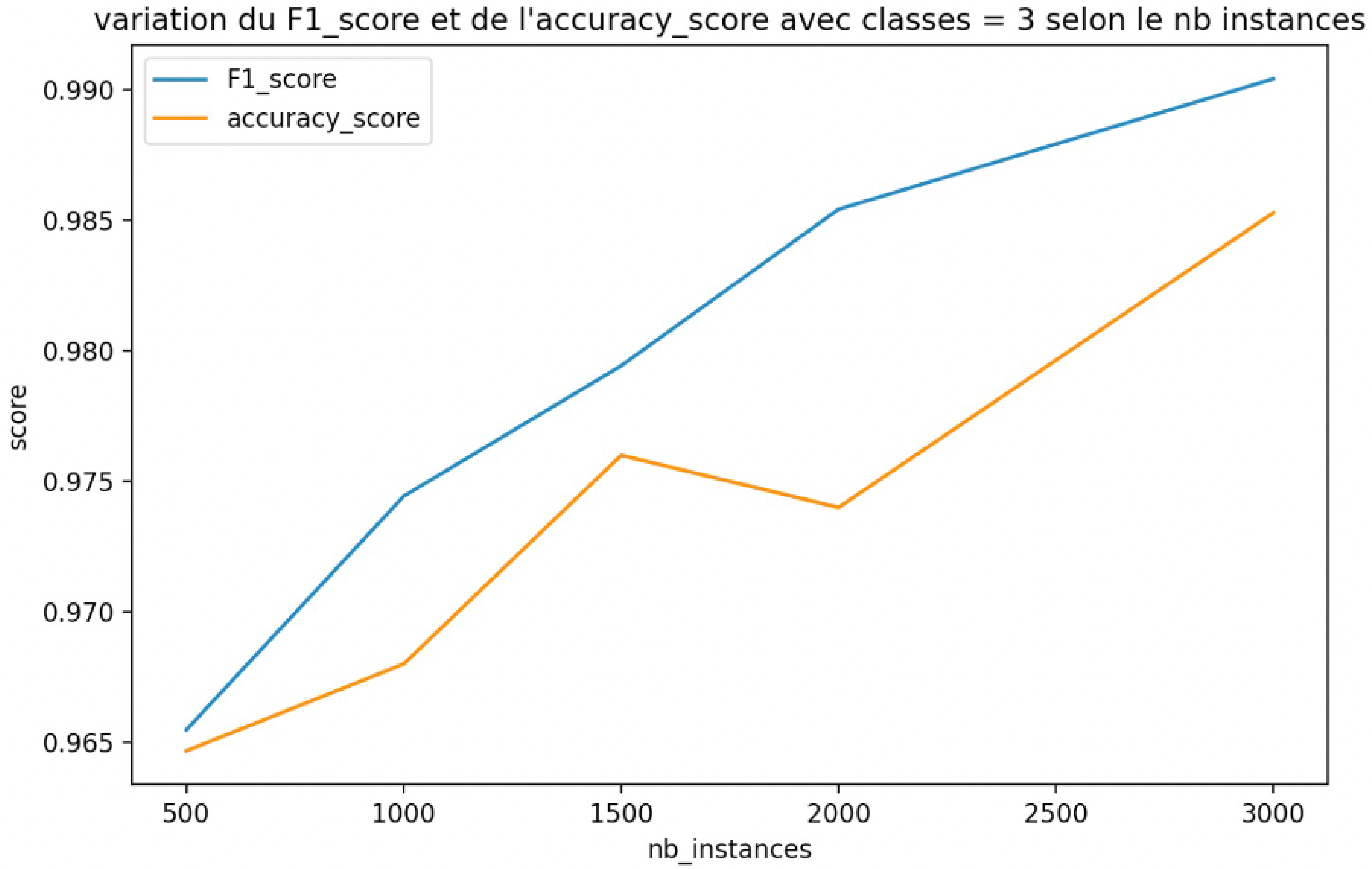
MR–SORT

Nos résultats

Impact du nombre d'instances

Le nombre d'instances améliore les performances mais augmente le temps de computation



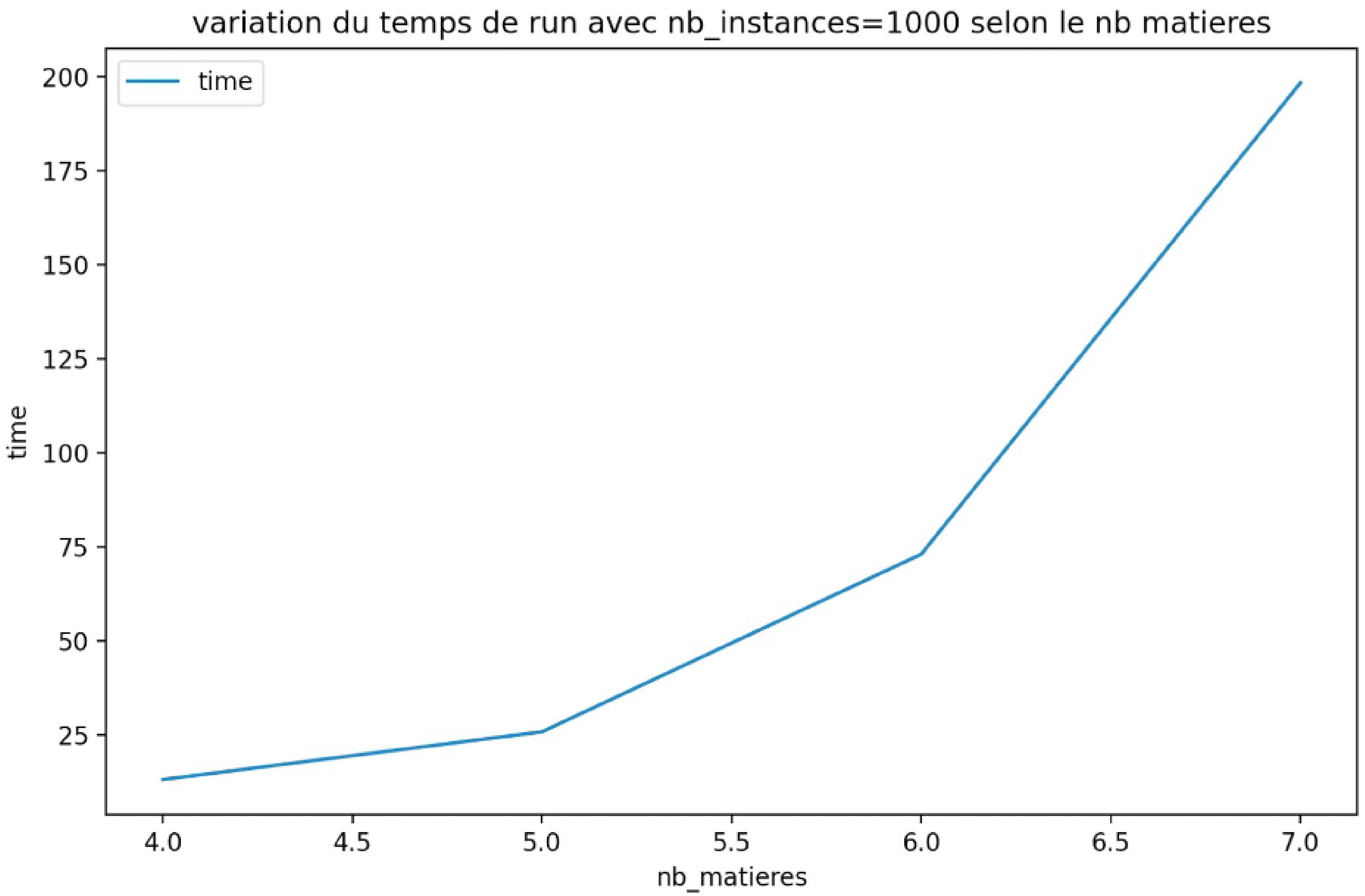


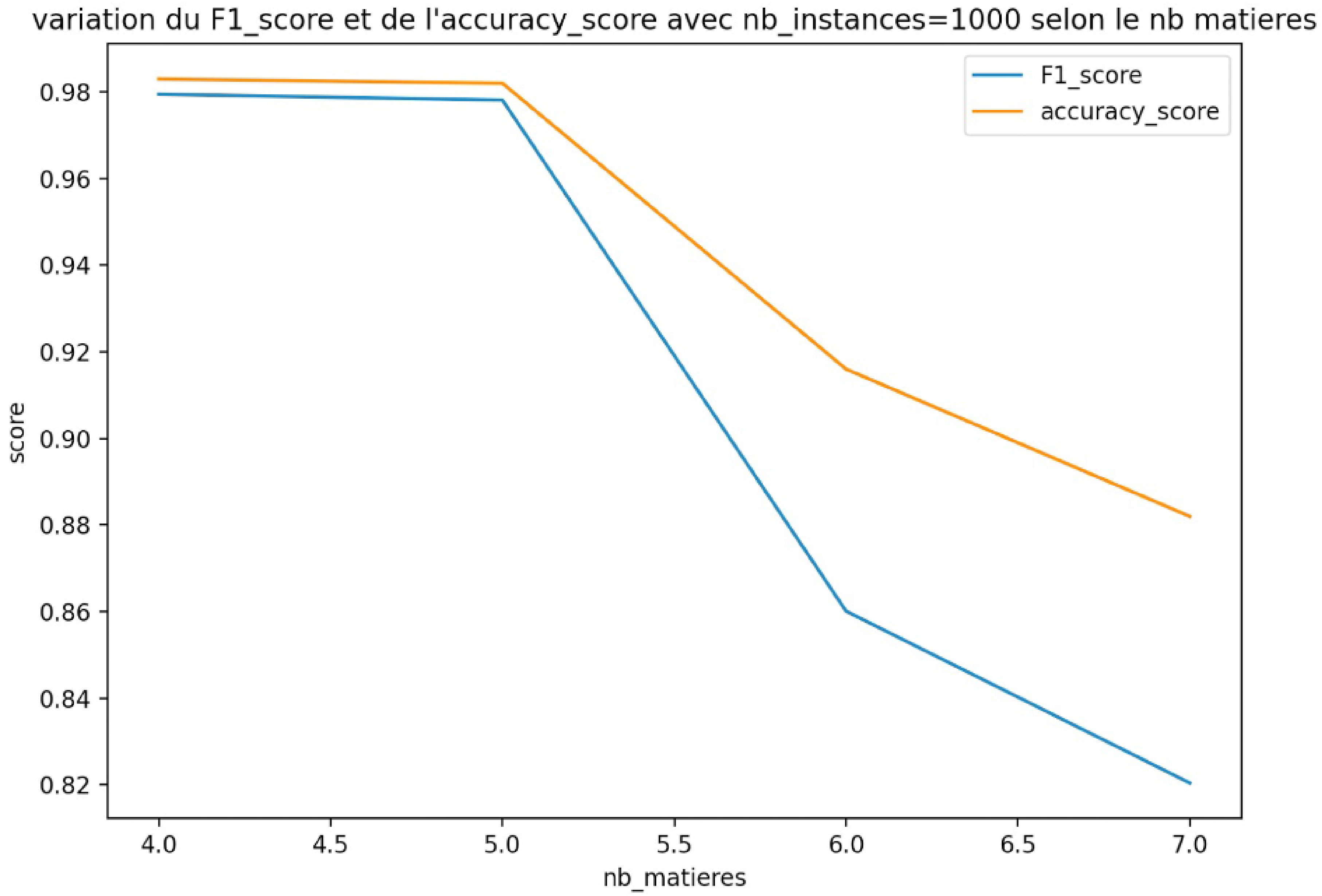
MR–SORT

Nos résultats

Impact du nombre de matières

Plus le nombre de matières augmente plus les performances en termes d'accuracy et de F1 score diminuent et le temps de computation augmente





MR-SORT

Impact du bruit

Le bruit est introduit en modifiant aléatoirement la classe de certaines instances. Le nombre de signaux bruités peut être controlés avec un paramètre.

```
Nombre de signaux bruités: 31
lambda truth: 0.59
lambda mrsort: 0.63
weight truth: ['0.20', '0.22', '0.16', '0.21', '0.21']
weight mrsort: ['0.25', '0.25', '0.00', '0.25', '0.25']
Temps d'exécution : 1.8e+02s
F1-score: 0.6024742041493113
accuracy: 0.66
```

En bruitant 3% des signaux, le F1-score diminue et passe de 0.9 à 0.6

NCS–SAT

Second algorithme du projet

Il existe d’autres méthodes permettant d’apprendre ces paramètres :

- MILP qui assurent une bonne restitution mais qui n’est pas applicable à de larges jeux de données.

- Heuristique : large jeu de données mais n'assure pas des paramètres optimaux

A SAT formulation based on coalitions

Binary Variables

- $x_{i,h,k}$: on criterion i is the value k sufficient at level h or not ?
- y_B : is coalition $B \subseteq \mathcal{N}$ sufficient or not ?

Clauses

1. **Ascending scales** : for all criteria, frontiers and ordered pairs of values, $k < k'$, $x_{i,h,k'} \vee \neg x_{i,h,k}$
2. **Hierarchy of profiles** : for all criteria, values and ordered pairs of frontiers, $h < h'$, $x_{i,h,k} \vee \neg x_{i,h',k}$
3. **Coalitions strength** : for all ordered pairs of coalitions, $B \subset B'$, $y_{B'} \vee \neg y_B$
4. **Alternatives are outranked by boundary above them** : for all coalitions, frontiers and alternatives a assigned immediately below the frontier $(\bigvee_{i \in B} \neg x_{i,h,u_i}) \vee \neg y_B$
5. **Alternatives outrank the boundary below them** : for all coalitions, frontiers and alternatives b assigned immediately below the frontier $(\bigvee_{i \in B} x_{i,h,a_i}) \vee y_{\mathcal{N} \setminus B}$

NCS–MAXSAT

Version MaxSat du NCS

- Le NCS ne convient pas au problème à partir de données réelles, car elle ne tolère pas la présence de bruit dans les données.

- Relaxation des formulations de décision : au lieu de trouver un modèle NCS qui restaure tous les exemples de l'ensemble d'apprentissage nous essayons de trouver le modèle qui en restaure le plus : d'où l'appellation MaxSAT.

this purpose, we define the following additional binary variables:

- 'z' variables, indexed by an alternative x, represent the set of alternatives properly classified by the inferred model, with the following semantic: $z_x = 1 \Leftrightarrow \alpha^{-1}(x) = NCS_\omega(x)$ i.e. the alternative x is properly classified

These variables are introduced in some clauses to serve as switches:

- For any exigence level $k \in [2.p]$, let $B \subseteq \mathcal{N}$ a coalition of criteria, and x an alternative assigned to C^{k-1} by α. If $z_k = 1$ and $B \subseteq \{i \in \mathcal{N} : x \in \mathcal{A}_i^k\}$ then $t_{B,k} = 0$. This leads to the following conjunction of clauses:

$$\phi_\alpha^{\widetilde{C5}} = \bigwedge_{B \subseteq \mathcal{N},\ k \in [2.p]} \bigwedge_{x \in \alpha^{-1}(C^{k-1})} \left(\bigvee_{i \in B} \neg a_{i,k,x} \vee \neg t_{B,k} \vee \neg z_x\right)$$

- For any exigence level $k \in [2.p]$, let $B \subseteq \mathcal{N}$ a coalition of criteria, and x an alternative assigned to C^k by α. If $z_k = 1$ and $B \subseteq \{i \in \mathcal{N} : x \in \mathcal{A}_i^k\}$ then $t_{\mathcal{N} \setminus B,k} = 0$. This leads to the following conjunction of clauses:

$$\phi_\alpha^{\widetilde{C6}} = \bigwedge_{B \subseteq \mathcal{N},\ k \in [2.p]} \bigwedge_{x \in \alpha^{-1}(C^k)} \left(\bigvee_{i \in B} a_{i,k,x} \vee t_{\mathcal{N} \setminus B,k} \vee \neg z_x\right)$$

The objective in the MaxSAT formulation is to maximize the portion of alternatives properly classified, this is the subject of the following soft clause:

$$\phi_\alpha^{goal} = \bigwedge_{x \in \mathbb{X}^*} z_x \tag{8}$$

The MaxSAT extension of the first formulation is the conjunction of the first four clauses of the SAT formulation given in Definition 4.1 and clauses $\phi_\alpha^{\widetilde{C5}}$, $\phi_\alpha^{\widetilde{C6}}$ and ϕ_α^{goal}.

Clauses composing the conjunctions ϕ_α^{C1}, ϕ_α^{C2}, ϕ_α^{C3}, ϕ_α^{C4}, $\phi_\alpha^{\widetilde{C5}}$ and $\phi_\alpha^{\widetilde{C6}}$ are hard, associated to the weight $w_{\max}$, and we associate to ϕ_α^{goal} the weight w_1 such that $w_{\max} > |\mathbb{X}^*| w_1$.

Ajoute d'une variable Z et de clauses supplémentaires

Single–Peaked

Données non monotones

Données sur les critères ne sont pas toujours monotones. Pour le single-peaked les critères ne sont pas à maximiser ou minimiser

On peut aborder ce problème en remplaçant l'inégalité de la clause par un intervalle

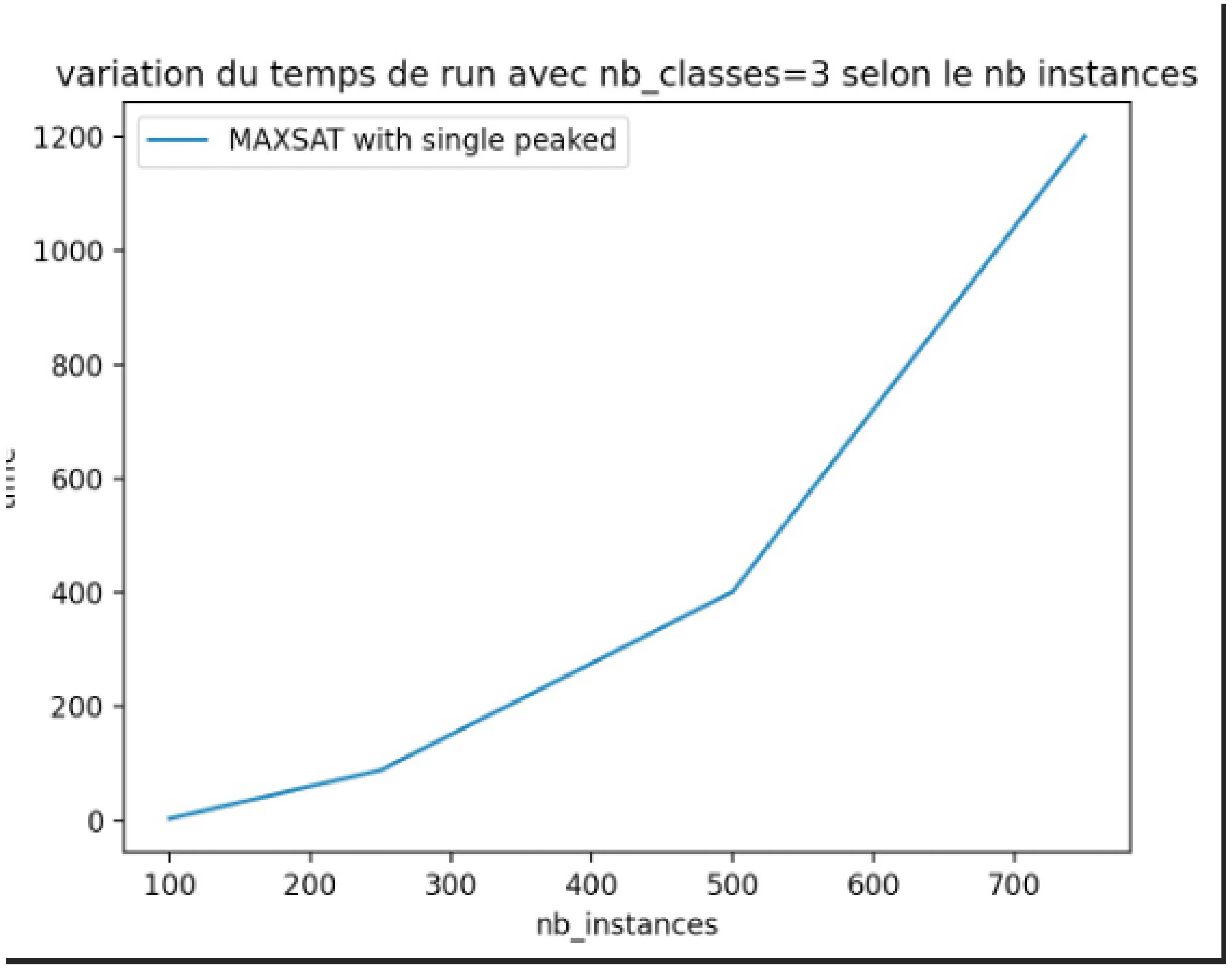
```
## Clauses 2a et 2b : Si un élève valide avec la note k,
## alors tous ceux qui valident avec une note k'>k valident aussi
## Adaptation sous forme d'intervalle pour la version single peaked

if single_peak :
    clause2a = []
    for i in subjects:
            for h in range(1, val):
                for k in X[i]:
                    for k1 in X[i]:
                        for k2 in X[i]:
                            if k < k1 and k1 < k2:
                                clause2a.append(
                                    [x[(i, h, k1)],-x[(i, h, k)],-x[(i, h, k2)]])
else :
    clause2a = [
        [x[i, h, p], -x[i, h, k]] for h in range(1, val) for i in subjects for k in X[i] for p in X[i] if k < p
    ]
```

Remplacement de l'inégalité par l'intervalle

Temps de computation

Nos résultats

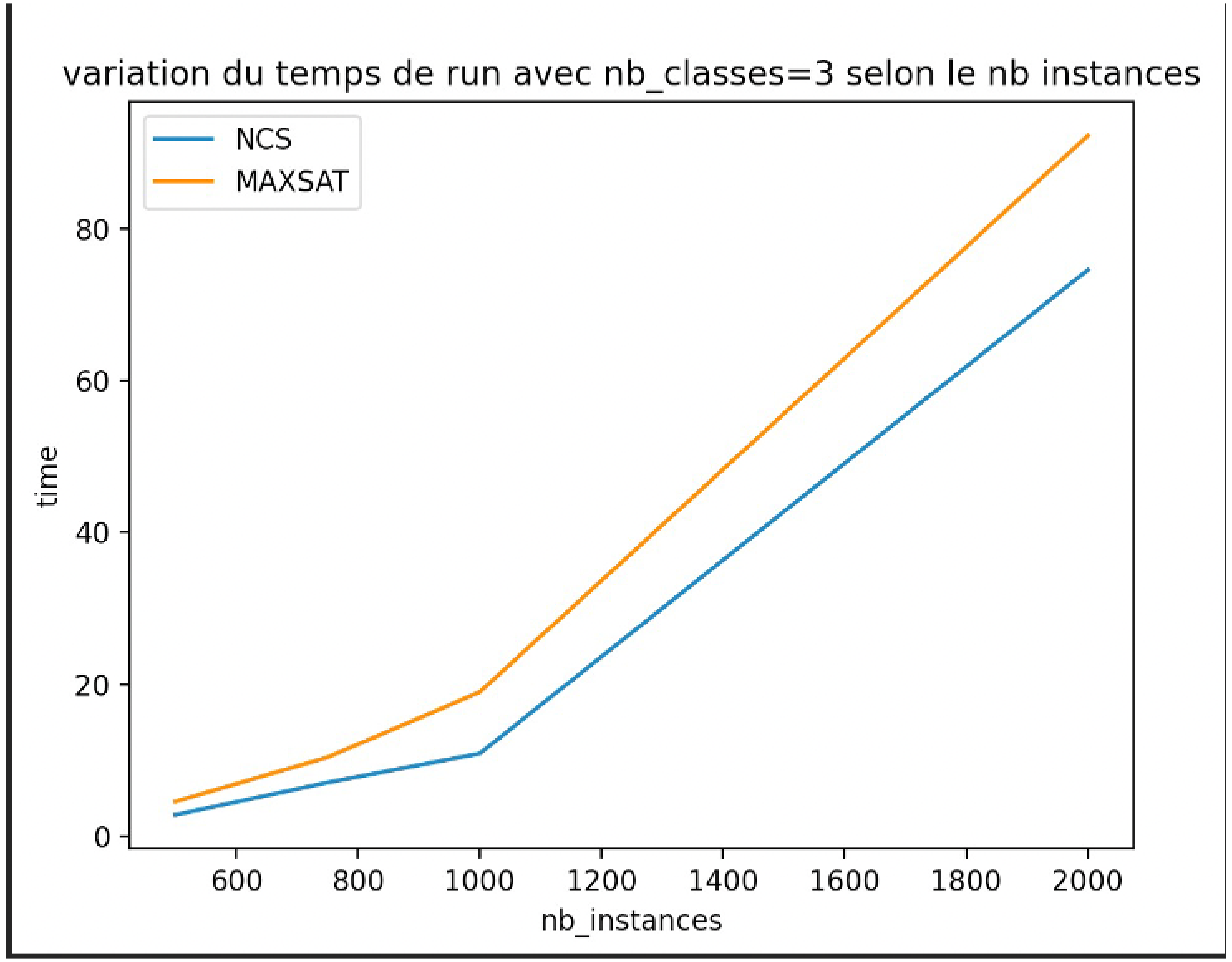


Impact du nombre d'instances

Le nombre d'instances augmente le temps de computation pour les 3 algorithmes. Cependant, en comparaison avec MR-SORT, NCS est plus efficace

Données avec bruit

Le NCS ne fonctionne pas : problème non satisfiable



Conclusion

Nous avons testé trois modèles et comparé les performances.

Piste d'amélioration :

- Comparer les F1 scores
- Comparer avec d'autres modèles (ML)
- Utiliser d'autres solveurs
- Utiliser plus de temps de calculs pour les graphiques

References:

Bouyssou, D., & Marchant, T. (2007). An axiomatic approach to noncompensatory sorting methods in MCDM, II : More than two categories. European Journal of Operational Research, 178(1), 246-276. https://doi.org/10.1016/j.ejor.2006.01.033

Belahcène, K., Labreuche, C., Maudet, N., Mousseau, V., & Ouerdane, W. (2018). An efficient SAT formulation for learning multiple criteria non-compensatory sorting rules from examples. Computers & Operations Research, 97, 58-71. https://doi.org/10.1016/j.cor.2018.04.019

Uyanık, E., Sobrie, O., Mousseau, V., & Pirlot, M. (2017). Enumerating and categorizing positive Boolean functions separable by a k-additive capacity. Discrete Applied Mathematics, 229, 17-30. https://doi.org/10.1016/j.dam.2017.04.010

Merci !

N'hésitez pas à nous contacter
si vous avez des questions.