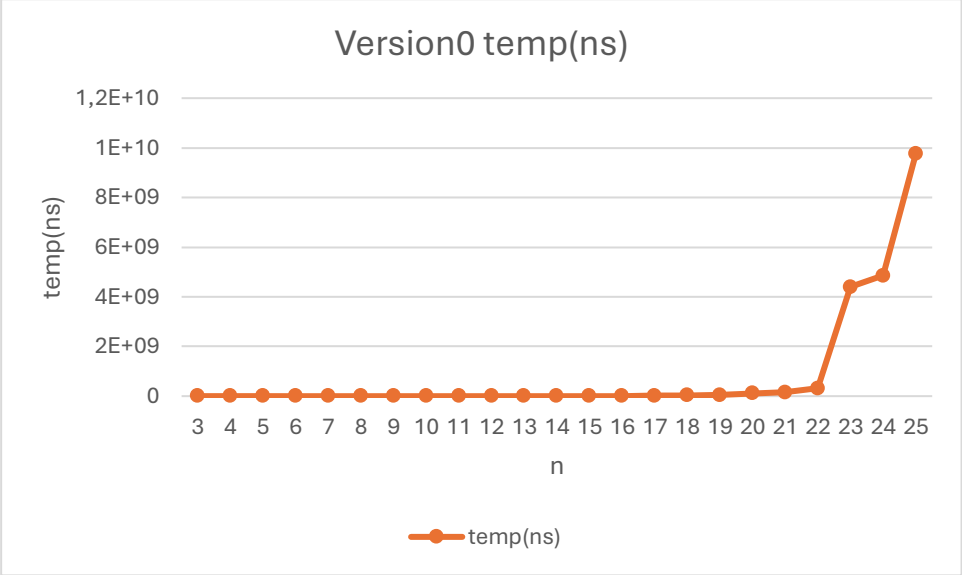
SAE S1.02 – Comparaison d'approches algorithmiques

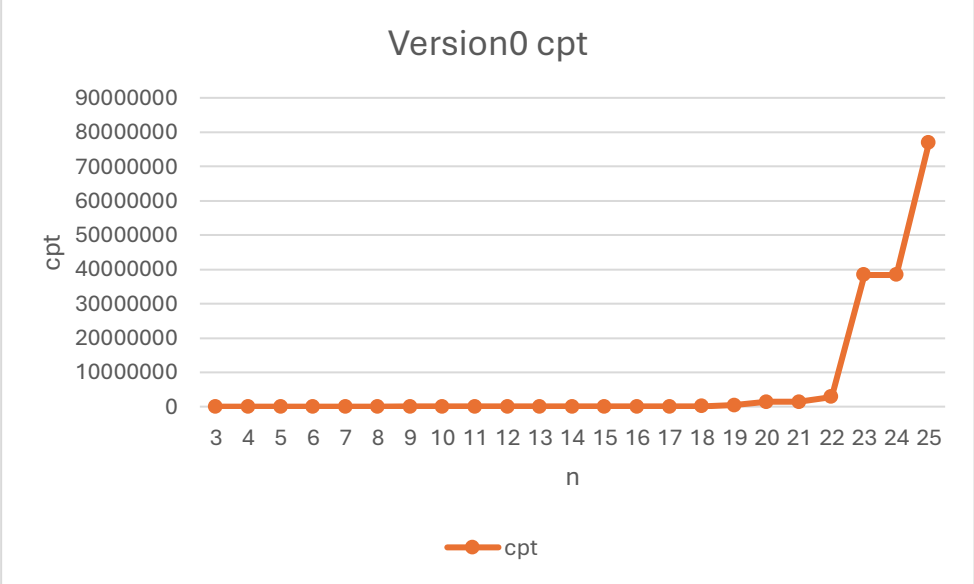
Objectif :

Mise en œuvre de différents algorithmes pour résoudre un même problème.
Par comparaison d'approches algorithmiques distinctes

Giovani Audebert Yohan Monnot

Version 0 : JeuGrundyRecBruteEff



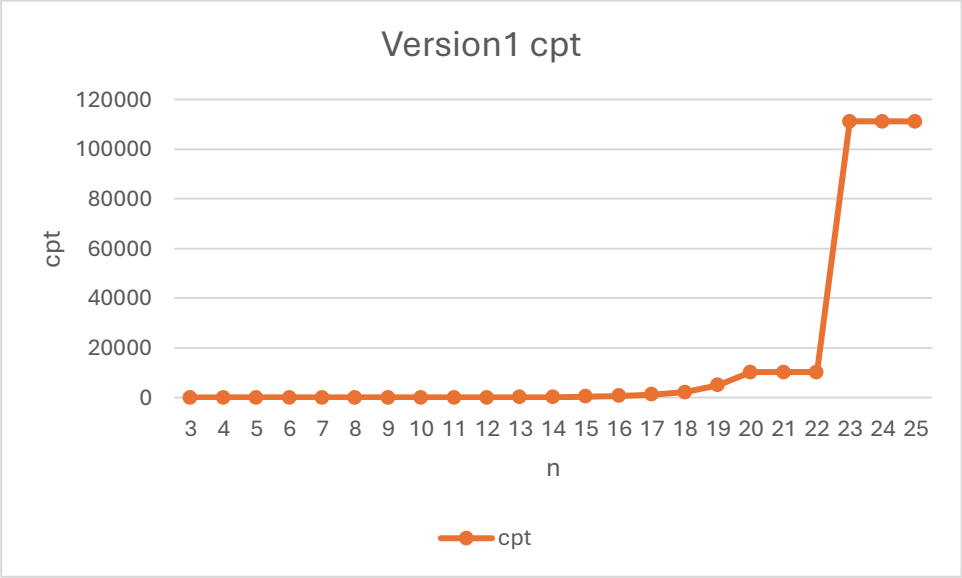


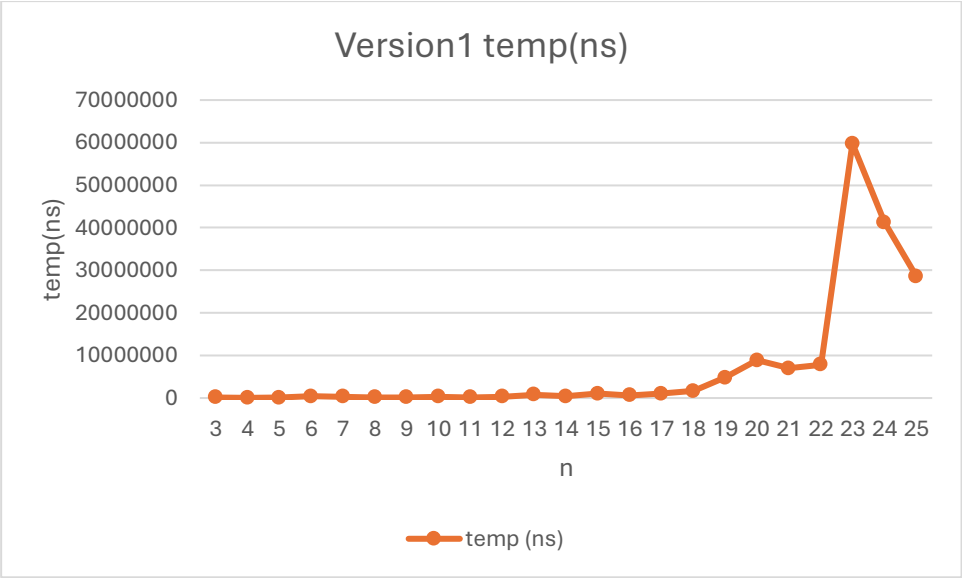
Version 0		
n	cpt	Temp(ns)
3	2	5600
4	3	9000
5	4	7900
6	8	35500
7	19	32600
8	20	22200
9	40	37100
10	113	136100
11	114	140300
12	228	168300
13	1268	678800

14	2560	794300
15	6528	1940000
16	20142	6612800
17	62802	23371500
18	128102	35688800
19	422886	47151300
20	1433263	114630000
21	1433264	155208800
22	2866528	318042800
23	38440187	4390063600
24	38440188	4848917300
25	76880376	9762430700

L'approche par force brute constitue une base fondamentale, mais elle souffre d'une croissance exponentielle du temps d'exécution à mesure que n augmente, la rendant impraticable pour de grandes valeurs.

Version1 : JeuGrundyRecPerdantes



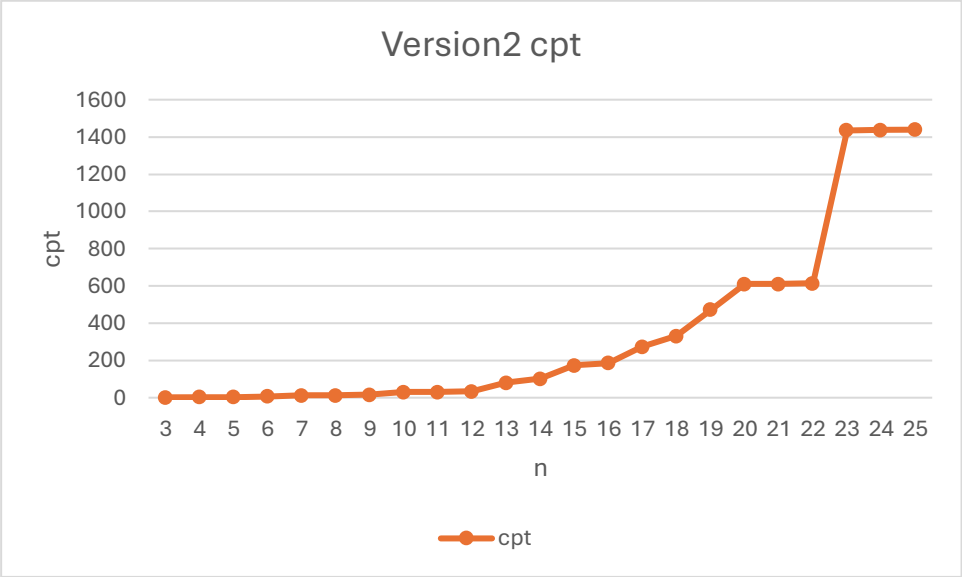


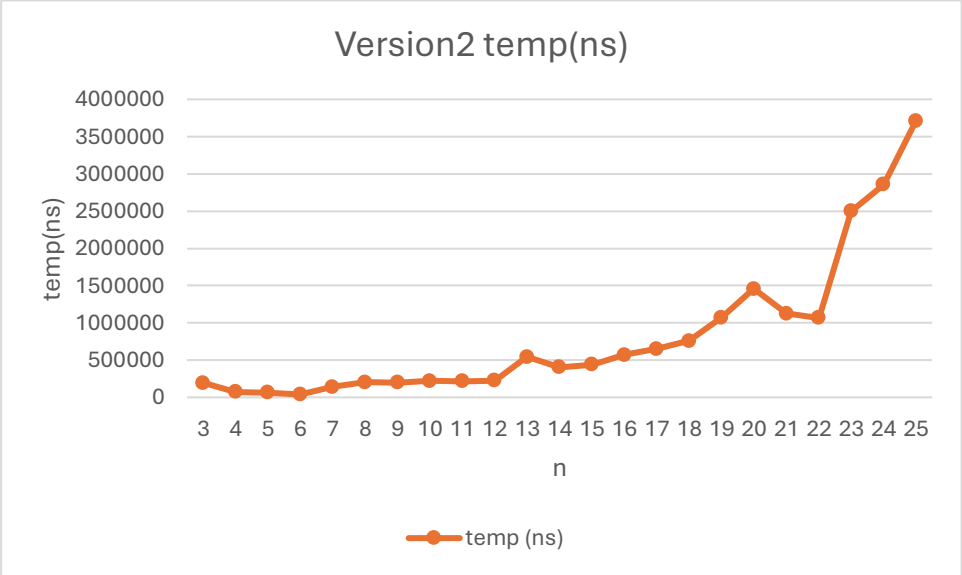
Version 1		
n	cpt	Temp(ns)
3	2	135600
4	3	46100
5	4	79400
6	6	382800
7	15	284100
8	16	127700
9	18	157600
10	41	338300
11	42	153500
12	44	317100
13	144	812700
14	188	390200
15	400	1052100
16	674	639400
17	1222	1058700
18	2140	1663300
19	4998	4749600
20	10177	8849200
21	10178	6944300
22	10180	7852300
23	111213	59802400
24	111214	41255100
25	111216	28569700

Nous constatons que la version1 a moins besoin d'utilisé la boucle car elle a un compteur « cpt » plus faible que la version0. La version1 met aussi moins de temps à n =23 que la version0 pour exécuter le programme.

L'introduction d'un cache pour les positions perdantes réduit significativement les calculs redondants, entraînant des gains de performance notables.

Version2 : JeuGrundyRecPerdEtGagn





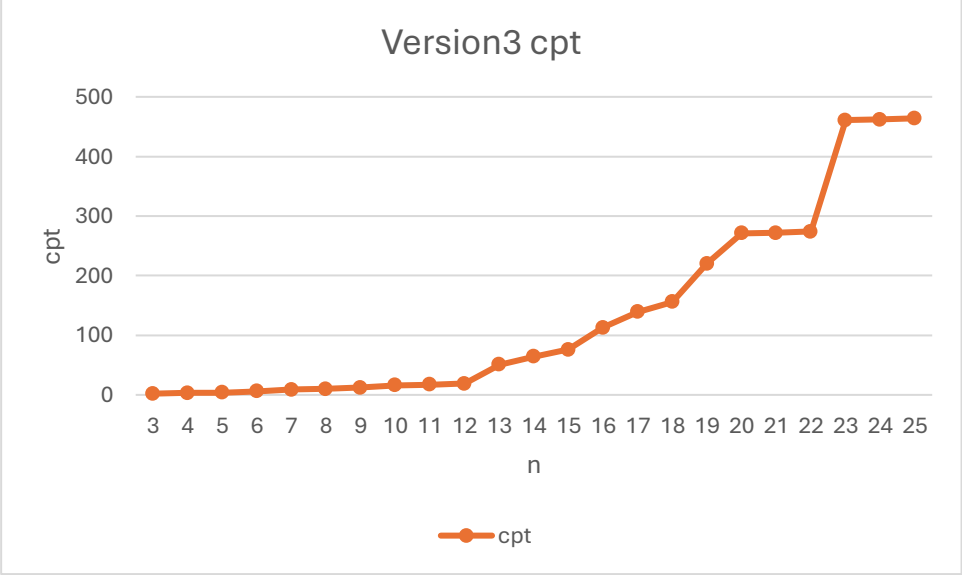
Version 2		
n	cpt	Temp(ns)
3	2	196100
4	3	75100
5	4	66600
6	6	39400
7	12	142300
8	13	205200
9	15	197100
10	30	220200
11	31	217900

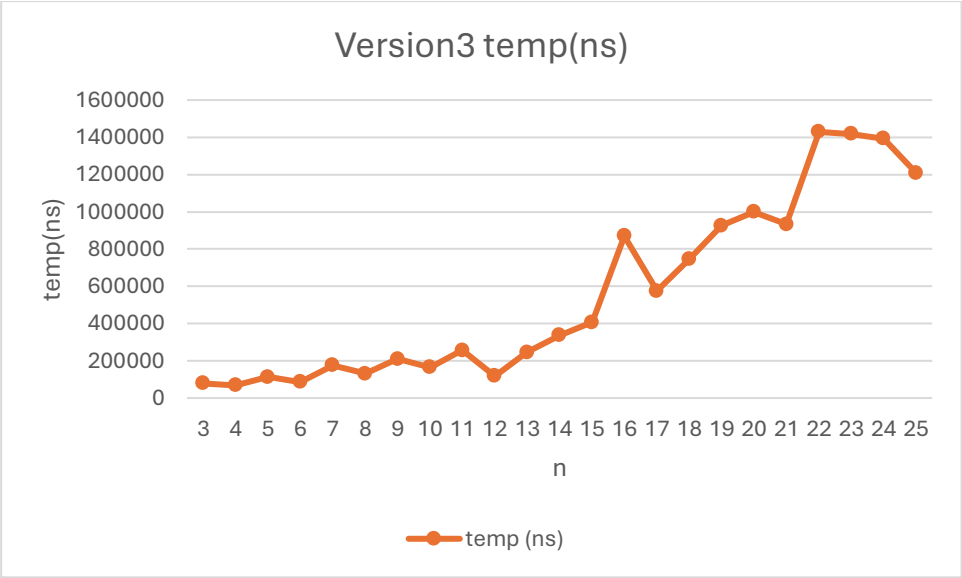
12	33	223600
13	80	543600
14	102	402900
15	174	438200
16	186	571300
17	273	649300
18	331	759500
19	473	1070600
20	610	1456800
21	611	1124600
22	613	1068400
23	1436	2501200
24	1437	2857300
25	1439	3711900

Nous constatons que la version2 a un plus faible compteur « cpt » que la version1 ce qui signifie qu'il y a eu moins d'appel récursif, de plus la version2 met aussi moins de temps que la version1 à exécuter le programme.

La mémorisation des positions perdantes et gagnantes diminue encore davantage les calculs, améliorant l'efficacité par rapport à la version 1.

Version3 : Jeu GrundyRecPerdanteNeutre



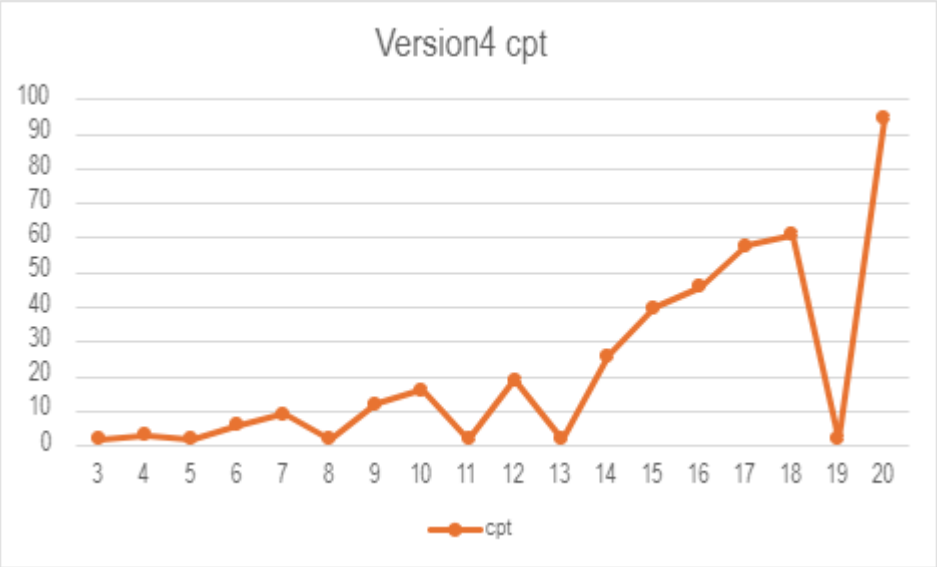


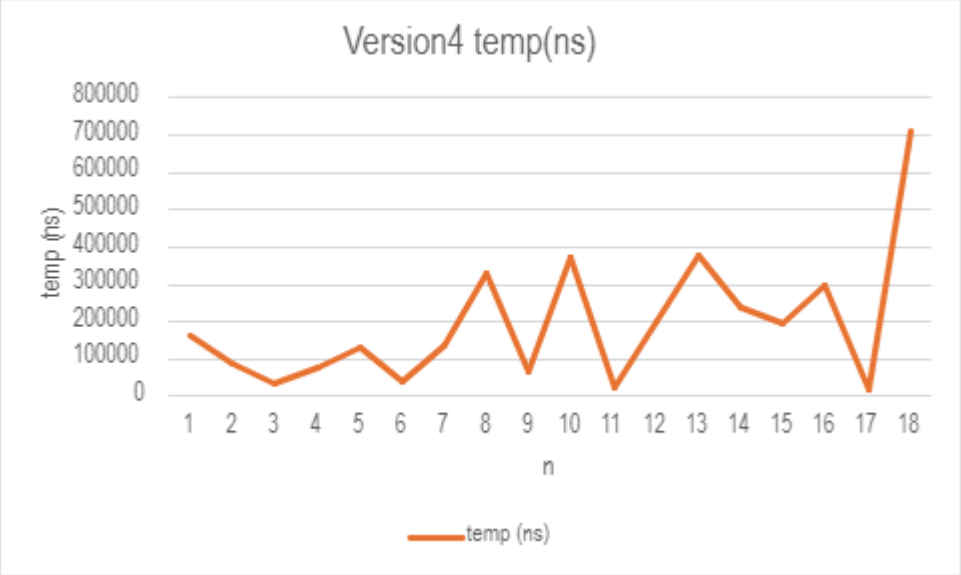
Version 3		
n	cpt	Temp(ns)
3	2	78400
4	3	67900
5	4	112501
6	6	86400
7	9	176101
8	10	130300
9	12	210401
10	16	165801
11	17	256200
12	19	118501
13	51	244900
14	64	335900
15	76	406300
16	113	870999
17	139	575399
18	156	745001
19	220	926101
20	271	998500
21	272	933500
22	274	1428799
23	461	1418401
24	462	1393600
25	464	1208899

Nous constatons que la version3 a un plus faible compteur « cpt » par rapport à la version2 ce qui veut dire que la méthode « estPerdante » fait moins d'appel récursif que la version2. Il y a aussi le fait que la version3 met moins de temps que la version1 à exécuter le programme.

L'application de la règle de suppression des tas neutres optimise encore l'algorithme, réduisant la complexité en éliminant les évaluations inutiles.

Version4 : Jeu GrundyRecGplusGequalsP





Version 4		
n	cpt	temp (ns)
3	2	162600
4	3	89900
5	2	33200
6	6	75700
7	9	130000

8	2	39000
9	12	137500
10	16	328400
11	2	65100
12	19	372800
13	2	25100
14	26	199600
15	40	378400
16	46	237400
17	58	197300
18	61	296300
19	2	16100
20	95	710700

Nous constatons que la version4 a un plus faible compteur « cpt » que la version3 ce qui signifie qu'il y a eu moins d'appel récursif, de plus la version4 met aussi moins de temps que la version3 à exécuter le programme.

On peut observer que parfois le cpt descend à 2. Ceci est dû au fait que l'IA reconnait directement les situations perdantes et donc fait moins d'appels.

L'utilisation de la théorie avancée des types simplifie les scénarios impliquant plusieurs tas gagnants, atteignant les meilleures performances en évitant totalement les opérations redondantes.

Conclusion :

À travers le développement successif des algorithmes du jeu de Grundy des versions 0 à 4, nous avons observé des améliorations significatives de l'efficacité. La version 0, bien que fonctionnelle, a montré une mise en pratique limitée en raison de son approche par force brute.

La version 1 a introduit la mise en cache des positions perdantes, réduisant ainsi les calculs redondants.

La version 2 a étendu cette optimisation en ajoutant la mise en cache des positions gagnantes, renforçant encore l'efficacité.

La version 3 a exploité le théorème de neutralisation, simplifiant les calculs en supprimant les tas perdants pendant l'évaluation.

Enfin, la version 4 a intégré des simplifications basées sur la théorie des types, maximisant les performances en s'appuyant sur des équivalences logiques.

Chaque amélioration a mis en évidence le compromis entre la complexité de mise en œuvre et l'efficacité en termes de temps d'exécution, la version 4 offrant le meilleur équilibre entre rapidité et évolutivité.