université de BORDEAUX

Collège Sciences et Technologies
UF Mathématiques et Interactions

Projet

Kea Horvath

M2 ROAD

2025 – 2026

Integer Programming

Table des matières

1 Structure du code

Le dossier include comprend les fichiers hpp utiles à l'exécution du code.

Le dossier src comprend l'implémentation des structures et méthodes de bases du problème (Instance.cpp et Solution.cpp) ainsi que l'implémentation des différents modèles mis en place.

A la racine du projet, il y a 4 fichiers cpp qui crée des éxecutables, ainsi que leurs éxecutables correspondant. Afin de s'en servir pour résoudre une instance, vous pouvez éxecuter le programme sans paramètres et son mode d'emploi s'affichera.

2 Travail préalable

Lors de mes expérimentations, j'ai observé que certaines instances n'avaient pas de solution réalisable. J'ai donc décidé d'ignorer ces instances en vérifiant la validité d'une instance avant de lancer quelconque solver dessus.

Je réalise le test de la manière suivante : je calcule d'un côté la somme des demandes de tous les clients et de l'autre la somme des capacités des p facilities ayant les plus grandes capacités. Si la capacité calculée (qui est donc la plus grande capacité possible pour les facilities qui vont être activées) est plus petite que la demande totale, alors on sait que le problème n'aura pas de solution réalisable.

J'ai ainsi trouvé que les instances uniform_26, uniform_27 et uniform_31 étaient irréalisables et ne sont donc pas prises en compte lors de mes expérimentations.

3 Modèle PLNE Compacte

Comme demandé, j'ai d'abord implémenté un modèle en nombre entiers compact.

On pose x_{fc} qui vaut 1 si le client $c \in C$ est servi par la facility $f \in F$ et 0 sinon. On note y_f qui vaut 1 si la facility $f \in F$ est ouverte et 0 sinon.

Un programme linéaire en nombres entiers pour ce problème peut être écrit de la manière suivante :

$$\min \sum_{f \in F} \sum_{c \in C} c_{fc} x_{fc} \tag{1}$$

$$sc \sum_{f \in F} x_{fc} = 1 \qquad \forall c \in C \tag{2}$$

$$\sum_{f \in F} y_f \leqslant p \tag{3}$$

$$\sum_{c \in C} d_c x_{fc} \leqslant u_f y_f \qquad \forall f \in F \tag{4}$$

$$x_{fc} \in \{0,1\} \qquad \forall f \in F, c \in C \tag{5}$$

$$y_f \in \{0,1\} \qquad \forall f \in F \tag{6}$$

L'objectif (1) consiste à minimiser la distance totale à parcourir pour servir tous les clients. Les contraintes (2) assure que chaque client soit servi. La contrainte (3) vérifie qu'on n'ouvre pas plus de facilities qu'autorisé. Les contraintes (4) assurent que chaque facility ne serve pas plus que sa capacité. Les contraintes (5) et (6) définissent le domaine des variables de décision.

J'ai ensuite effectué des expérimentations avec ce modèle compact basique. Les résultats peuvent être vus en détails dans la table 1. Pour tous les tests réalisés lors de ce rapport, j'ai fixé le temps limite à 60s. Cette limite est assez courte, mais j'ai trouvé cela plus pratique afin de pouvoir réaliser plus de tests.

Sur les 44 instances testées, ce modèle MIP arrive à trouver une solution optimale pour 13 instances.

On observe également que la relaxation linéaire n'est pas de très bonne qualité pour cette formulation : sur les instances données, on obtient un gap (entre la meilleure solution trouvée et la valeur de la relaxation) de 30.93% en moyenne, ce qui est très élevé !

4 Génération de colonnes basique

4.1 Relaxed Master Problem

On va maintenant essayer de résoudre le problème relâché avec une approche de génération de colonnes. Chaque colonne $g \in G$ va représenter une facility $f_g \in F$ et un ensemble de clients $S_g \subseteq C$ qui lui sont assignés. Pour écrire le problème maître, on aura besoin d'une variable α_g qui indique le poids de la colonne $g \in G$ dans la combinaison convexe. La donnée c_g indique le coût de la colonne $g \in G$. On pourra la déduire facilement à partir des distances entre les clients de S_g et f_g.

$$\min \sum_{g \in G} c_g \alpha_g \tag{7}$$

$$sc \sum_{g \in G : c \in S_g} \alpha_p = 1 \qquad \forall c \in C \tag{8}$$

$$\sum_{g \in G} \alpha_g \leqslant p \tag{9}$$

$$\alpha_g \geqslant 0 \qquad \forall g \in G \tag{10}$$

L'objectif (7) consiste encore une fois à minimiser la distance totale à parcourir pour servir tous les clients. Les contraintes (8) assurent que chaque client soit servi. En effet, on vérifie que la somme des coefficients des colonnes où le client c apparaît soit égale à 1. La contrainte (9) vérifie qu'on n'ait pas plus de p colonnes pour couvrir tous les clients. Les contraintes (10) définissent le domaine des variables de décision.

4.2 Première solution valide

Pour pouvoir faire fonctionner notre algorithme de génération de colonnes, nous avons besoin d'une solution réalisable initiale (pour calculer des valeurs de variables duales à la première itération). J'ai donc implémenté une heuristique très basique. On choisit les p facilities avec le plus de capacité puis chaque client est assigné à la première facility qui a encore de la place.

J'avais prévu d'implémenter une heuristique qui donnerait des solutions de meilleure qualité mais n'ai pas eu le temps.

4.3 Pricing Subproblem

On a également besoin de définir un sous-problème pricing qui va nous permettre de déterminer quelle colonne ajouter à notre problème maître à chaque étape. On note π_c les variables duales associées aux contraintes (8) et θ la variable duale associée à la contrainte (9).

J'ai décidé de créer un sous-problème de pricing par facility. Pour chaque $f \in F$ on cherche le sous-ensemble S_f qui nous donnera le plus petit coût réduit. On doit donc introduire une nouvelle variable z_c qui vaudra 1 si on place le client $c \in C$ dans le sous-ensemble S_g et 0 sinon. Autrement dit, on résout ce programme linéaire :

$$\min \ -\theta + \sum_{c \in C} z_c (c_{fc} - \pi_c) \tag{11}$$

$$sc \sum_{c \in C} d_c z_c \leqslant u_f \tag{12}$$

$$z_c \in \{0, 1\} \qquad \forall c \in C \tag{13}$$

L'objectif (11) permet de trouver le sous-ensemble de clients avec le plus petit coût réduit (on a ignoré la variable duale θ car c'est une constante). La contrainte (12) permet de vérifier qu'on ne dépasse pas la capacité de la facility f. Les contraintes (13) définissent le domaine des variables de décision.

On voit que ce modèle est très compacte : seulement $|C|$ variables et une contrainte. Le solveur MIP devrait donc rapidement trouver une solution. Cependant, en choisissant ces formulations du

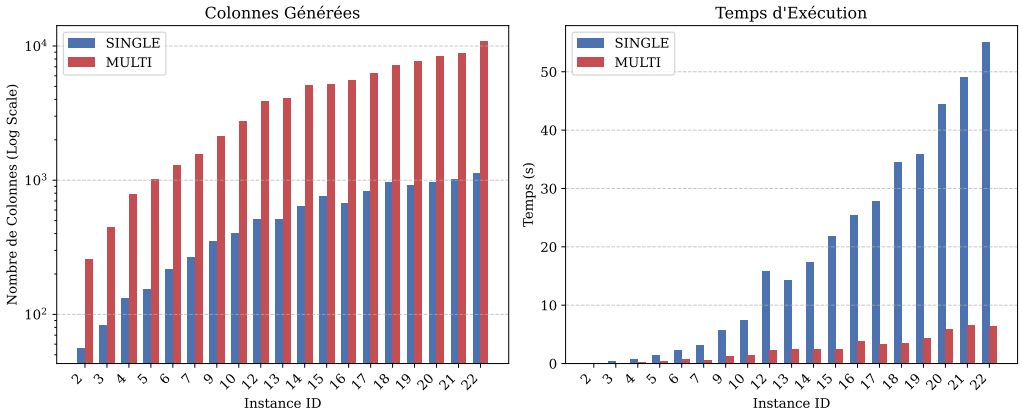

FIGURE 1 – Comparaison entre les méthodes SINGLE et MULTI

pricing, on va devoir résoudre ce modèle pour chaque facility $f \in F$ à chaque étape. On devra alors soit comparer les résultats pour n'ajouter que la meilleure colonne, soit en ajouter plusieurs, soit toutes les ajouter à notre problème maître : les expérimentations me permettront de décider quelle est la meilleure approche.

Avant de passer à la suite, je vais donc réaliser des expérimentations pour voir quelle formulation du pricing est la plus intéressante. Je note SINGLE la méthode qui consiste à n'ajouter que la meilleure colonne parmi toutes les colonnes intéressantes (i.e le coût réduit est négatif) trouvées (une pour chaque facility) et MULTI la méthode qui consiste à ajouter toutes les colonnes intéressantes trouvées.

Le détail des résultats de ces expérimentations sont visibles dans la table 2. Les graphes de la figure 1 présentent une comparaison des résultats obtenus entre les deux méthodes pour les instances où les deux méthodes ont trouvé une solution optimale dans le temps imparti. On peut voir que bien qu'avec la méthode MULTI, on ajoute beaucoup plus de colonnes avant de converger vers la solution optimale (environ 2 fois plus en moyenne), on arrive à cette solution bien plus rapidement (7 fois plus vite environ en moyenne). On note aussi que la version SINGLE n'arrive à trouver la solution optimale en moins de 60 secondes que pour 19 instances sur 44, alors que sur les 44 instances, la version MULTI met au maximum 52 secondes. Dans le reste de mes expérimentations, je vais donc utiliser la méthode MULTI, qui est bien meilleure, pour l'ajout des colonnes à chaque itération.

4.4 Comparaison avec la relaxation du modèle compact

Notre problème est un problème de minimisation. Notre but est donc de trouver une relaxation la plus grande possible, qui se rapproche le plus possible de la vraie solution optimale (en nombres entiers). On a vu que la relaxation du modèle compact n'était vraiment pas bonne : gap de 30.93% en moyenne avec la meilleure solution trouvée. En revanche, le gap moyen entre la meilleure solution trouvée par le modèle compact et la valeur obtenue par la génération de colonnes est de 9.94%, ce qui prouve que la relaxation obtenue par la génération de colonnes est bien meilleure.

5 Améliorations

5.1 Pricing avec programmation dynamique

Afin d'accélérer la résolution de l'algorithme de génération de colonnes, j'ai ensuite implémenté une autre méthode pour résoudre les sous problèmes pricing : la programmation dynamique. On peut voir le détail des résultats obtenus dans la table 3. On remarque tout d'abord que l'on obtient exactement les mêmes valeurs pour les deux méthodes, ce qui est rassurant puisque que c'est ce qui était attendu.

On voit aussi, dans la figure 2 que la programmation dynamique est plus rapide que l'utilisation d'un MIP pour résoudre les sous-problèmes de pricing.

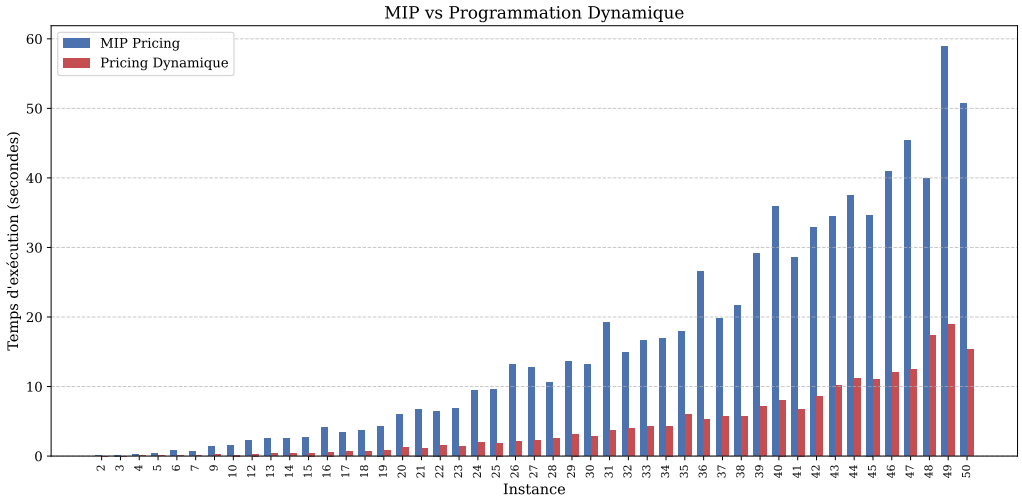

FIGURE 2 – Comparaison entre les méthodes MIP et DP

5.2 Stabilisation

Pour encore accélérer davantage la convergence de la génération de colonnes, j'ai ensuite mis en place un système de stabilisation avec la méthode de séparation in out décrite dans le cours.

J'ai remarqué un petit problème une fois avoir fini l'implémentation :les valeurs obtenues avec et sans stabilisation avaient parfois quelques millièmes d'écart alors qu'elles devraient être identiques. J'ai donc ajouté une dernière étape à la stabilsation. Une fois la séparation in-out finie, on ajoute des colonnes en faisant un pricing normal jusqu'à ce qu'on en ait plus à ajouter. On est ainsi sûrs que la solution obtenue est bien valide.

On peut voir dans la table 4 que cette modification accélère encore la résolution.

6 Recherche d'une solution entière

Afin de trouver une solution entière à partir du résultat obtenu avec la génération de colonnes, j'ai implémenté une heuristique diving. Celle-ci fonctionne de la manière suivante :

- Je résous une première fois la génération de colonnes de base.
- Je calcule les valeurs théoriques de $x[f][c]$ pour chaque f et c en convertissant mes variables λ en variables x.
- Je sélectionne la paire f-c avec la plus grande valeur fractionnaire.
- Je nettoie ensuite le master en supprimant les colonnes incompatibles
  - les colonnes représentant f mais sans le client c
  - les colonnes ne représentant pas f mais avec le client c
- Je résous le master mis à jour
- J'ajoute les meilleures colonnes compatibles

6.1 Comparaison avec le modèle compact

Les détails des résultats obtenus pour les instances données sont dans la table 5. Le gap moyen avec la borne inférieure obtenue par génération de colonnes est de 9.94% pour le modèle compact et de 16.84% pour l'heuristique diving. En revanche, on voit que l'heuristique est beaucoup plus rapide : son temps moyen de résolution pour les instances données est de 10 secondes, alors que le modèle compact ne trouve une solution optimale que pour 13 instances. Pour bien comparer les deux méthodes, il aurait fallu voir la valeur de la meilleure solution obtenue par le modèle compact dans la même durée que l'heuristique trouve son résultat, mais je n'ai pas eu le temps.

A noter que cet algorithme trouve une solution invalide pour l'instance 40. Je n'ai pas eu le temps de regarder pourquoi.

J'ai créé tous les fichiers svg et sol pour toutes les solutions trouvées à l'aide de cette heuristique. Ils sont visibles dans le dossier solutions de mon code.

Je n'ai pas eu le temps d'attaquer à l'algorithme de Branch and Price.

7 Annexe

Vous trouverez dans cette section les tableaux avec tous les résultats obtenus lors de mes expérimentations. Le temps limite a été fixé à 60 secondes pour tous les tests. L'acronyme TLR signifie Time Limit Reached.

TABLE 1 – Résultats numériques du modèle compact (en 60s)

Instance	Opt ?	Best Sol	Dual Bound	Gap	Duration(s)	Relax Sol	Relax Gap	Duration(s)
uniform_2	YES	3.2609	3.2609	0.00%	0.0101	2.2194	31.94%	0.0006
uniform_3	YES	4.4126	4.4126	0.00%	0.0414	2.5117	43.08%	0.0005
uniform_4	YES	4.7908	4.7908	0.00%	0.1121	3.2288	32.60%	0.0007
uniform_5	YES	6.1633	6.1633	0.00%	0.2455	3.6370	40.99%	0.0010
uniform_6	YES	7.8542	7.8542	0.00%	0.2715	6.1891	21.20%	0.0013
uniform_7	YES	7.2308	7.2308	0.00%	0.4464	4.7331	34.54%	0.0016
uniform_9	YES	9.7517	9.7517	0.00%	0.6245	7.9648	18.32%	0.0034
uniform_10	YES	8.8464	8.8464	0.00%	4.8267	5.9186	33.10%	0.0057
uniform_12	YES	11.0556	11.0556	0.00%	15.7420	8.3713	24.28%	0.0088
uniform_13	YES	10.6065	10.6065	0.00%	35.4666	7.7471	26.96%	0.0089
uniform_14	YES	9.6509	9.6509	0.00%	19.6302	5.7797	40.11%	0.0150
uniform_15	YES	9.7158	9.7158	0.00%	30.8370	6.3091	35.06%	0.0186
uniform_16	NO	15.6976	14.7477	6.05%	60.0219	12.7526	18.76%	0.0266
uniform_17	NO	10.6740	10.5057	1.58%	60.0082	6.7991	36.30%	0.0224
uniform_18	YES	10.5733	10.5733	0.00%	57.1451	5.8438	44.73%	0.0251
uniform_19	NO	11.9861	11.7723	1.78%	60.0283	8.3581	30.27%	0.0225
uniform_20	NO	14.5556	14.2809	1.89%	60.0164	11.6217	20.16%	0.0237
uniform_21	NO	13.0889	12.6954	3.01%	60.0021	9.7339	25.63%	0.0327
uniform_22	NO	13.1997	12.2009	7.57%	60.0114	8.6707	34.31%	0.0571
uniform_23	NO	11.4247	11.3074	1.03%	60.0428	7.6734	32.84%	0.0565
uniform_24	NO	14.2112	13.8244	2.72%	60.0159	11.0286	22.39%	0.0398
uniform_25	NO	14.6449	13.7503	6.11%	60.0051	10.1600	30.62%	0.0464
uniform_28	NO	14.5976	13.8782	4.93%	60.0222	10.1652	30.36%	0.0606
uniform_29	NO	21.9684	18.3480	16.48%	60.0225	16.0705	26.85%	0.0854
uniform_30	NO	14.9128	14.3597	3.71%	60.0143	10.3920	30.32%	0.0715
uniform_32	NO	16.3292	15.0619	7.76%	60.0318	11.2045	31.38%	0.0673
uniform_33	NO	14.3106	13.7607	3.84%	60.0202	9.1582	36.00%	0.0571
uniform_34	NO	15.6106	14.7247	5.67%	60.0194	10.4791	32.87%	0.0800
uniform_35	NO	13.8968	13.5950	2.17%	60.0198	8.7598	36.97%	0.0999
uniform_36	NO	21.3013	18.2111	14.51%	60.0553	15.3671	27.86%	0.1444
uniform_37	NO	15.7605	14.8650	5.68%	60.1752	9.8530	37.48%	0.0686
uniform_38	NO	17.1421	16.1655	5.70%	60.0206	11.5985	32.34%	0.0815
uniform_39	NO	20.9242	18.6000	11.11%	60.0140	15.5503	25.68%	0.1031
uniform_40	NO	27.0132	24.0071	11.13%	60.3406	21.9928	18.58%	0.3442
uniform_41	NO	20.8250	18.0423	13.36%	60.6482	15.1976	27.02%	0.1195
uniform_42	NO	23.5641	20.2584	14.03%	60.0125	17.4000	26.16%	0.2192
uniform_43	NO	22.0848	18.2954	17.16%	60.0473	14.5128	34.29%	0.1870
uniform_44	NO	26.3959	22.6307	14.26%	60.0203	19.7475	25.19%	0.2870
uniform_45	NO	19.1757	17.6605	7.90%	60.0573	13.7529	28.28%	0.2968
uniform_46	NO	25.5870	20.6160	19.43%	60.0454	17.6394	31.06%	0.2578
uniform_47	NO	27.8387	21.6932	22.08%	60.0536	18.9699	31.86%	0.2585
uniform_48	NO	19.3624	17.0020	12.19%	60.0112	12.3814	36.05%	0.2405
uniform_49	NO	19.7386	16.7673	15.05%	60.0283	12.0474	38.97%	0.3701
uniform_50	NO	19.9934	18.1077	9.43%	60.0233	13.1259	34.35%	0.2183

TABLE 2 – Comparaison des méthodes SINGLE et MULTI pour l'ajout des colonnes (60s)

Instance	SINGLE Value	Nb cols	Durations(s)	MULTI Value	Nb cols	Duration(s)
uniform_2	3.2609	56	0.15	3.2609	255	0.07
uniform_3	4.4126	83	0.39	4.4126	446	0.14
uniform_4	4.5890	131	0.84	4.5890	784	0.28
uniform_5	5.6891	154	1.44	5.6891	1013	0.45
uniform_6	7.6812	216	2.25	7.6812	1284	0.79
uniform_7	7.0258	265	3.24	7.0258	1550	0.64
uniform_9	8.9866	347	5.75	8.9866	2118	1.35
uniform_10	8.4609	400	7.53	8.4609	2756	1.45
uniform_12	10.3767	509	15.81	10.3767	3835	2.38
uniform_13	10.0520	514	14.35	10.0520	4095	2.47
uniform_14	9.3762	641	17.40	9.3762	5124	2.40
uniform_15	9.4575	756	21.87	9.4575	5175	2.55
uniform_16	13.7697	673	25.45	13.7697	5583	3.92
uniform_17	10.3786	823	27.84	10.3786	6281	3.28
uniform_18	10.3500	957	34.57	10.3500	7179	3.51
uniform_19	11.4993	917	35.98	11.4993	7692	4.33
uniform_20	12.9949	962	44.48	12.9949	8326	5.95
uniform_21	12.3859	1006	49.11	12.3859	8774	6.60
uniform_22	12.1359	1117	55.15	12.1359	10902	6.42
uniform_23	11.1971(TLR)	1134	60.00	11.1252	10629	6.61
uniform_24	14.2595(TLR)	1043	60.04	13.6443	11607	8.41
uniform_25	15.0131(TLR)	986	60.06	13.7025	12329	8.59
uniform_28	20.4982(TLR)	836	60.03	13.7468	15905	10.49
uniform_29	26.5025(TLR)	757	60.07	16.4404	16203	13.82
uniform_30	35.6422(TLR)	712	60.01	14.2201	18196	14.32
uniform_32	51.6215(TLR)	615	60.02	14.7096	18965	14.50
uniform_33	71.8679(TLR)	520	60.00	13.6904	21316	22.09
uniform_34	80.1438(TLR)	457	60.06	14.6342	22622	22.04
uniform_35	118.2396(TLR)	448	60.05	13.5831	23211	21.85
uniform_36	117.1736(TLR)	417	60.04	17.3402	26340	33.52
uniform_37	113.9680(TLR)	323	60.10	14.8283	23064	22.99
uniform_38	119.2056(TLR)	379	60.05	16.2680	24894	25.60
uniform_39	116.6482(TLR)	388	60.07	18.4140	27740	32.90
uniform_40	125.0635(TLR)	369	60.07	19.9123	31948	43.69
uniform_41	124.9377(TLR)	341	60.07	18.0479	27460	35.44
uniform_42	130.1364(TLR)	327	60.06	19.6141	31694	44.22
uniform_43	131.6941(TLR)	374	60.06	18.1065	33034	37.58
uniform_44	129.4426(TLR)	373	60.07	20.4505	30628	39.71
uniform_45	140.0112(TLR)	359	60.04	17.5671	35815	33.97
uniform_46	145.6087(TLR)	380	60.15	19.7402	35253	38.86
uniform_47	147.6986(TLR)	364	60.12	20.1386	36121	44.78
uniform_48	153.0965(TLR)	345	60.10	17.2657	37366	45.63
uniform_49	163.3222(TLR)	270	60.01	17.0555	43593	52.09
uniform_50	159.9087(TLR)	299	60.08	18.2410	40786	43.95

TABLE 3 – Comparaison entre pricing MIP et pricing dynamique

Instance	MIP Value	Nb cols	Durations(s)	DP Value	Nb cols	Duration(s)
uniform_2	3.2609	255	0.07	3.2609	266	0.01
uniform_3	4.4126	446	0.16	4.4126	452	0.02
uniform_4	4.5890	784	0.26	4.5890	790	0.03
uniform_5	5.6891	1013	0.39	5.6891	1009	0.04
uniform_6	7.6812	1284	0.79	7.6812	1268	0.07
uniform_7	7.0258	1550	0.66	7.0258	1563	0.07
uniform_9	8.9866	2118	1.36	8.9866	2207	0.17
uniform_10	8.4609	2756	1.47	8.4609	2750	0.16
uniform_12	10.3767	3835	2.18	10.3767	3748	0.30
uniform_13	10.0520	4095	2.58	10.0520	3899	0.33
uniform_14	9.3762	5124	2.52	9.3762	4606	0.39
uniform_15	9.4575	5175	2.65	9.4575	5205	0.45
uniform_16	13.7697	5583	4.11	13.7697	5351	0.58
uniform_17	10.3786	6281	3.40	10.3786	6532	0.64
uniform_18	10.3500	7179	3.65	10.3500	6902	0.66
uniform_19	11.4993	7692	4.33	11.4993	7519	0.81
uniform_20	12.9949	8326	5.97	12.9949	9254	1.29
uniform_21	12.3859	8774	6.76	12.3859	8536	1.13
uniform_22	12.1359	10902	6.48	12.1359	10822	1.47
uniform_23	11.1252	10629	6.90	11.1252	10663	1.39
uniform_24	13.6443	11607	9.48	13.6443	12783	1.98
uniform_25	13.7025	12329	9.55	13.7025	12808	1.89
uniform_28	13.7468	15905	10.62	13.7468	15602	2.47
uniform_29	16.4404	16203	13.59	16.4404	16279	3.17
uniform_30	14.2201	18196	13.22	14.2201	16345	2.79
uniform_32	14.7096	18965	14.94	14.7096	19911	3.96
uniform_33	13.6904	21316	16.64	13.6904	21959	4.26
uniform_34	14.6342	22622	16.95	14.6342	23371	4.33
uniform_35	13.5831	23211	17.95	13.5831	24380	6.00
uniform_36	17.3402	26340	26.54	17.3402	21765	5.23
uniform_37	14.8283	23064	19.77	14.8283	23911	5.66
uniform_38	16.2680	24894	21.69	16.2680	25265	5.71
uniform_39	18.4140	27740	29.16	18.4140	26536	7.16
uniform_40	19.9123	31948	35.83	19.9123	30405	8.07
uniform_41	18.0479	27460	28.51	18.0479	24300	6.76
uniform_42	19.6141	31694	32.87	19.6141	32467	8.58
uniform_43	18.1065	33034	34.52	18.1065	32930	10.09
uniform_44	20.4505	30628	37.51	20.4505	39159	11.13
uniform_45	17.5671	35815	34.66	17.5671	37114	10.96
uniform_46	19.7402	35253	40.95	19.7402	36837	12.05
uniform_47	20.1386	36121	45.34	20.1386	38033	12.51
uniform_48	17.2657	37366	39.88	17.2657	39005	17.35
uniform_49	17.0555	43593	58.94	17.0555	40459	18.91
uniform_50	18.2410	40786	50.68	18.2410	42951	15.35

TABLE 4 – Comparaison sans et avec separation in out

Instance	NOSTAB Value	Nb cols	Durations(s)	INOUT Value	Nb cols	Duration(s)
uniform_2	3.2609	266	0.01	3.2609	221	0.01
uniform_3	4.4126	452	0.02	4.4126	356	0.01
uniform_4	4.5890	790	0.03	4.5890	592	0.02
uniform_5	5.6891	1009	0.03	5.6891	852	0.03
uniform_6	7.6812	1268	0.06	7.6812	1019	0.05
uniform_7	7.0258	1563	0.07	7.0258	1257	0.05
uniform_9	8.9866	2207	0.14	8.9866	1631	0.10
uniform_10	8.4609	2750	0.16	8.4609	1997	0.11
uniform_12	10.3767	3748	0.29	10.3767	2927	0.23
uniform_13	10.0520	3899	0.30	10.0520	2984	0.24
uniform_14	9.3762	4606	0.37	9.3762	3431	0.25
uniform_15	9.4575	5205	0.46	9.4575	3781	0.32
uniform_16	13.7697	5351	0.50	13.7697	3847	0.36
uniform_17	10.3786	6532	0.59	10.3786	4878	0.41
uniform_18	10.3500	6902	0.61	10.3500	5451	0.47
uniform_19	11.4993	7519	0.80	11.4993	5265	0.53
uniform_20	12.9949	9254	1.14	12.9949	6044	0.68
uniform_21	12.3859	8536	1.06	12.3859	5483	0.67
uniform_22	12.1359	10822	1.21	12.1359	7728	0.84
uniform_23	11.1252	10663	1.26	11.1252	7466	0.86
uniform_24	13.6443	12783	1.78	13.6443	9114	1.17
uniform_25	13.7025	12808	1.84	13.7025	8355	1.11
uniform_28	13.7468	15602	2.29	13.7468	9683	1.47
uniform_29	16.4404	16279	2.72	16.4404	10335	1.71
uniform_30	14.2201	16345	2.61	14.2201	10461	1.61
uniform_32	14.7096	19911	3.94	14.7096	11993	2.10
uniform_33	13.6904	21959	4.14	13.6904	13497	2.56
uniform_34	14.6342	23371	4.45	14.6342	16145	2.68
uniform_35	13.5831	24380	5.80	13.5831	16844	3.14
uniform_36	17.3402	21765	5.16	17.3402	16811	3.83
uniform_37	14.8283	23911	4.98	14.8283	17591	3.57
uniform_38	16.2680	25265	5.46	16.2680	20304	4.39
uniform_39	18.4140	26536	6.79	18.4140	16789	3.82
uniform_40	19.9123	30405	7.63	19.9123	20020	7.15
uniform_41	18.0479	24300	9.47	18.0479	17838	5.93
uniform_42	19.6141	32467	10.04	19.6141	17804	5.67
uniform_43	18.1065	32930	14.30	18.1065	21864	9.14
uniform_44	20.4505	39159	20.17	20.4505	22792	20.20
uniform_45	17.5671	37114	26.97	17.5671	24750	7.94
uniform_46	19.7402	36837	14.39	19.7402	24192	7.73
uniform_47	20.1386	38033	15.84	20.1386	21382	9.18
uniform_48	17.2657	39005	15.50	17.2657	25118	8.57
uniform_49	17.0555	40459	14.69	17.0555	25402	9.55
uniform_50	18.2410	42951	15.65	18.2410	25139	9.23

TABLE 5 – Comparaisons entre modèle compact et heuristique diving

Instance	LB	Comp BS	Gap	Dur(s)	Diving BS	Gap	Dur(s)
uniform_2	3.2609	3.2609	0.00%	0.01	3.2609	0.00%	0.01
uniform_3	4.4126	4.4126	0.00%	0.04	4.4126	0.00%	0.02
uniform_4	4.589	4.7908	4.40%	0.11	5.4856	19.54%	0.06
uniform_5	5.6891	6.1633	8.34%	0.25	7.1031	24.85%	0.09
uniform_6	7.6812	7.8542	2.25%	0.27	8.3575	8.80%	0.15
uniform_7	7.0258	7.2308	2.92%	0.45	7.6626	9.06%	0.15
uniform_9	8.9866	9.7517	8.51%	0.62	9.9047	10.22%	0.33
uniform_10	8.4609	8.8464	4.56%	4.83	9.9244	17.30%	0.48
uniform_12	10.3767	11.0556	6.54%	15.74	11.6648	12.41%	0.67
uniform_13	10.052	10.6065	5.52%	35.47	12.7451	26.79%	0.92
uniform_14	9.3762	9.6509	2.93%	19.63	10.4141	11.07%	0.79
uniform_15	9.4575	9.7158	2.73%	30.84	10.1386	7.20%	1.05
uniform_16	13.7697	15.6976	14.00%	60.02	17.0629	23.92%	1.36
uniform_17	10.3786	10.674	2.85%	60.01	11.2199	8.11%	1.56
uniform_18	10.35	10.5733	2.16%	57.15	11.1035	7.28%	1.56
uniform_19	11.4993	11.9861	4.23%	60.03	13.2417	15.15%	1.81
uniform_20	12.9949	14.5556	12.01%	60.02	16.8526	29.69%	2.69
uniform_21	12.3859	13.0889	5.68%	60.00	13.5194	9.15%	2.30
uniform_22	12.1359	13.1997	8.77%	60.01	13.5486	11.64%	2.75
uniform_23	11.1252	11.4247	2.69%	60.04	13.0673	17.46%	3.98
uniform_24	13.6443	14.2112	4.15%	60.02	16.0664	17.75%	4.98
uniform_25	13.7025	14.6449	6.88%	60.01	16.1731	18.03%	4.25
uniform_28	13.7468	14.5976	6.19%	60.02	17.3842	26.46%	6.88
uniform_29	16.4404	21.9684	33.62%	60.02	22.2765	35.50%	8.01
uniform_30	14.2201	14.9128	4.87%	60.01	18.9964	33.59%	8.12
uniform_32	14.7096	16.3292	11.01%	60.03	18.0776	22.90%	10.59
uniform_33	13.6904	14.3106	4.53%	60.02	16.3962	19.76%	10.69
uniform_34	14.6342	15.6106	6.67%	60.02	16.5595	13.16%	10.75
uniform_35	13.5831	13.8968	2.31%	60.02	14.282	5.15%	12.61
uniform_36	17.3402	21.3013	22.84%	60.06	22.0564	27.20%	14.87
uniform_37	14.8283	15.7605	6.29%	60.18	16.3249	10.09%	17.10
uniform_38	16.268	17.1421	5.37%	60.02	17.965	10.43%	15.27
uniform_39	18.414	20.9242	13.63%	60.01	20.9141	13.58%	20.46
uniform_41	19.9123	20.825	4.58%	60.65	22.2831	11.91%	22.89
uniform_42	19.6141	23.5641	20.14%	60.01	22.6484	15.47%	24.70
uniform_43	18.1065	22.0848	21.97%	60.05	21.9639	21.30%	30.70
uniform_44	20.4505	26.3959	29.07%	60.02	25.3985	24.20%	31.86
uniform_45	17.5671	19.1757	9.16%	60.06	20.3062	15.59%	35.65
uniform_46	19.7402	25.587	29.62%	60.05	25.8794	31.10%	31.29
uniform_47	20.1386	27.8387	38.24%	60.05	25.5589	26.91%	28.03
uniform_48	17.2657	19.3624	12.14%	60.01	20.048	16.11%	32.13
uniform_49	17.0555	19.7386	15.73%	60.03	21.0846	23.62%	35.78
uniform_50	17.0555	19.9934	17.23%	60.02	19.5921	14.87%	31.08