

MOMH pour le TSUFLP

Nicolas Compère & Nacim Chafaa

Master 2 ORO - Décembre 2023

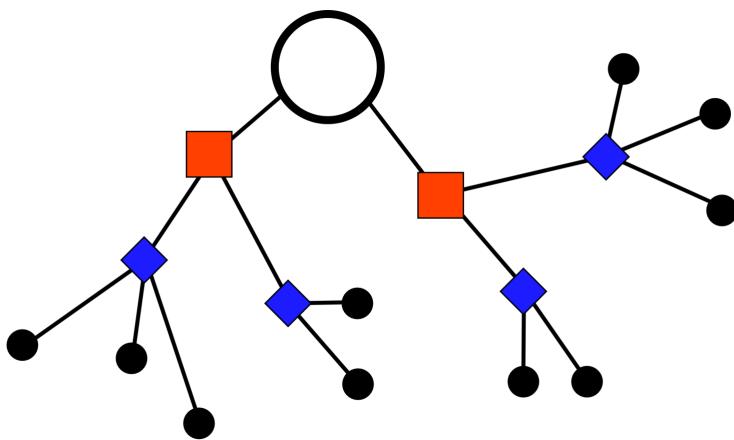


Table des matières

1	Introduction	3
2	Exemple d'application	3
3	Formulation du problème	3
3.1	Données	3
3.2	Variables	4
3.3	Seconde fonction objectif et contrainte additionnelle	4
3.4	Modèle mathématique	4
4	Scatter search pour le TSUFLP	4
4.1	Algorithme GRASP	5
4.2	Algorithme Tabu	5
4.2.1	Tabu Search Best Improvement	6
4.2.2	Tabu Search First Improvement	6
4.3	Combinaisons et amélioration des solutions	7
5	Expérimentation numérique	8
5.1	Environnements de tests	9
5.2	Structures de données	9
5.3	Premières observations et intuition	9
5.4	Résultats obtenus avec le solveur	9
5.5	Résultats obtenus avec Scatter Search	10
6	Comparaison des résultats	10
6.1	Tabu Best vs First	13
7	Pistes d'améliorations	14
8	Conclusion	14
A	Plots des instances	16
A.1	small1	16
A.2	small5	18
A.3	small10	21
A.4	small15	23
A.5	small20	26
A.6	medium1	28
A.7	medium5	31
A.8	medium10	33
A.9	large2	36

1 Introduction

Le problème de localisation d'usines sans contraintes de capacité est un problème central dans le monde de la logistique et de la recherche opérationnelle, où l'objectif est de déterminer un emplacement optimal de lieux de production afin de minimiser les coûts totaux, entre installation et approvisionnement des clients. Parmi les nombreuses variantes de ce problème, le "Two-Stage Uncapacitated Facility Location Problem" (Problème de Localisation d'Usines à Deux Étapes sans Contraintes de Capacité), que l'on notera "TSUFLP" occupe une place significative en raison de sa pertinence dans des contextes réels.

Nous commençons avec un ensemble donné d'emplacements de terminaux, un ensemble d'emplacements potentiels pour l'installation des concentrateurs de premier niveau et un ensemble de sites potentiels pour l'emplacement des concentrateurs de deuxième niveau. Une solution efficace, compte tenu de multiples objectifs, vise à choisir des emplacements au premier et deuxième niveau pour l'installation d'un certain nombre de concentrateurs, et à effectuer les affectations nécessaires de chaque terminal à un concentrateur de premier niveau installé, qui doit ensuite être attribué à l'un des concentrateurs établis sur le deuxième niveau. Le problème implique un schéma d'allocation unique, ce qui signifie que chaque terminal est attribué à exactement un concentrateur de premier niveau précédemment établi, tandis que chaque concentrateur est attribué à au plus un concentrateur précédemment localisé au deuxième niveau.

2 Exemple d'application

Il existe de nombreux exemples de problèmes qui nécessitent la résolution d'un TSUFLP. Par exemple, nous considérons une entreprise de distribution qui doit décider où construire des usines ainsi que des entrepôts, les usines (concentrateurs de niveau 2) servant à approvisionner les entrepôts (concentrateurs de niveau 1), qui eux mêmes livrent les clients (terminaux). Le but étant de minimiser les coûts logistiques, à savoir le coût total des coûts d'installation des entrepôts et de transport des marchandises.

3 Formulation du problème

3.1 Données

- $I = \{1, \dots, nI\}$: Ensemble des lieux où sont situés les terminaux,
- $J = \{1, \dots, nJ\}$: Ensemble de potentiels lieux d'installation pour les concentrateurs de premier niveau,
- $K = \{1, \dots, nK\}$: Ensemble de potentiels lieux d'installation pour les concentrateurs de deuxième niveau,
- c_{ij} : Prix d'affectation d'un terminal $i \in I$ à un concentrateur de niveau 1 au lieu $j \in J$,
- b_{jk} : Coût d'installation d'un concentrateur de niveau 1 au lieu $j \in J$ et sa connexion à un concentrateur de niveau 2 au lieu $k \in K$,
- s_k : Coût d'installation d'un concentrateur de niveau 2 dans le lieu $k \in K$.

3.2 Variables

- $x_{ij} = \begin{cases} 1 & \text{si le terminal } i \in I \text{ est relié au concentrateur de niveau 1 } j \in J \\ 0 & \text{sinon} \end{cases}$,
- $y_{jk} = \begin{cases} 1 & \text{si le concentrateur de niveau 1 dans } j \in J \text{ est relié au concentrateur de niveau 2 } k \in K \\ 0 & \text{sinon} \end{cases}$,
- $z_k = \begin{cases} 1 & \text{si un concentrateur de niveau 2 est installé au lieu } k \in K \\ 0 & \text{sinon} \end{cases}$.

3.3 Seconde fonction objectif et contrainte additionnelle

Afin de traiter un problème bi-objectif, il a fallu choisir une fonction à rajouter au problème de localisation d'usines parmi trois fonctions additionnelles.

Après réflexion, notre choix s'est porté sur l'ajout d'un objectif "pull". Ce dernier vise à minimiser la distance maximale entre les terminaux et leur concentrateur de premier niveau le plus proche. Cert objectif se formule comme suit :

$$\min f_2(x, y) = \max\{c_{ij}x_{ij}\} \quad (1)$$

Comme contrainte additionnelle à ajouter au problème, nous avons choisi de restreindre le nombre maximal de concentrateurs de niveau 1 à un nombre C. Nous avons fixé ce dernier à 1/3 du nombre de concentrateurs de niveau 1. Cette contrainte est définie comme suit :

$$\sum_{j \in J} \sum_{k \in K} y_{jk} \leq C \quad (2)$$

3.4 Modèle mathématique

Nous avons rajouté la seconde fonction objectif ainsi que la contrainte additionnelle au modèle initial du problème de localisation d'usine à deux étages. Le modèle s'écrit donc comme suit :

$$\left[\begin{array}{l} \min f_1(x, y, z) = \sum_{i \in I} \sum_{j \in J} c_{ij}x_{ij} + \sum_{j \in J} \sum_{k \in K} b_{jk}y_{jk} + \sum_{k \in K} s_kz_k \quad (1) \\ \min f_2(x, y) = \max\{c_{ij}x_{ij}\} \quad (2) \\ \text{s/t} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \quad (3) \\ \quad x_{ij} \leq \sum_{k \in K} y_{jk} \quad \forall i \in I, \forall j \in J \quad (4) \\ \quad \sum_{k \in K} y_{jk} \leq z_k \quad \forall j \in J, \forall k \in K \quad (5) \\ \quad y_{jk} \leq 1 \quad \forall j \in J \quad (6) \\ \quad \sum_{j \in J} \sum_{k \in K} y_{jk} \leq C \quad (7) \\ \quad x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J \quad (8) \\ \quad y_{jk} \in \{0, 1\} \quad \forall j \in J, \forall k \in K \quad (9) \\ \quad z_k \in \{0, 1\} \quad \forall k \in K \quad (10) \end{array} \right]$$

4 Scatter search pour le TSUFLP

Dans le cadre de ce projet, une métaheuristique à population de type Scatter Search a été implémentée afin de résoudre le TSUFLP.

Cette métaheuristique, proposée par F. Glover en 1997 pour des problèmes mono-objectifs, a été reprise dans les années 2000 pour calculer une approximation d'ensemble de points non-dominés Y_N d'un problème multi-objectif.

Dans ce projet, nous nous baserons sur la méthode proposée dans l'article "A New Scatter Search Design for Multiobjective Combinatorial Optimization with an Application to Facility Location", de A. D. López-Sánchez, J. Sánchez-Oro et M. Laguna. [2]

Le principe de cette méthode se présente comme suit :

1. Dans un premier temps, une métaheuristique GRASP est utilisée afin de générer une première population de solutions réalisables, c'est la partie "génération et diversification" de la méthode.
2. Ensuite, un algorithme tabou permet d'améliorer les solutions initiales, et ce, sur les deux objectifs.
3. À partir de là, deux ensembles de solutions élites par rapport à f_1 et f_2 sont créés.
4. Pour finir, des combinaisons de solutions sont testées un algorithme Path Relinking sur plusieurs itérations. C'est la phase d'amélioration.

4.1 Algorithme GRASP

La génération d'une population de solutions initiales a été effectuée avec une métaheuristique GRASP. Cette dernière est une hybridation entre méthode gloutonne et aléatoire. Elle a comme entrée les données de l'instance ainsi qu'une valeur α . Après quelques tests, nous avons fixé sa valeur à 0.7.

Au terme d'un nombre d'itérations que l'on a fixé à 200, un nuage de 200 solutions réalisables est obtenu. À chacune de ces itérations, le GRASP s'effectue en quatres grandes étapes :

1. Choix aléatoire d'un premier concentrateur de niveau 1 puis ajout des autres en suivant α et en respectant le nombre maximum fixé à C .
2. Selection des arcs entre les terminaux et les concentrateurs de niveau 1 en suivant α .
3. Selection aléatoire d'un premier concentrateur de niveau 2, puis ajout d'autres concentrateurs de niveau 2 avec le paramètre α .
4. Selection des arcs qui relient chaque concentrateur de niveau 1 à l'un des concentrateurs de niveau 2 sélectionnés en suivant α .

Si bien que l'article [2] parle de réaliser un GRASP différent pour chacun des deux objectifs, nous avons convenu qu'il était préférable de n'en coder qu'un seul qui les satisferait tous les deux, évitant ainsi un coût calculatoire supplémentaire inutile. En effet, de part la nature aléatoire de la génération de solutions ainsi que la corrélation existante entre les deux objectifs, les résultats obtenus étaient équilibrés et ne penchaient en faveur daucun des critères, ce qui permettait d'envisager une amélioration par la suite.

4.2 Algorithme Tabu

Après obtention d'un ensemble de solutions réalisables initiales, nous avons utilisé un algorithme Tabu Search afin d'améliorer ces solutions. Dans un premier temps, nous avons développé un Tabu Search Best Improvement mais nous avons réalisé par la suite que ce n'était pas un tabu adapté pour notre problème car trop lourd. Nous avons donc implémenté un Tabu Search First Improvement qui correspond mieux à ce dont nous avions besoin pour notre Scatter Search. Pour les deux versions,

nous avons une liste Tabu de taille 7 comme conseillé dans le papier Tabu search de Fred GLOVER [1] et le nombre d'itération est fixé à 10.

4.2.1 Tabu Search Best Improvement

Pour notre première version du Tabu Search du type Best Improvement, nous avons implémenté un algorithme assez simple qui explore un seul type de voisinage, celui des concentrateurs de niveau 1. L'algorithme est comme suit :

Algorithme 1 : Tabu search Best Improvement

Entrées : Solution du GRASP, nbIter, tailleTabu

Sorties : Solution améliorée

iter $\leftarrow 1$;

tant que iter \leq nbIter **et** liste candidats non vide **faire**

Construire la liste candidats avec tout les mouvements possibles* ;

Choisir le candidat avec la valeur objective la plus petite ;

si mouvement est dans liste tabu **alors**

| Choisir le meilleur mouvement après le mouvement sélectionné au préalable

fin

Faire le swap des terminaux de niveau 1 ;

Les terminaux sont attachés aux concentrateurs de niveau 1 les plus proches ;

Le nouveau concentrateur de niveau 1 hérite des concentrateurs de niveau 2 de l'ancien concentrateur de niveau 1 ;

si liste tabu de taille 7 **alors**

Supprimer le premier mouvement inséré ;

Insérer le mouvement réalisé

fin

si nouvelle solution est la meilleure **alors**

meilleur solution \leftarrow nouvelle solution

fin

iter \leftarrow iter + 1

fin

retourner meilleur solution

* C'est à dire tout les swaps de concentrateurs de niveau 1 possibles.

Notre implémentation de cet algorithme ne prend en compte que l'amélioration par rapport à la première fonction objectif z_1 . Cependant, étant donné que les deux objectifs ne sont pas très contradictoires, nous avons observé une amélioration sur les deux axes en même temps lors de l'exécution. Les résultats obtenus se rapprochent très fortement des solutions fournis par le solveur MOA. Cependant, l'algorithme a une complexité dans le temps qui grandit très rapidement sur les plus grandes instances (voir section Expérimentation numérique).

Suite à la présentation de nos avancés, nous avons discuté avec nos camarades des différentes méthodes du tabu search et nous avons réalisé qu'une méthode Best Improvement serait plus adapté à notre problème.

4.2.2 Tabu Search First Imporvement

Pour cette version du Tabu Search, nous explorons toujours un seul type de voisinage, les swaps de concentrateurs de niveau 1 mais nous nous intéressons uniquement au premier mouvement améliorant.

Ceci permet de grandement limité le nombre de voisinage à explorer et nous amélioront ainsi les temps de calculs au détriment de la qualité des solutions.

Algorithme 2 : Tabu search First Improvement

Entrées : Solution du GRASP, nbIter, tailleTabu

Sorties : Solution améliorée

iter $\leftarrow 1$;

tant que iter \leq nbIter **et on a un candidat faire**

Choisir le premier mouvement améliorant parmi les swaps de concentrateurs de niveau 1 possibles ;

si mouvement est dans liste tabu **alors**

| Choisir un autre mouvement améliorant

fin

Faire le swap des terminaux de niveau 1 ;

Les terminaux sont attachés aux concentrateurs de niveau 1 les plus proches ;

Les concentrateurs de niveau 2 sont attachés aux concentrateurs de niveau 1 les plus proches ;

si liste tabu de taille 7 **alors**

| Supprimer le premier mouvement inséré ;

| Insérer le mouvement réalisé

fin

si nouvelle solution est la meilleure **alors**

| meilleur solution \leftarrow nouvelle solution

fin

iter \leftarrow iter + 1

fin

retourner meilleur solution

Les solutions se rapprochent moins des solutions du solveur MOA mais améliorent tout de même les solutions du GRASP en des temps bien plus rapide. Nous avons implémenté cette fois ci une fonction tabu pour chaque fonction objectif z_1 et z_2 .

4.3 Combinaisons et amélioration des solutions

Une fois que l'on a obtenu deux ensembles de solutions réalisables, améliorées sur chacun des objectifs respectivement, nous les mettons dans deux ensembles que l'on appellera refSets (ensembles de référence). Ces derniers possèdent chacun β solutions, dont les premières moitiées sont constituées des solutions élites par rapport à chacun des objectifs respectivement.

Afin d'aider à la diversification, la deuxième moitié de chaque refSet possède les solutions dont la distance avec ce dernier est maximale. La distance d'une solution par rapport à un refSet est évaluée comme la distance minimale entre cette solution et chaque autre solution dans le refSet. Une distance entre deux solutions S_1 et S_2 est définie comme le nombre de concentrateurs présents dans S_1 mais pas dans S_2 , et vice-versa. Donc, pour chaque solution $S_i \in \text{refSet}$, on calcule la distance $d_i(S_i, \text{refSet})$. La solution avec la distance maximale et sélectionnée et rajoutée au refSet. Ceci constituera donc la seconde moitié de l'ensemble.

Suite à cela, les solutions présentes dans chacun des refSets sont combinées entre elles en utilisant un algorithme de Path Relinking. Ce dernier est une technique utilisée dans les métahéuristiques pour explorer l'espace des solutions d'une manière plus intensive en combinant deux solutions déjà trouvées, et ce, dans le but d'améliorer les solutions.

Algorithme 3 : Path Relinking

Entrées : sol1, sol2, nbIter

Sorties : Ensemble de solutions

new_sol \leftarrow sol1 ;

diff1 \leftarrow les concentrateurs de niveau 1 qui sont dans sol1 mais pas dans sol2 ;

diff2 \leftarrow les concentrateurs de niveau 1 qui sont dans sol2 mais pas dans sol1 ;

tant que iter \leq nbIter **et** diff1 non vide **et** diff2 non vide **faire**

 In \leftarrow Choisir un concentrateur dans diff2 ;

 Out \leftarrow Choisir un concentrateur dans diff1 ;

 Enlever Out et ajouter In dans new_sol ;

 Connecter à In les terminaux et concentrateurs de niveau 2 de Out ;

si new_sol est meilleur **alors**

 Ajouter new_sol à l'ensemble de solutions ;

 Recalculer diff1 et diff2 ;

sinon

 | new_sol \leftarrow sol1 ;

fin

fin

fin

new_sol \leftarrow sol1 ;

diff1 \leftarrow les concentrateurs de niveau 2 qui sont dans sol1 mais pas dans sol2 ;

diff2 \leftarrow les concentrateurs de niveau 2 qui sont dans sol2 mais pas dans sol1 ;

tant que iter \leq nbIter **et** diff1 non vide **et** diff2 non vide **faire**

 In \leftarrow Choisir un concentrateur dans diff2 ;

 Out \leftarrow Choisir un concentrateur dans diff1 ;

 Enlever Out et ajouter In dans new_sol ;

 Connecter à In les concentrateurs de niveau 2 de Out ;

si new_sol est meilleur **alors**

 Ajouter new_sol à l'ensemble de solutions ;

 Recalculer diff1 et diff2 ;

sinon

 | new_sol \leftarrow sol1 ;

fin

fin

fin

retourner ensemble de solutions

L'algorithme est implémenté avec 20 itérations par défaut. Nous prenons deux solutions **sol1** et **sol2** et nous calculons les différences entre les deux solutions pour les concentrateurs de niveau 1 puis de niveau 2. Ensuite, nous remplaçons les concentrateurs de **sol1** par ceux de **sol2** un par un. Les nouveaux concentrateurs héritent des anciens les terminaux et concentrateurs de niveau 2. Seules les solutions améliorantes sont gardées puis améliorer avec le tabu search.

5 Expérimentation numérique

Afin de tester l'efficacité de la méthode ainsi que sa robustesse dans un contexte pratique, nous avons mené une expérimentation numérique sur certaines instances. Nous avons ensuite comparé ces

résultats à ceux obtenus avec un solveur exacte.

5.1 Environnements de tests

Les tests ont été effectués sur des instances d'UFLP de "Sanchez" que l'on a trouvé sur internet. Ces dernières proposaient, à différentes proportions, des ensembles de coordonnées pour des terminaux ainsi que pour de potenntiels concentrateurs. Étant donné que ces instances ne proposaient pas de séparation du nombre de concentrateurs en deux étages, nous avons divisé cet ensemble en allouant 4/5 de ces localisations aux concentrateurs de premier niveau, le cinquième restant étant pour les concentrateurs de second niveau.

Les essais avec le solveur exact ont été réalisées sur l'ecosystème MultiObjectiveAlgorithms (MOA), successeur de vOptGeneric, dans sa version 1.3.1. La résolution du problème a ensuite été confiée au solveur HiGHS dans sa version 1.7.5. Le tout sous Julia 1.9.4 et l'environnement JuMP v1.17.0.

Les test ont été effectués sur une machine équipée d'un processeur Apple M2 à 8 coeurs couplé à 16Gb de mémoire vive partagée entre CPU et GPU.

5.2 Structures de données

Les structures de données utilisées lors de la réalisation de ce projet sont les suivantes :

- Objet **instance** : Permet de rassembler tous les attributs de l'instance, tels que les terminaux, les concentrateurs de niveau 1 et 2 respectivement, les coûts d'ouverture des concentrateurs, les distances séparant les terminaux et les concentrateurs de niveau 1, ainsi que les concentrateurs de niveau 1 et 2 entre eux.
- Objet **solution** : Rassemble les attributs d'une solution, on y trouve les valeurs des deux objectifs, les concentrateurs de niveau 1 et 2 sélectionnés, les arcs qui relient teminaux/concentrateurs de niveau 1 ainsi que les arcs qui relient les concentrateurs de niveau 1 et 2 entre eux.
- SkipLists : Structure de données probabilistes qui permet de stocker les solutions non-dominées et de maintenir un front de Pareto. Cette dernière n'a pas été implémentée par nous mêmes. En effet, nous avons repris un code de SkipList sur Julia trouvé sur GitHub [3].

5.3 Premières observations et intuition

Dès les premiers essais avec le solveur, nous avons pu constater que le nombre de points non-dominés résultant était relativement faible. Ceci nous a amené à penser qu'il y avait une certaine corrélation entre les deux fonctions objectifs. En effet, ceci s'explique par le fait que ces dernières visent toutes les deux à minimiser des distances. Elles ne sont donc pas tout à fait contradictoires.

Nous avons également pensé que modifier le paramètre C pouvait influer sur les solutions ce qui s'est avéré exact.

5.4 Résultats obtenus avec le solveur

Les résultats obtenus avec le solveur sont sans surprise de bonne qualité. Nous obtenons plusieurs solutions non dominés pour toutes les instances **small** et **medium**. En revanche le solveurs MOA ne

parviens pas à fournir des résultats pour les instances `large` avec la limite de temps fixé à 300 secondes.

5.5 Résultats obtenus avec Scatter Search

Nous obtenons des résultats satisfaisants pour le Scatter Search. L'ensemble de solutions générées aléatoirement par GRASP est bien amélioré par le Tabu Search. Le Best Improvement permet d'obtenir des résultats très proche de MOA au détriment du temps de calcul tandis que le First Improvement permet une amélioration moins agressive mais très rapidement.

Le Path Relinking ne permet pas une amélioration très nette des solutions obtenus avec Tabu, les meilleures solutions obtenus étant parfois moins bonnes. Ceci est certainement du à une mauvaise gestion de l'algorithme du Path Relinking (voir Pistes d'améliorations).

6 Comparaison des résultats

Instance	CPU MOA	CPU Méta
<code>small1</code>	6,37	1,23
<code>small5</code>	1,45	1,17
<code>small10</code>	3,48	1,1
<code>small15</code>	0,39	1,25
<code>small20</code>	0,31	1,14
<code>medium1</code>	39,49	8,26
<code>medium5</code>	10,03	6,82
<code>medium10</code>	18,6	8,06
<code>large2</code>	NA	938,98
<code>large5</code>	NA	1011,63

TABLE 1 – Temps CPU en secondes

Nous obtenons des temps de calculs généralement meilleurs pour les instances `small` hormis certaines exceptions en rouge. Notre métaheuristique est cependant meilleur pour toutes les autres instances `medium` et `large`. Pour ces derniers, nous obtenons des temps de calculs supérieurs à la limite fixé de 300 secondes pour le solveurs MOA.

	MOA		GRASP		Tabu		Path Relinking	
Instance	z_1	z_2	z_1	z_2	z_1	z_2	z_1	z_2
small1	1179	38	1807	66	1546	47	1242	43
small5	1106	29	1672	65	1407	42	1166	37
small10	1107	41	1729	61	1378	49	1198	47
small15	1152	36	1814	66	1493	48	1271	61
small20	1065	36	1635	58	1436	46	1149	41
medium1	1439	26	2864	53	2715	41	2225	35
medium5	1428	28	3372	60	3045	50	2268	40
medium10	1417	25	3069	58	2789	47	2252	44
large2	NaN	NaN	11740	57	11624	50	10626	56
large5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

TABLE 2 – Évolution des valeurs moyennes des solutions

On peut constater une évolution constante des solutions du GRASP vers le Path Relinking pour les deux fonctions objectifs, en se rapprochant des solutions du solveur MOA.

Instance	MOA				GRASP			
	Min z ₁	Min z ₂	Max z ₁	Max z ₂	Min z ₁	Min z ₂	Max z ₁	Max z ₂
small1	1091	36	1253	43	1407	43	2241	95
small5	1068	29	1070	38	1382	43	2272	130
small10	1081	34	1238	46	1466	46	2166	101
small15	1130	36	1130	36	1450	42	2382	111
small20	1062	36	1062	36	1303	41	1949	113
medium1	1389	21	1421	30	2525	33	3232	100
medium5	1474	25	1474	25	2829	43	4050	95
medium10	1389	25	1392	30	2618	40	3595	90
large2	NaN							
large5	NaN							
	Tabu				Path Relinking			
	Min z ₁	Min z ₂	Max z ₁	Max z ₂	Min z ₁	Min z ₂	Max z ₁	Max z ₂
small1	1247	36	1985	86	1130	43	1338	46
small5	1120	29	2079	82	NaN	NaN	NaN	NaN
small10	1211	36	1834	64	1174	46	1377	64
small15	1306	36	2068	75	1242	42	1402	60
small20	1129	38	1895	79	1154	40	1155	41
medium1	2241	31	3200	78	2101	33	2644	73
medium5	2147	31	3618	87	1985	31	2492	61
medium10	2199	28	3459	89	2057	32	2583	62
large2	NaN							
large5	NaN							

TABLE 3 – Évolution des valeurs min et max des solutions

On peut constater sur le Path Relinking qu'il ne fournit parfois aucune solutions améliorante par rapport au Tabu Search comme observé sur l'instance small5.

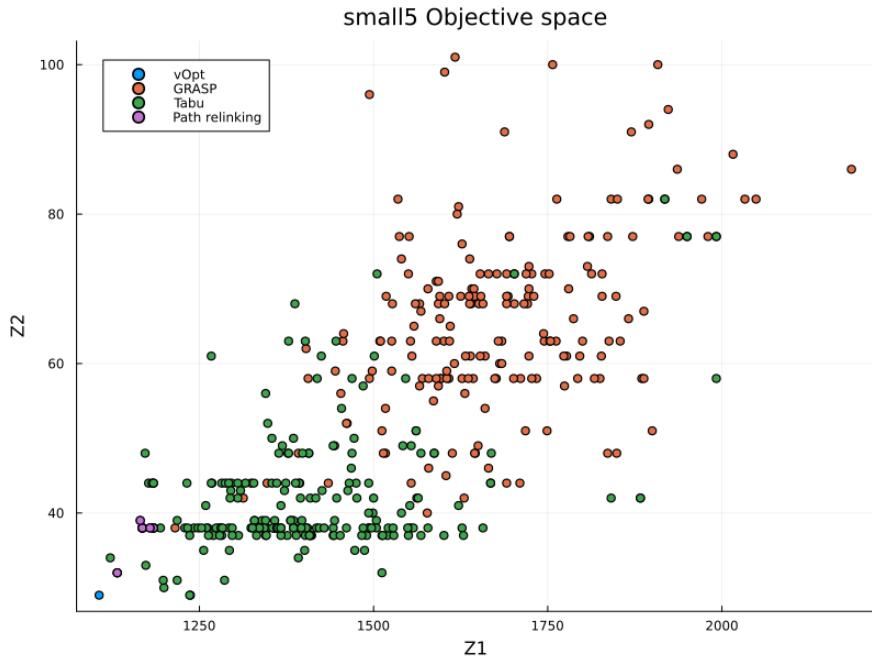


FIGURE 1 – small5 Solution space

On constate bien sur l'exemple de la figure 1 une évolution des solutions vers les solutions de MOA. Nous pouvons isoler les points non-dominés grâce à la Skip-List et voir que le Tabu améliore grandement les solutions du GRASP et que le Path Relinking améliore légèrement les solutions du Tabu.

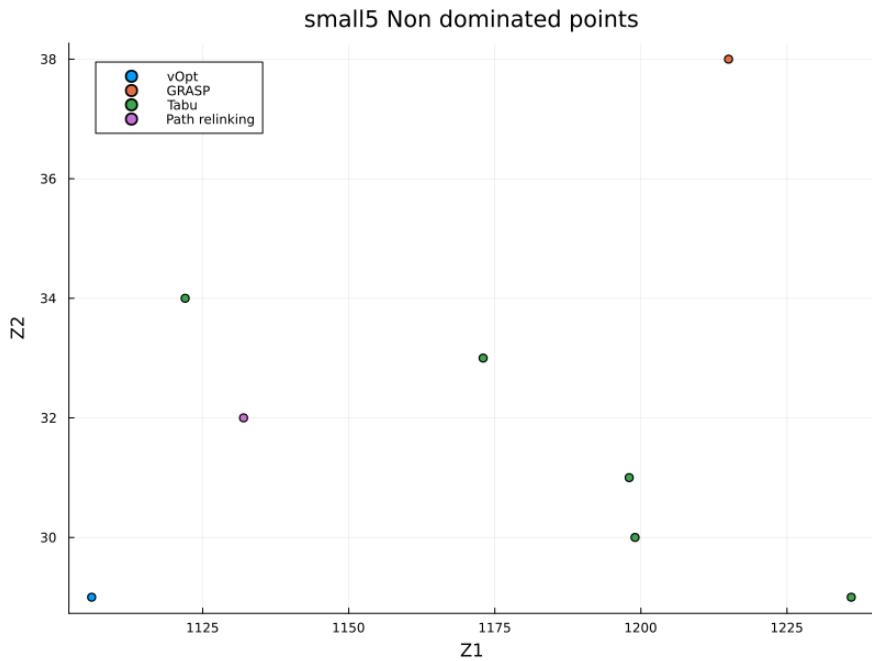


FIGURE 2 – small5 Non dominated points

6.1 Tabu Best vs First

Comme évoqué précédemment, un Tabu Search Best Improvement permet d'obtenir des solutions très efficaces et qui se rapproche beaucoup des solutions exactes.

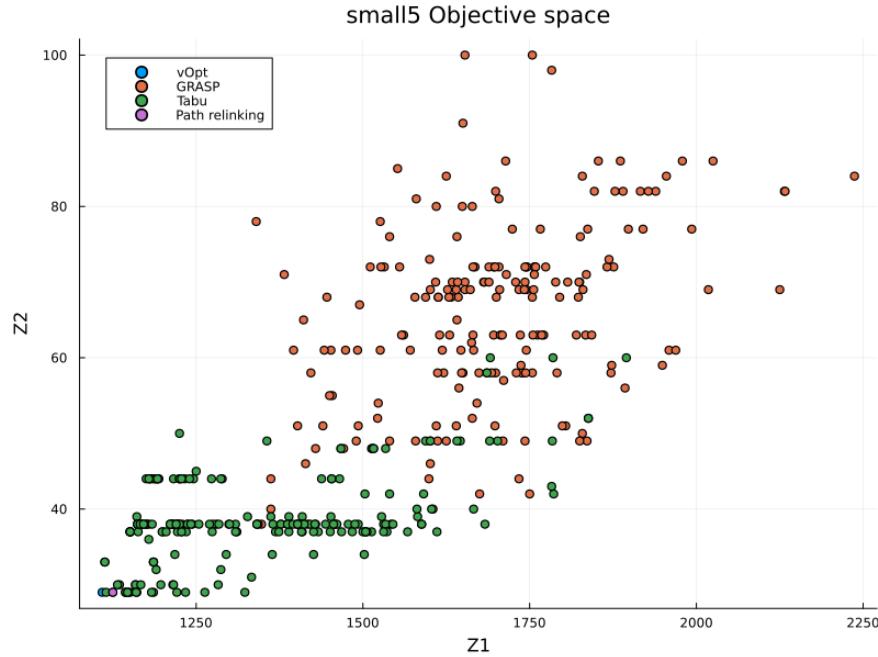


FIGURE 3 – small5 Solution space with Tabu Search Best Improvement

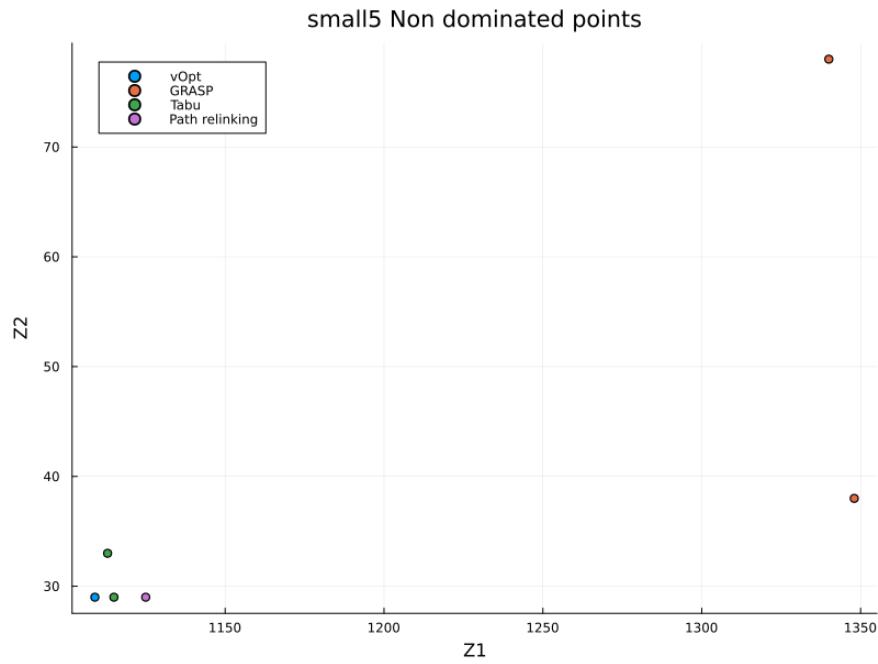


FIGURE 4 – small5 Non dominated points with Tabu Search Best Improvement

7 Pistes d'améliorations

De nombreux points pourraient être améliorés dans ce projet.

- Tester d'autres voisinages pour le tabu search (swaps de concentrateurs de niveau 2, différentes stratégies pour les nouvelles liaisons,...)
- Améliorer le path relinking pour qu'il colle mieux à ce qui est décrit dans le papier [2]
- Essayer plus de variations de paramètres pour voir les différents comportements du Scatter Search

8 Conclusion

Ce projet a été l'occasion pour nous d'approfondir nos connaissances et d'accroître nos compétences en météahéuristique, plus particulièrement dans un contexte multi-objectif. Le fait d'étudier cela sur le TSUFLP a été très intéressant et enrichissant. Nous avons notamment pu voir ce que la méthode Scatter Search avait à offrir, ses avantages ainsi que ses limites. Il aurait été intéressant d'explorer différents paramètres pour observer les comportements des algorithmes.

Nous sommes tout de même satisfait d'avoir une implémentation qui fonctionne.

Références

- [1] Fred GLOVER et Manuel LAGUNA. *Tabu search I*. T. 1. Jan. 1999. ISBN : 978-0-7923-9965-0. DOI : 10.1287/ijoc.1.3.190.
- [2] AD LÓPEZ-SÁNCHEZ, Jesús SÁNCHEZ-ORO et Manuel LAGUNA. “A new scatter search design for multiobjective combinatorial optimization with an application to facility location”. In : *INFORMS Journal on Computing* 33.2 (2021), p. 629-642.
- [3] *ScatterSearchMOMH/SkipList.jl at master · ValentinDeguil/ScatterSearchMOMH — github.com*. <https://github.com/ValentinDeguil/ScatterSearchMOMH/blob/master/SkipList.jl>. [Accessed 30-12-2023].

A Plots des instances

A.1 small1

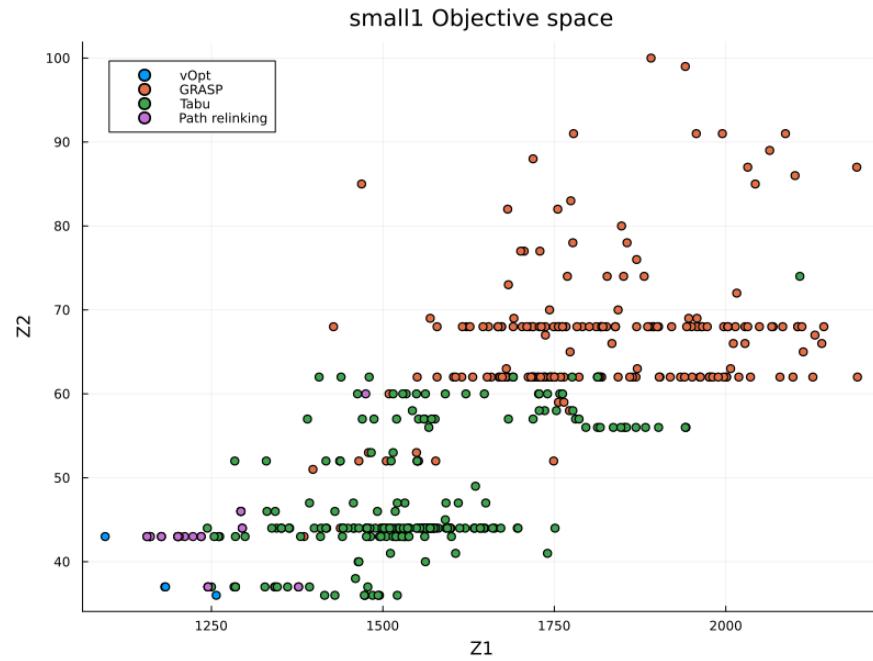


FIGURE 5 – small1 Solution space

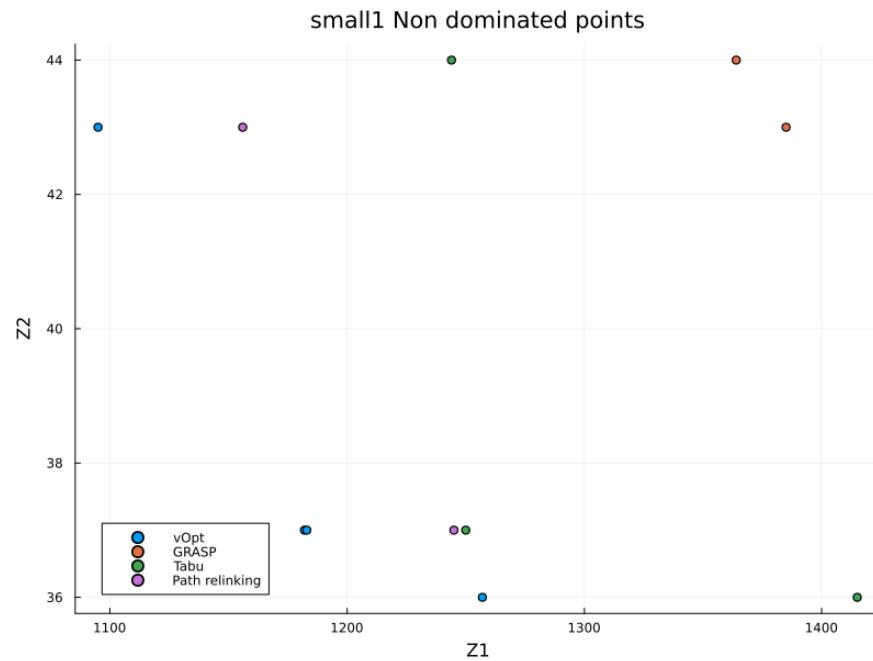


FIGURE 6 – small1 Y_N

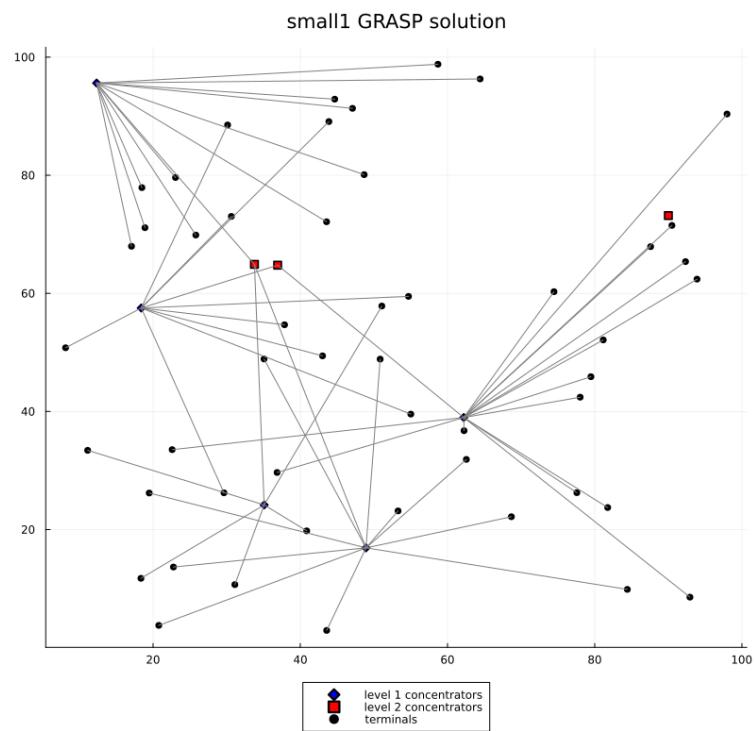


FIGURE 7 – Exemple : small1 GRASP solution

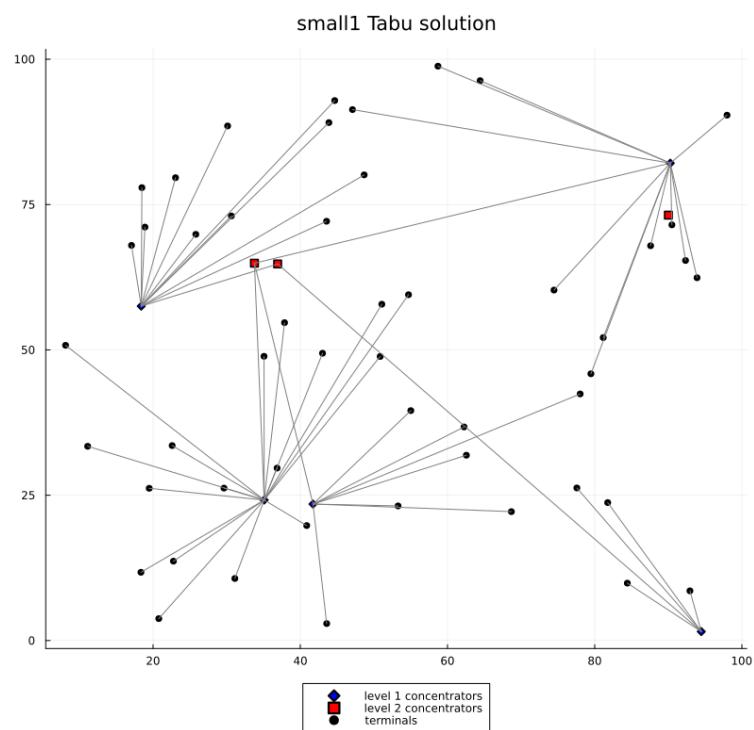


FIGURE 8 – Exemple : small1 Tabu solution

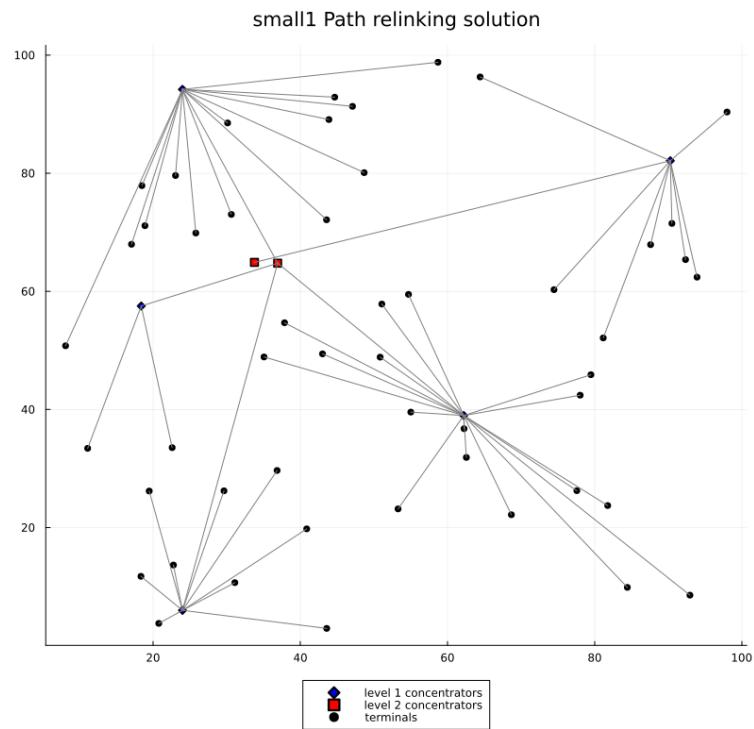


FIGURE 9 – Exemple : small1 Path Relinking solution

A.2 small5

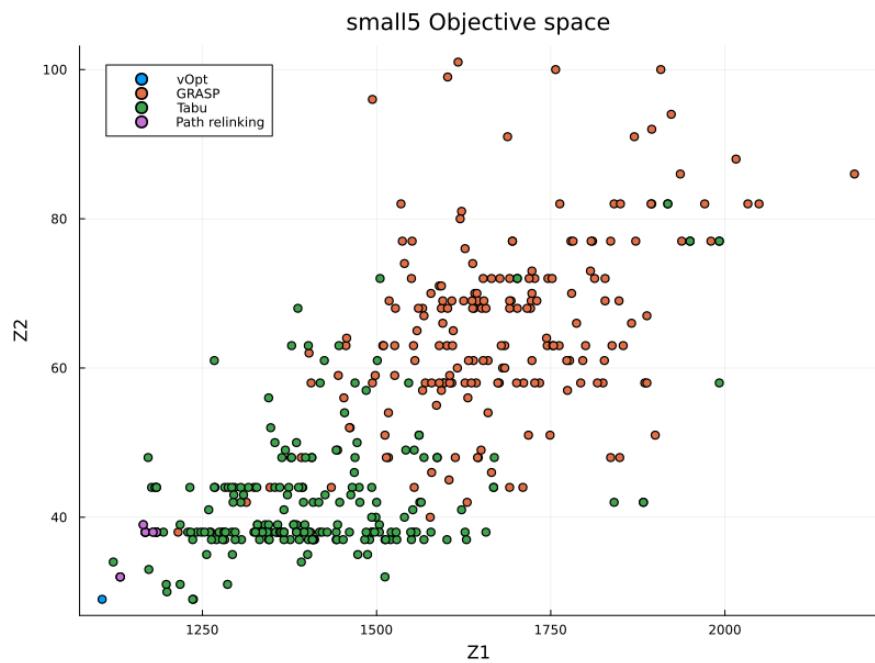


FIGURE 10 – small15 Solution space

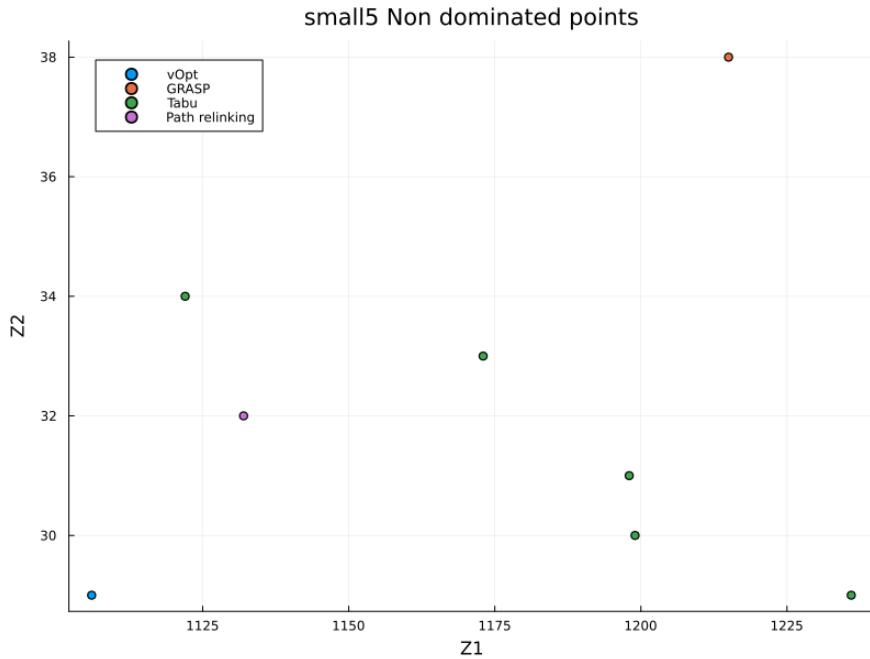


FIGURE 11 – small15 Y_N

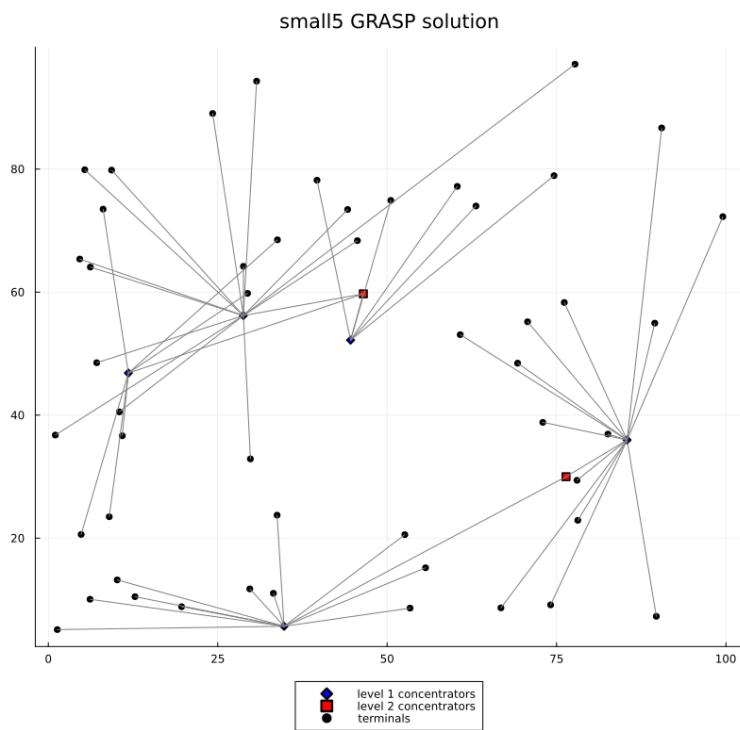


FIGURE 12 – Exemple : small15 GRASP solution

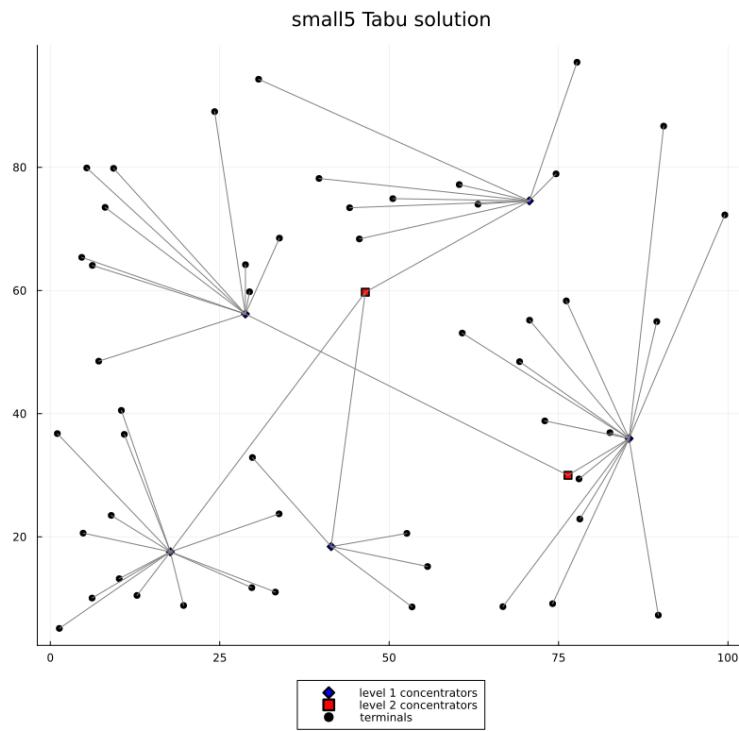


FIGURE 13 – Exemple : **small5** Tabu solution

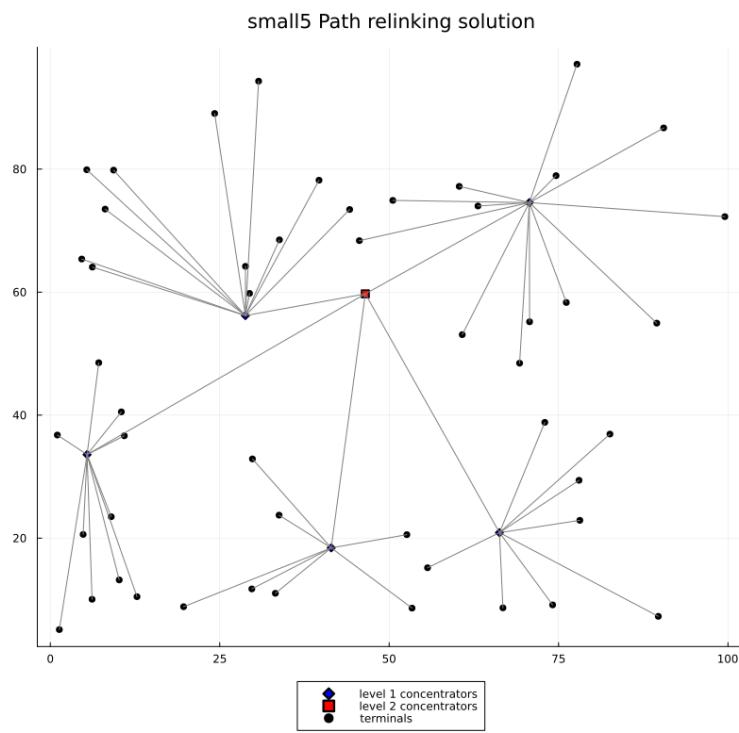


FIGURE 14 – Exemple : **small5** Path Relinking solution

A.3 small10

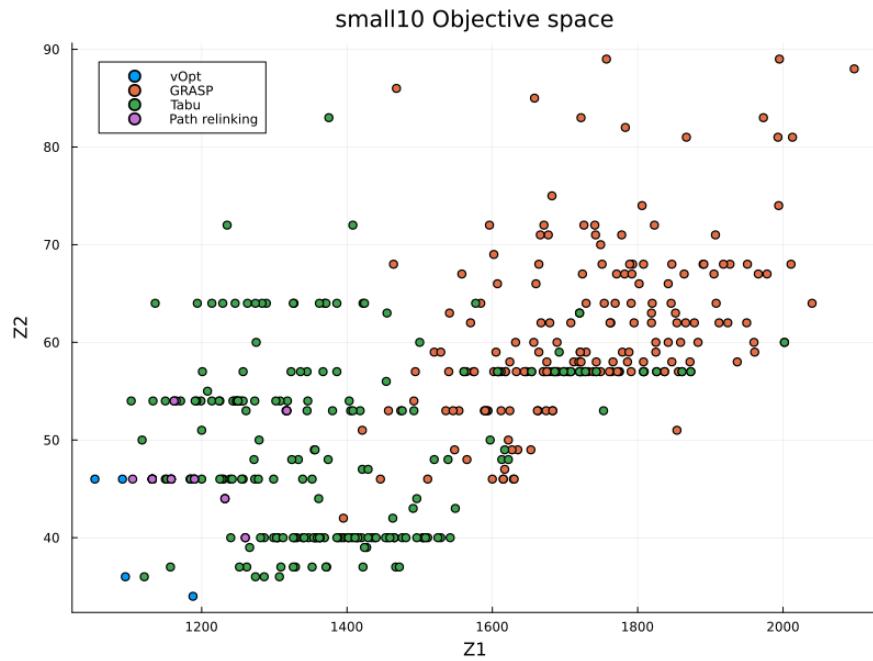


FIGURE 15 – small10 Solution space

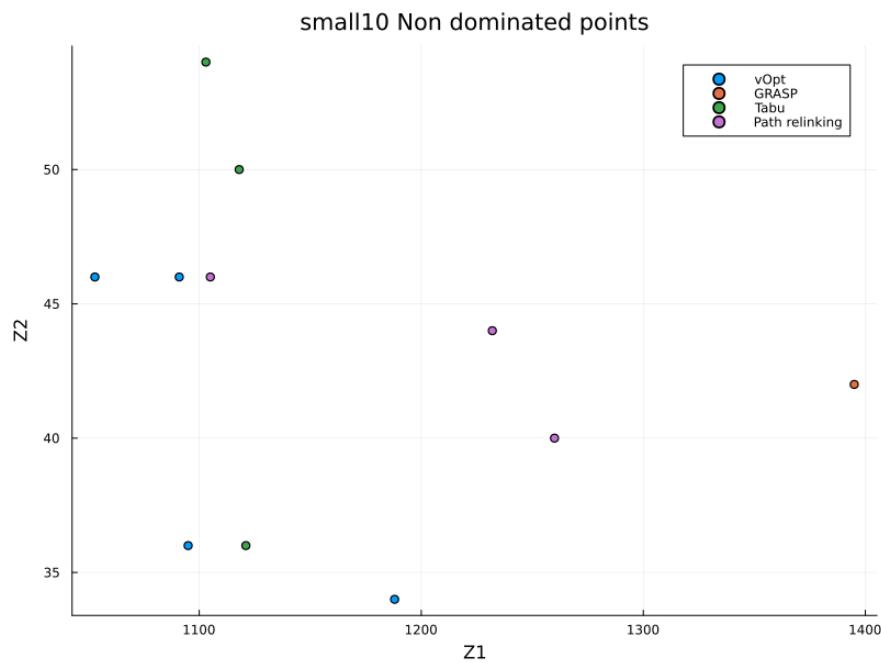


FIGURE 16 – small10 Y_N

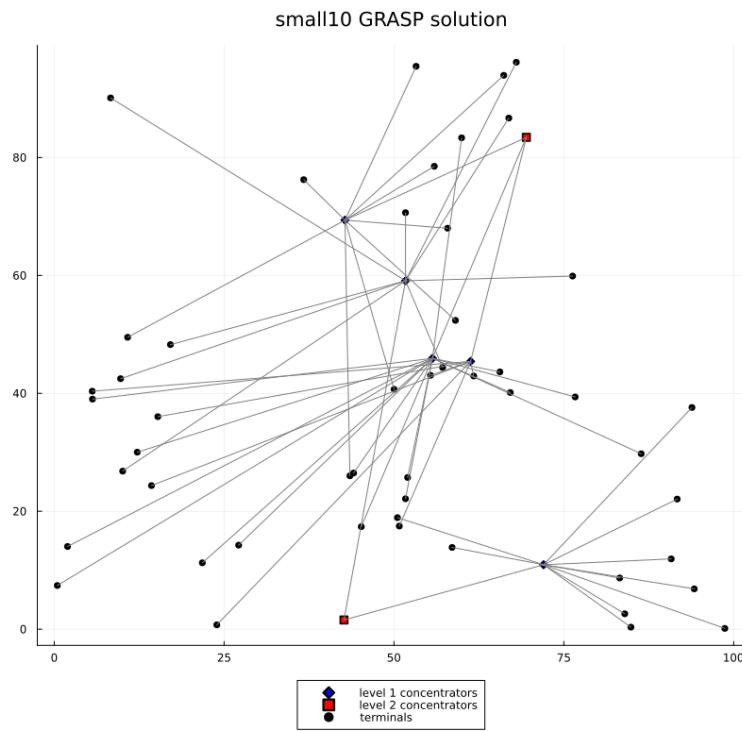


FIGURE 17 – Exemple : small10 GRASP solution

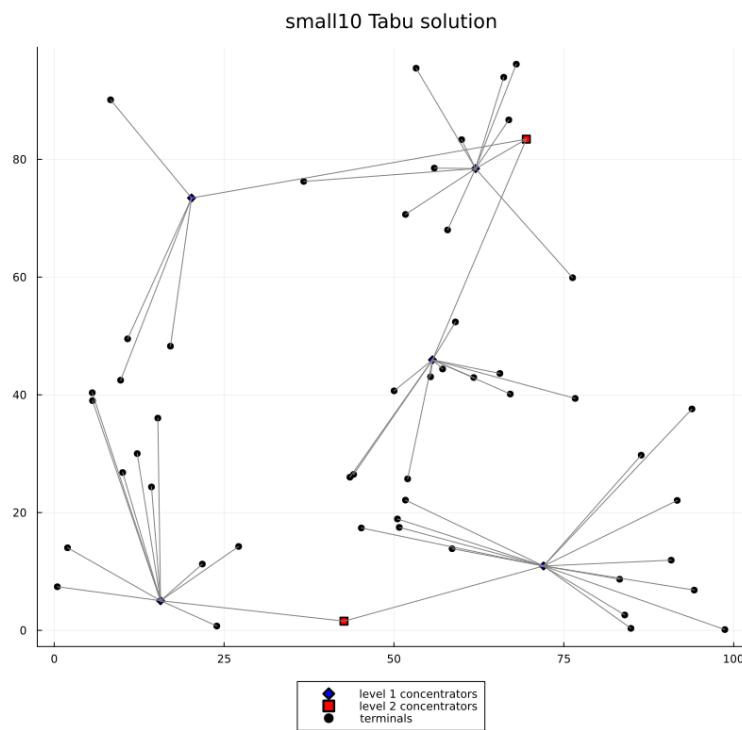


FIGURE 18 – Exemple : small10 Tabu solution

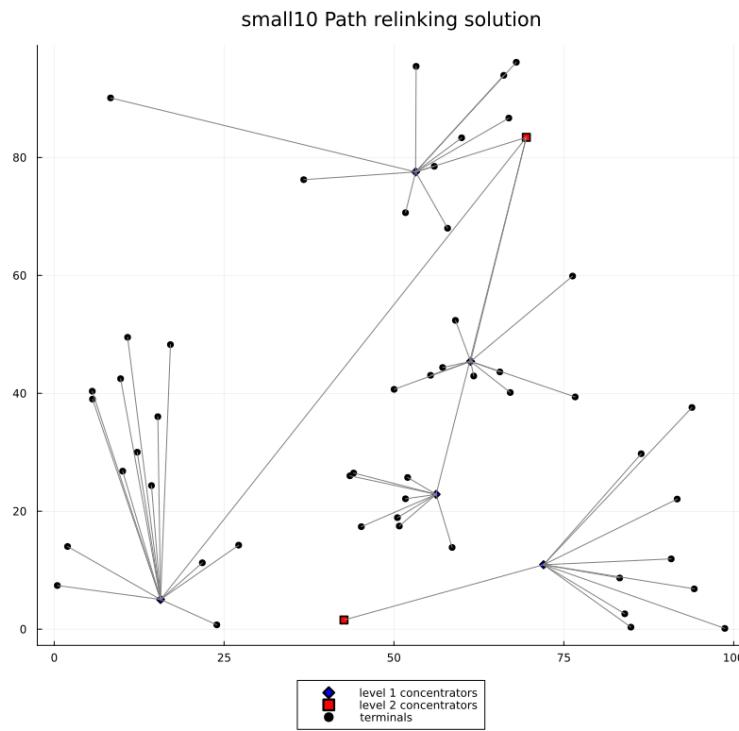


FIGURE 19 – Exemple : **small10** Path Relinking solution

A.4 small15

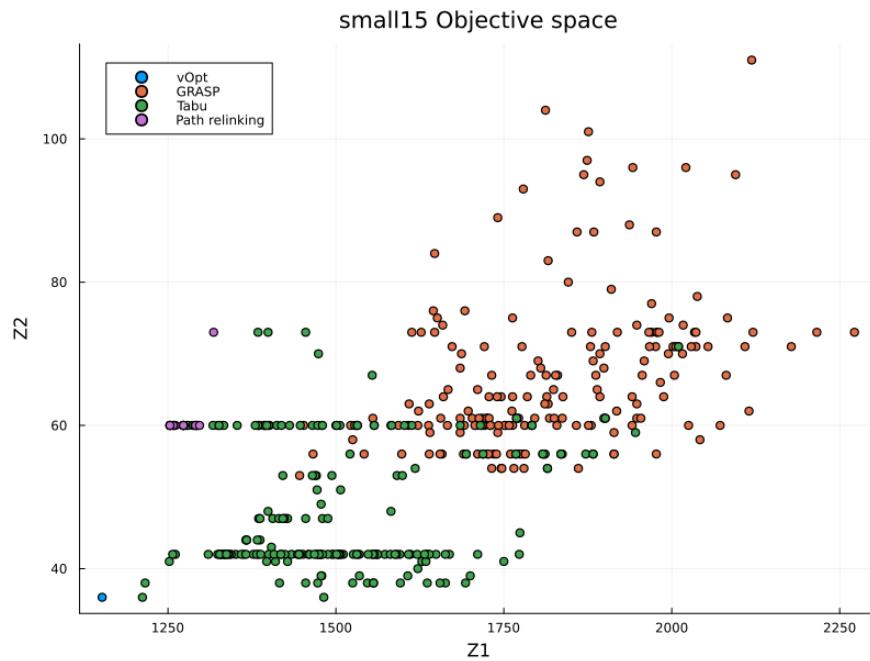


FIGURE 20 – **small15** Solution space

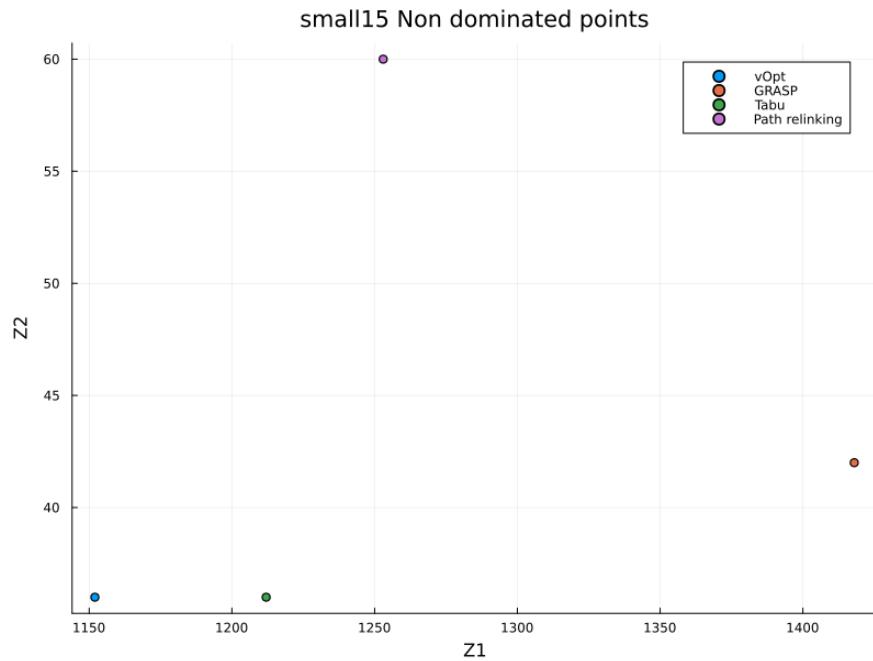


FIGURE 21 – small15 Y_N

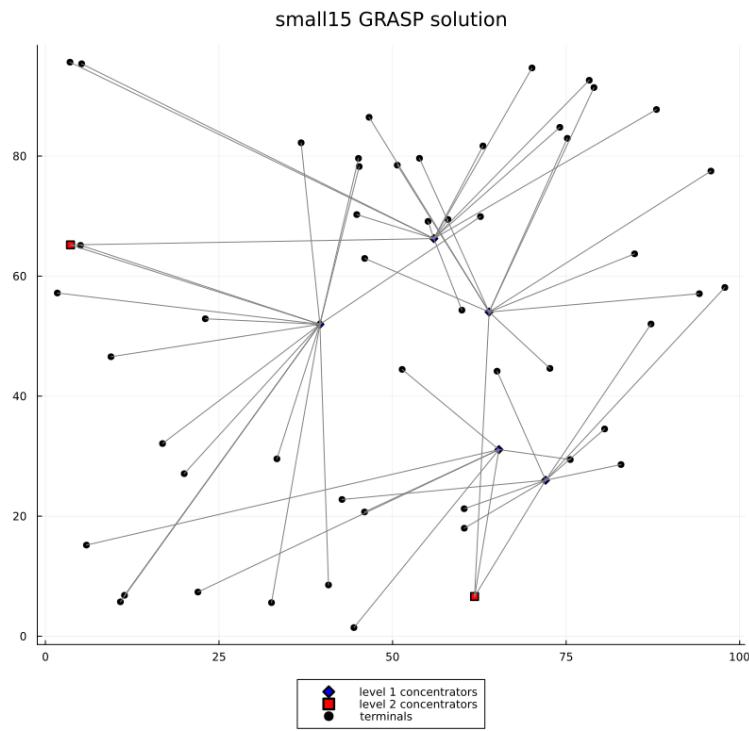


FIGURE 22 – Exemple : small15 GRASP solution

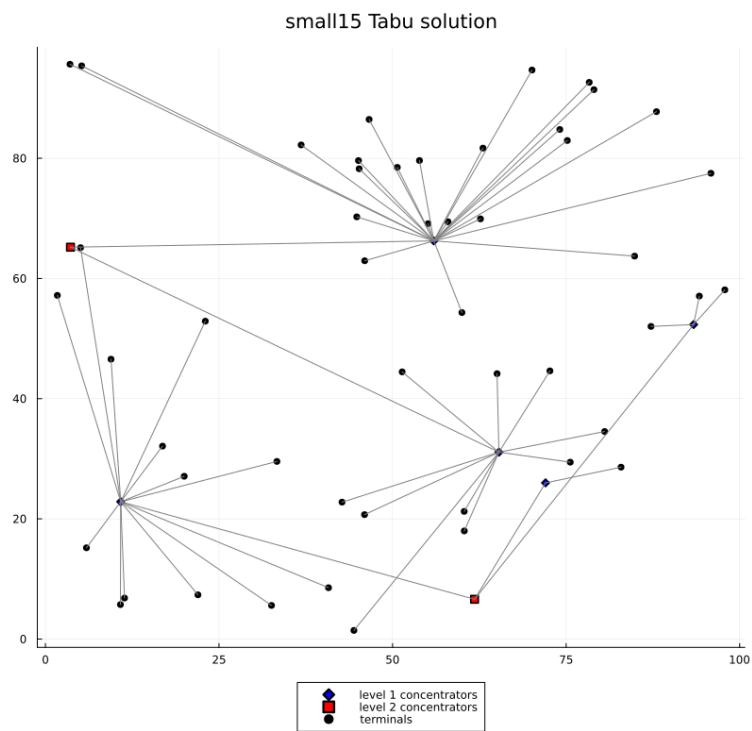


FIGURE 23 – Exemple : small15 Tabu solution

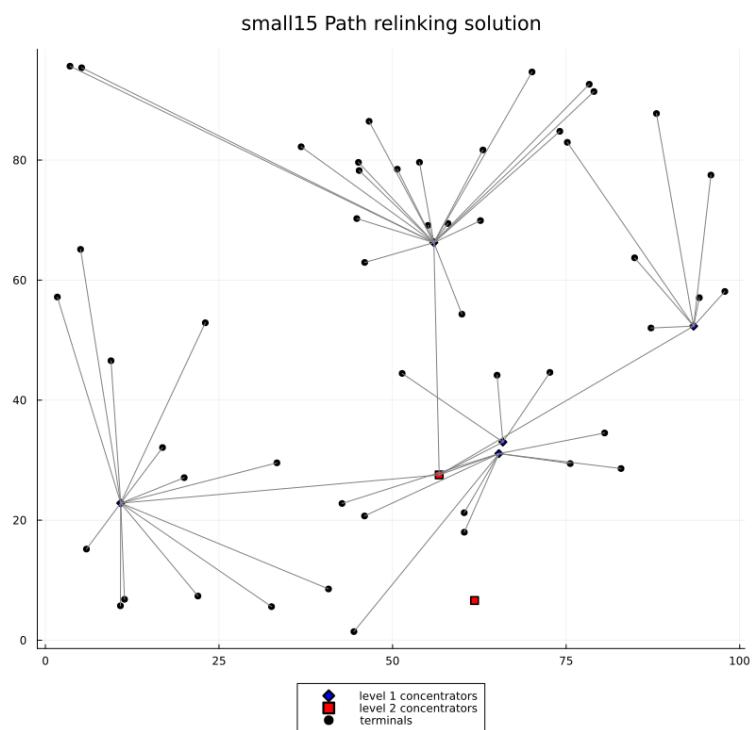


FIGURE 24 – Exemple : small15 Path Relinking solution

A.5 small20

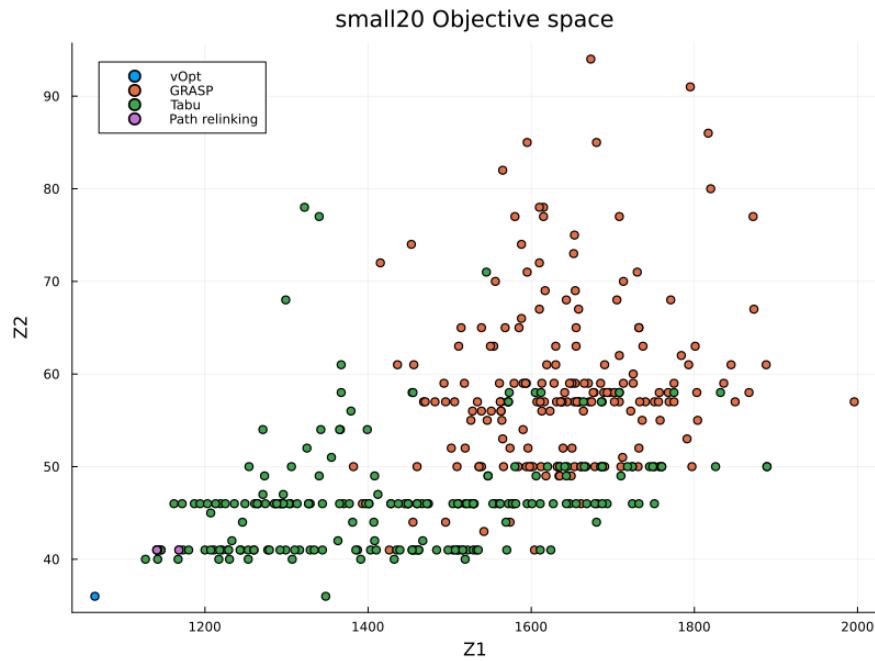


FIGURE 25 – small20 Solution space

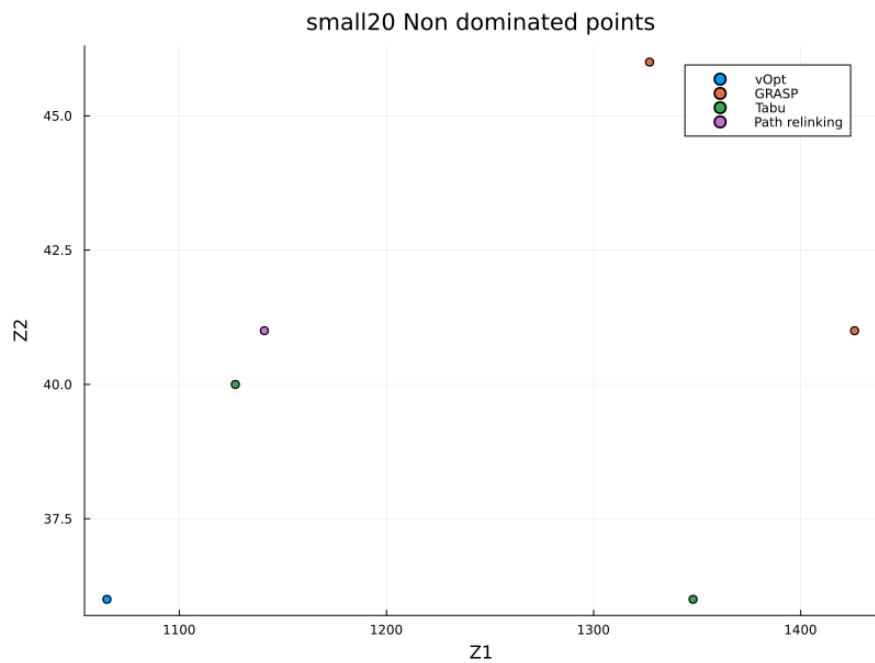


FIGURE 26 – small20 Y_N

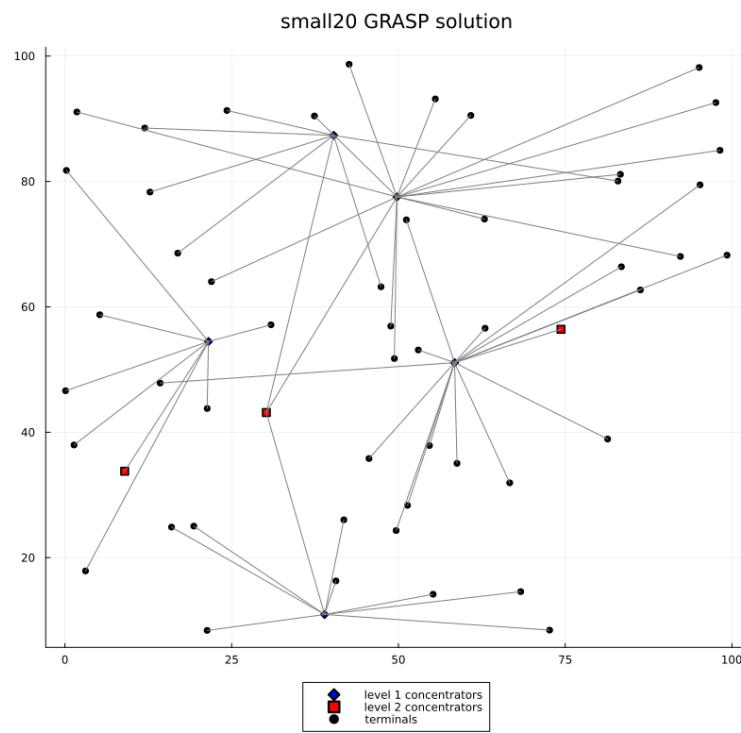


FIGURE 27 – Exemple : small20 GRASP solution

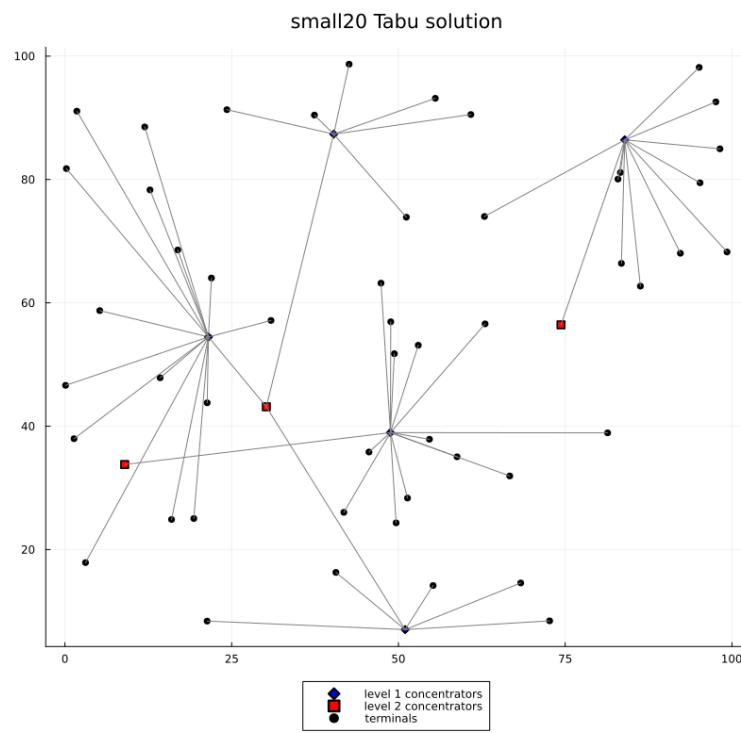


FIGURE 28 – Exemple : small20 Tabu solution

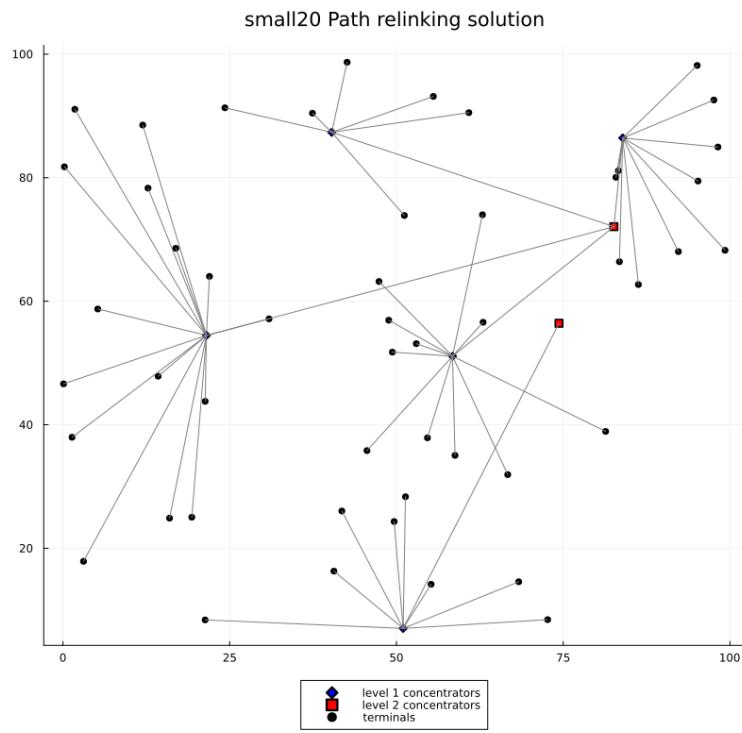


FIGURE 29 – Exemple : **small20** Path Relinking solution

A.6 medium1

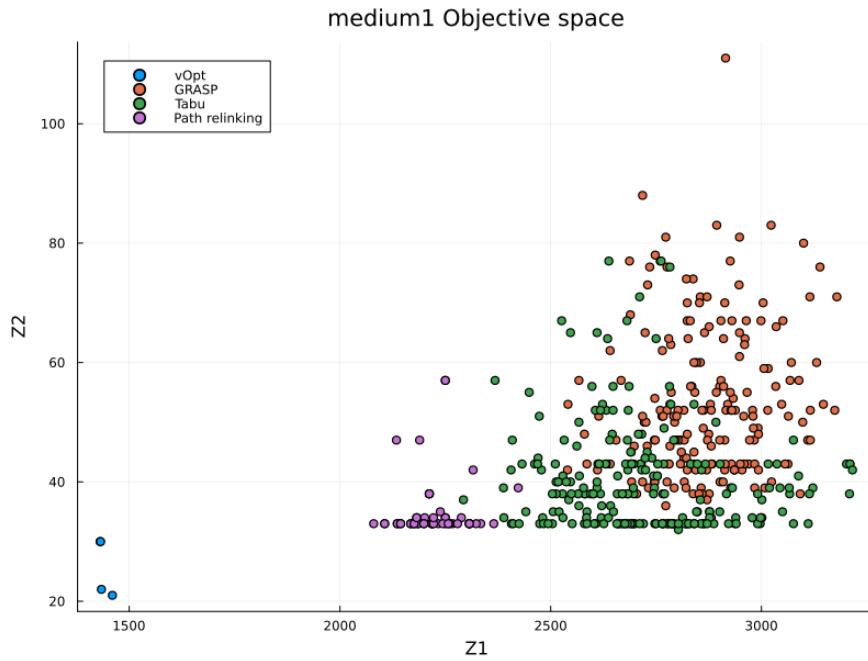


FIGURE 30 – **medium1** Solution space

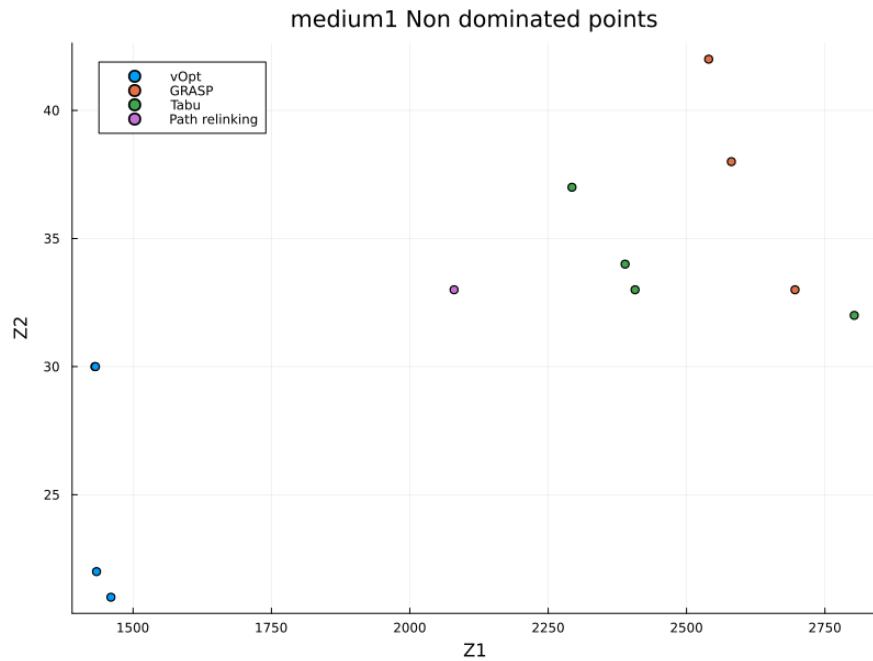


FIGURE 31 – medium1 Y_N

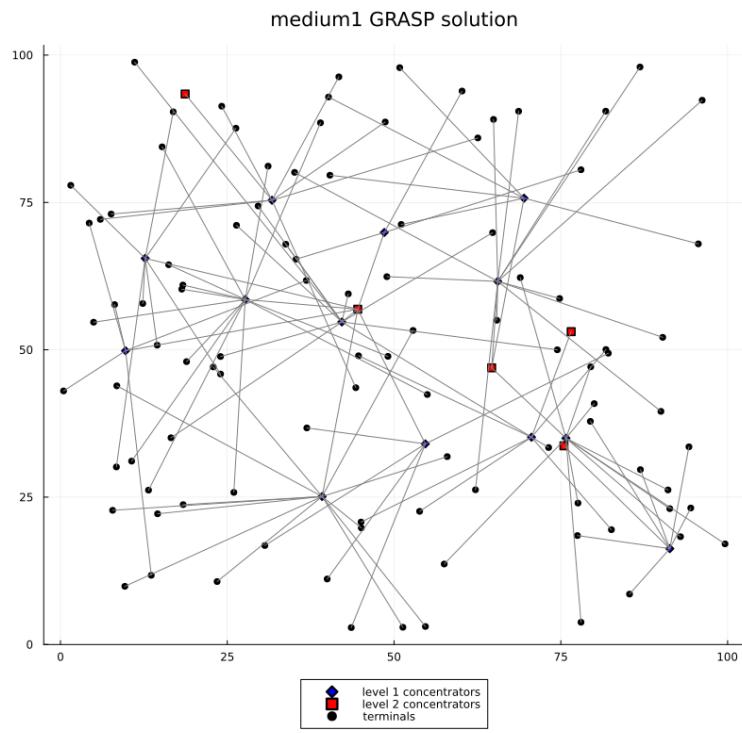


FIGURE 32 – Exemple : medium1 GRASP solution

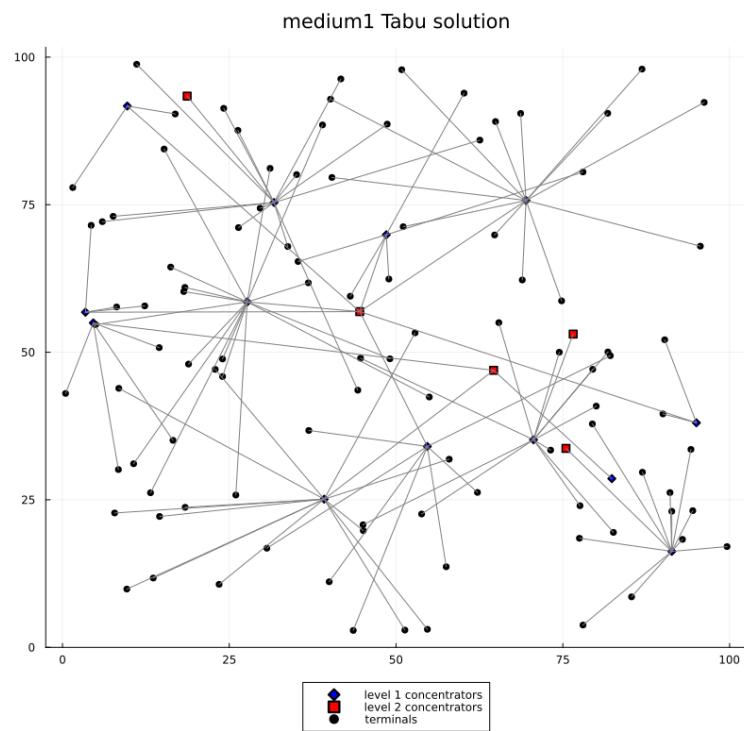


FIGURE 33 – Exemple : medium1 Tabu solution

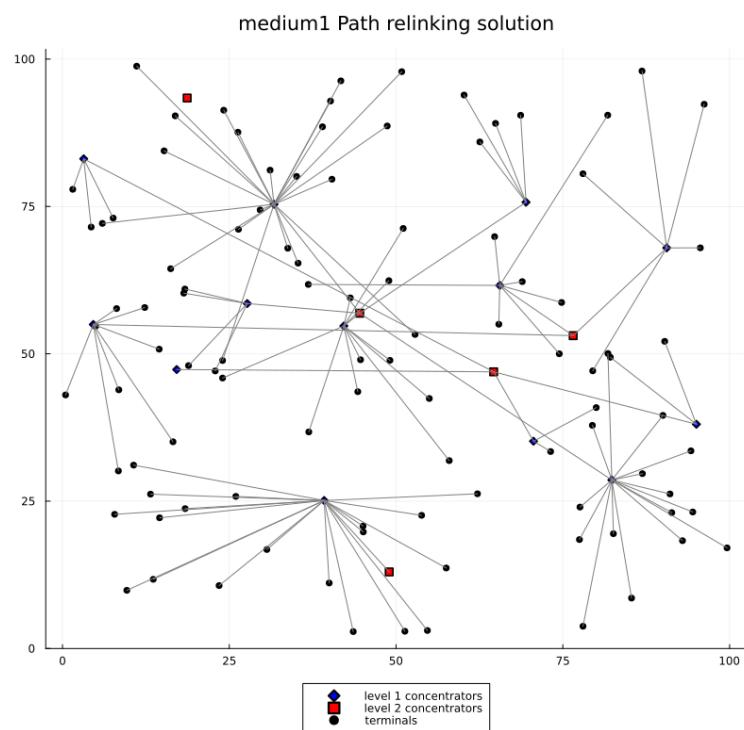


FIGURE 34 – Exemple : medium1 Path Relinking solution

A.7 medium5

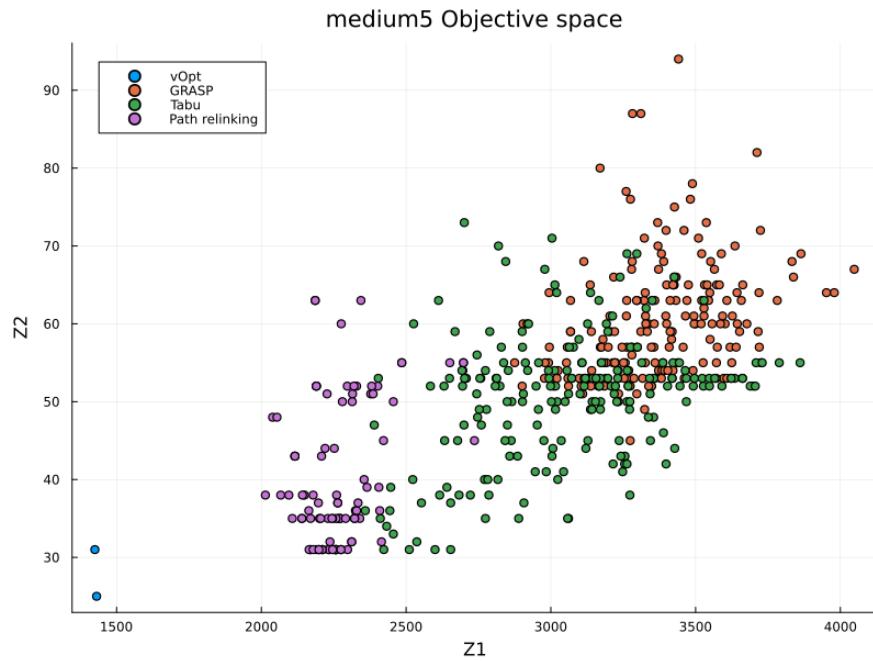


FIGURE 35 – medium5 Solution space

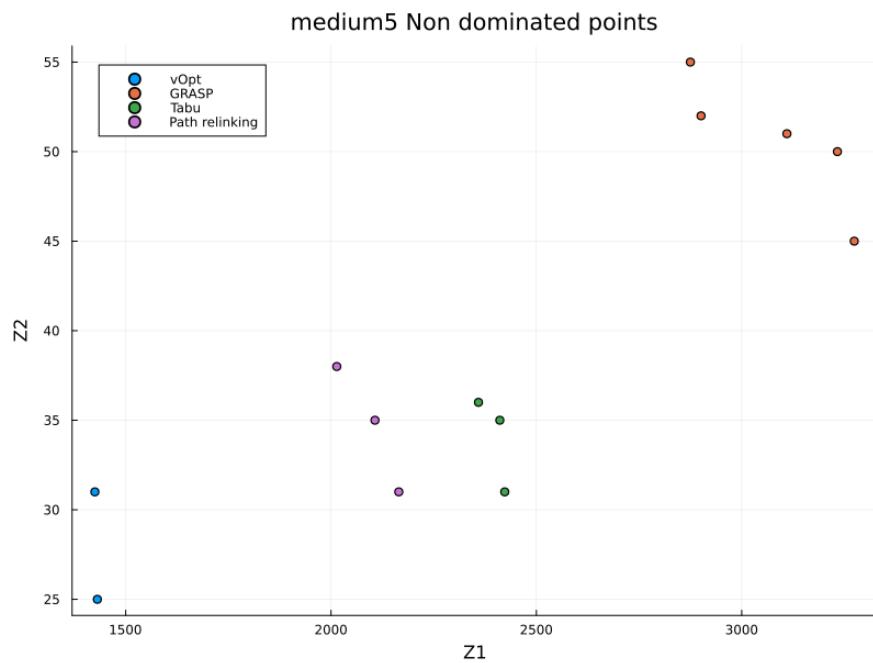


FIGURE 36 – medium5 Y_N

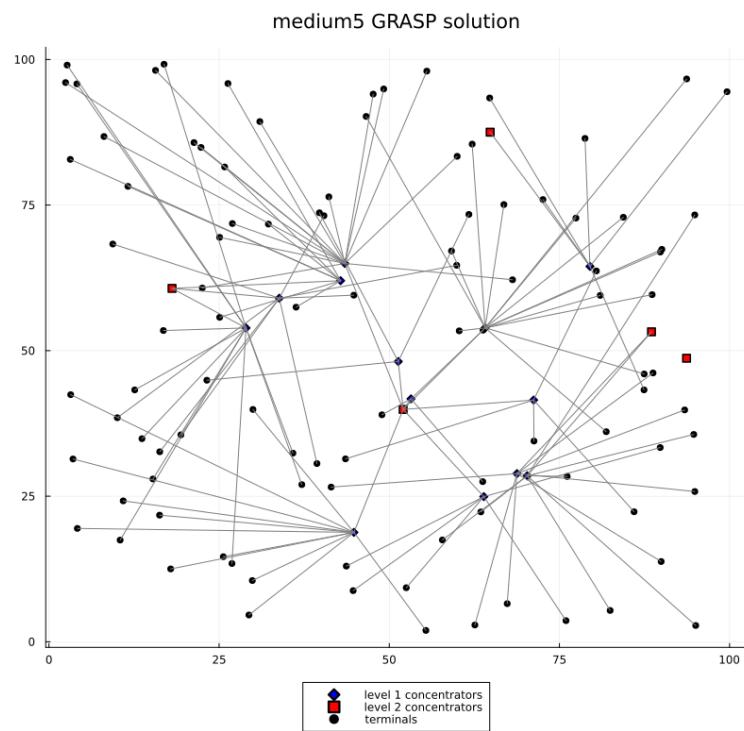


FIGURE 37 – Exemple : medium5 GRASP solution

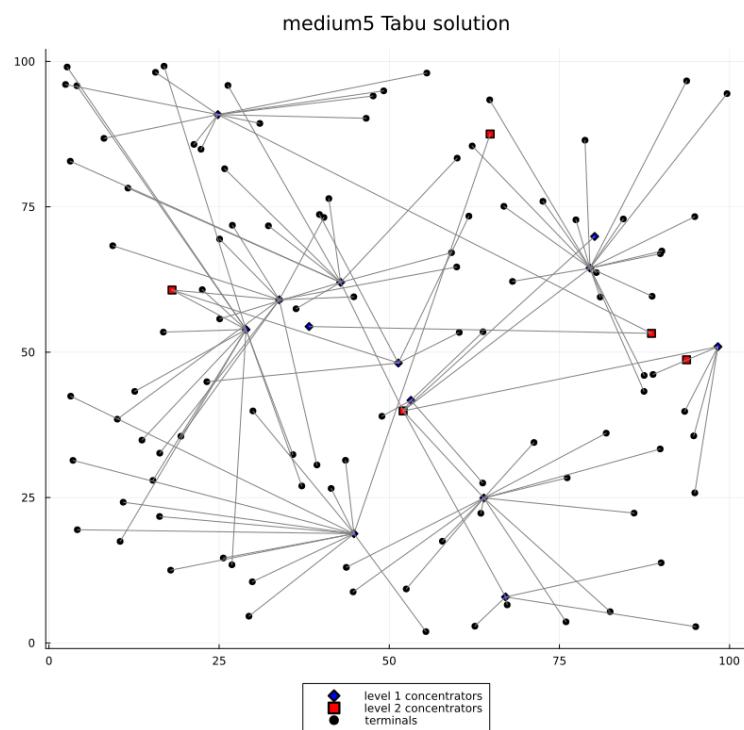


FIGURE 38 – Exemple : medium5 Tabu solution

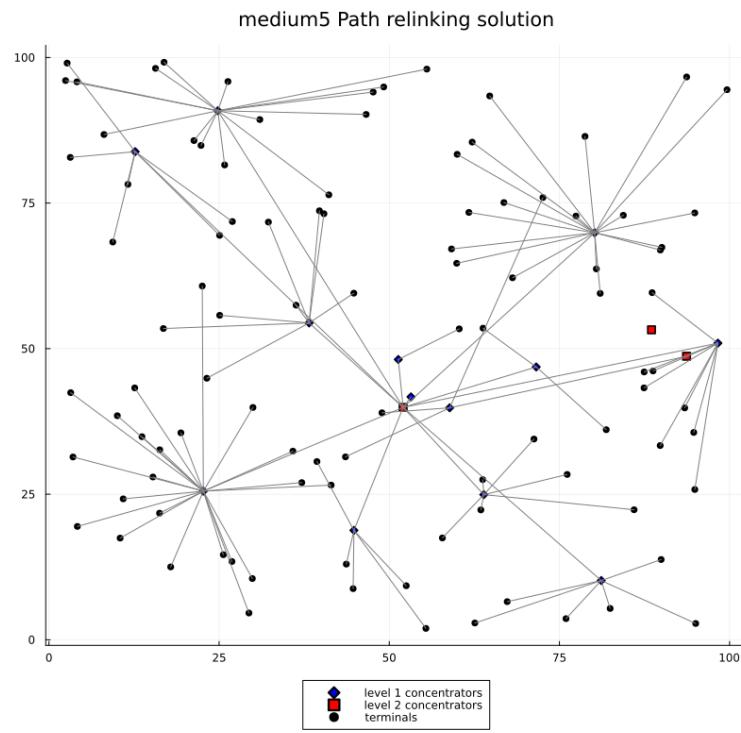


FIGURE 39 – Exemple : medium5 Path Relinking solution

A.8 medium10

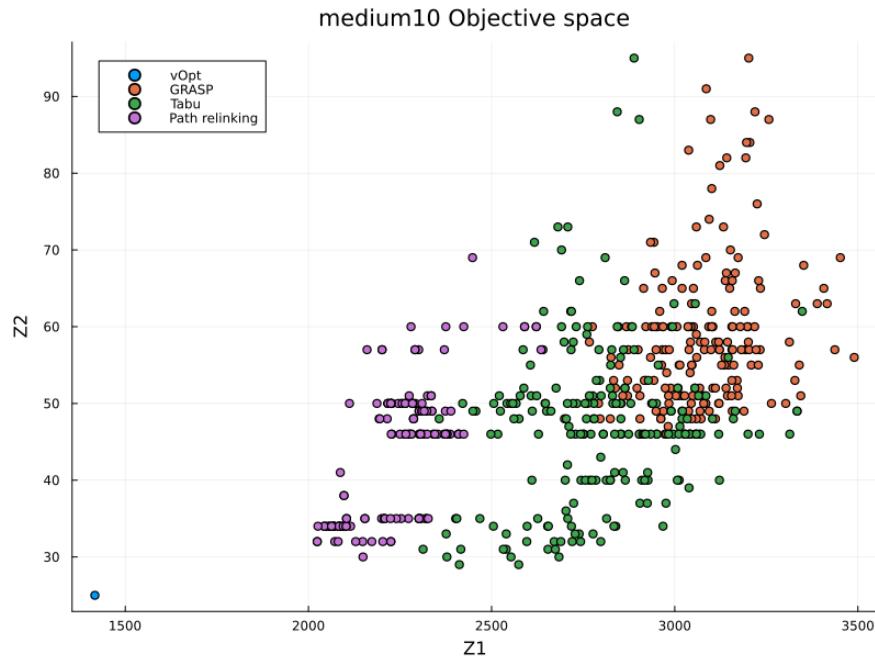


FIGURE 40 – medium10 Solution space

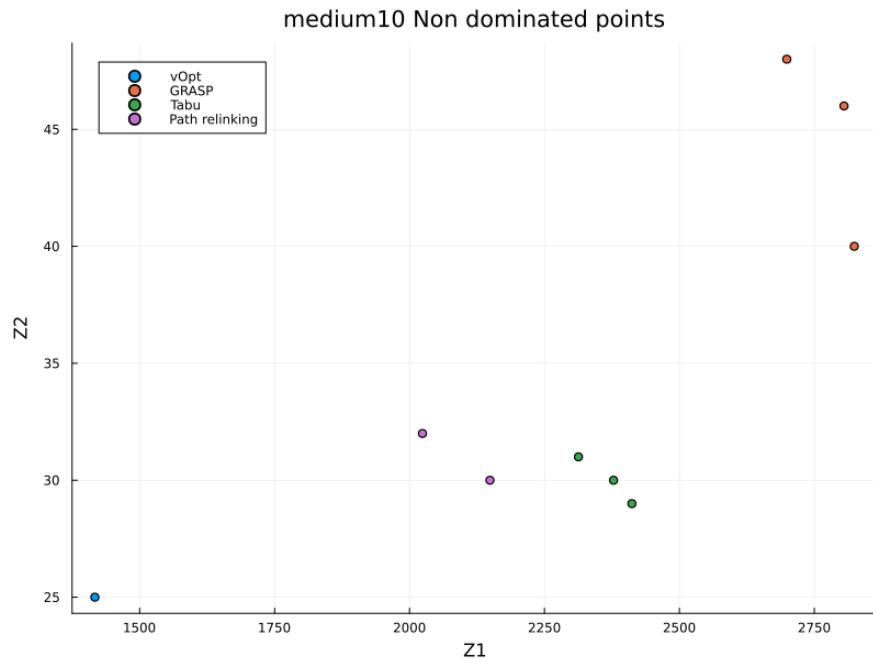


FIGURE 41 – medium10 Y_N

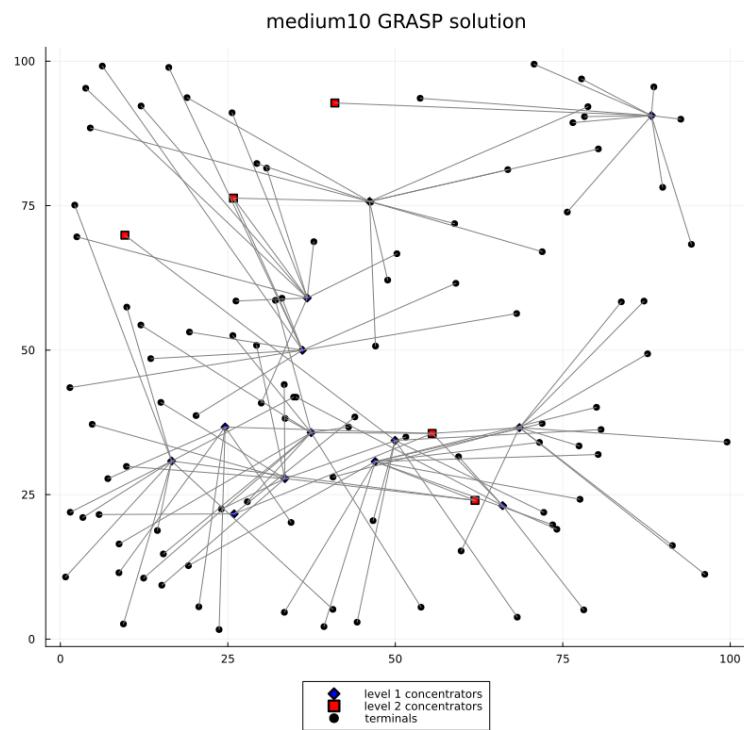


FIGURE 42 – Exemple : medium10 GRASP solution

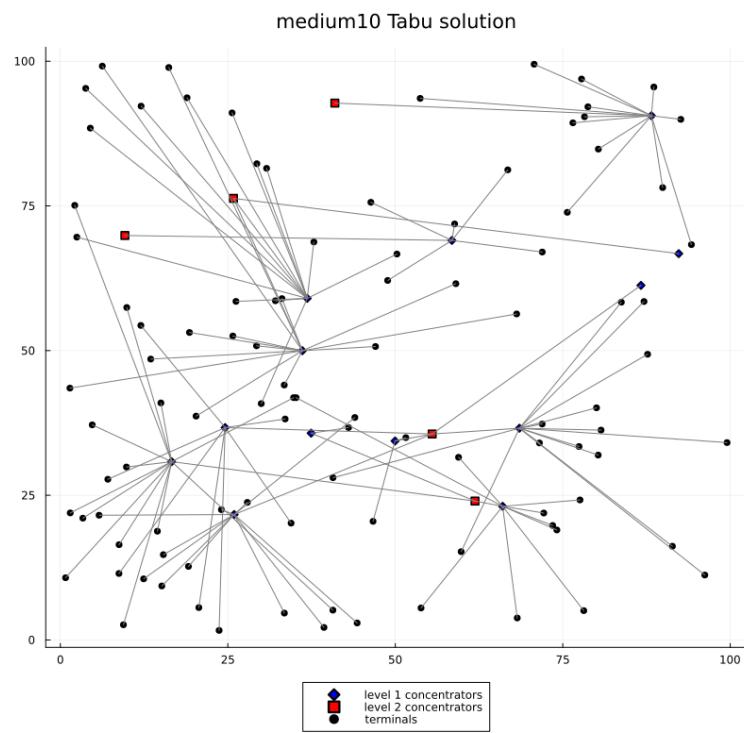


FIGURE 43 – Exemple : medium10 Tabu solution

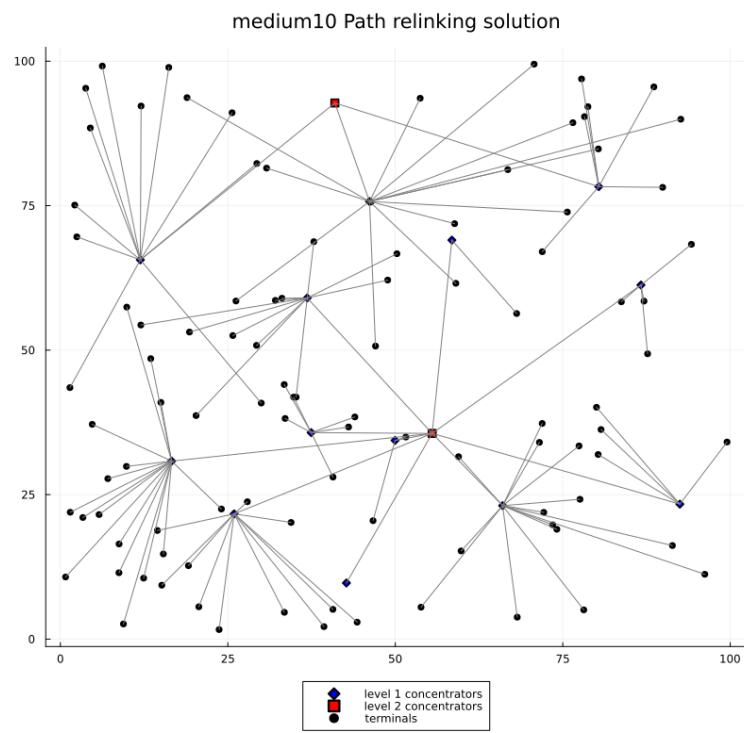


FIGURE 44 – Exemple : medium10 Path Relinking solution

A.9 large2

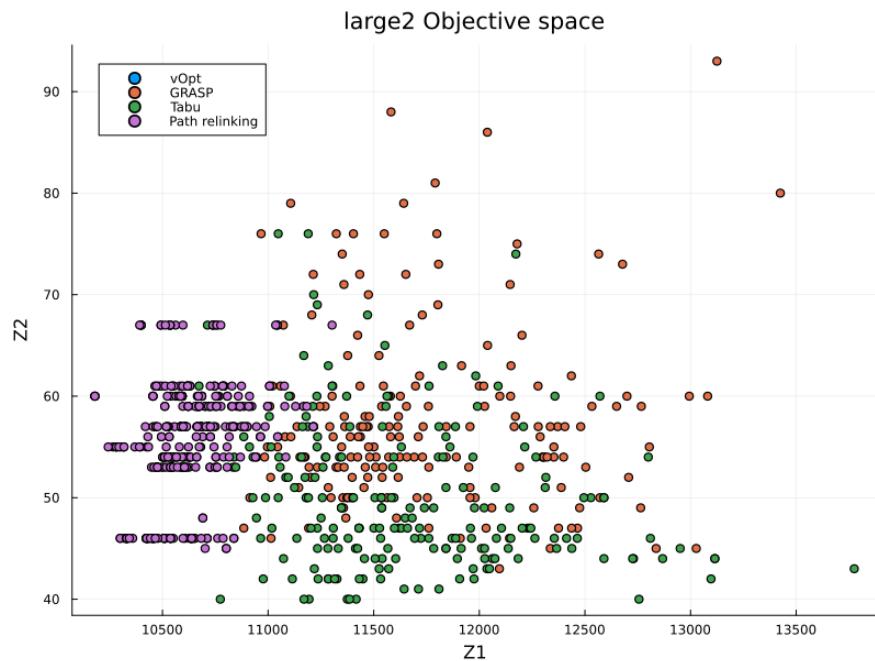


FIGURE 45 – large2 Solution space

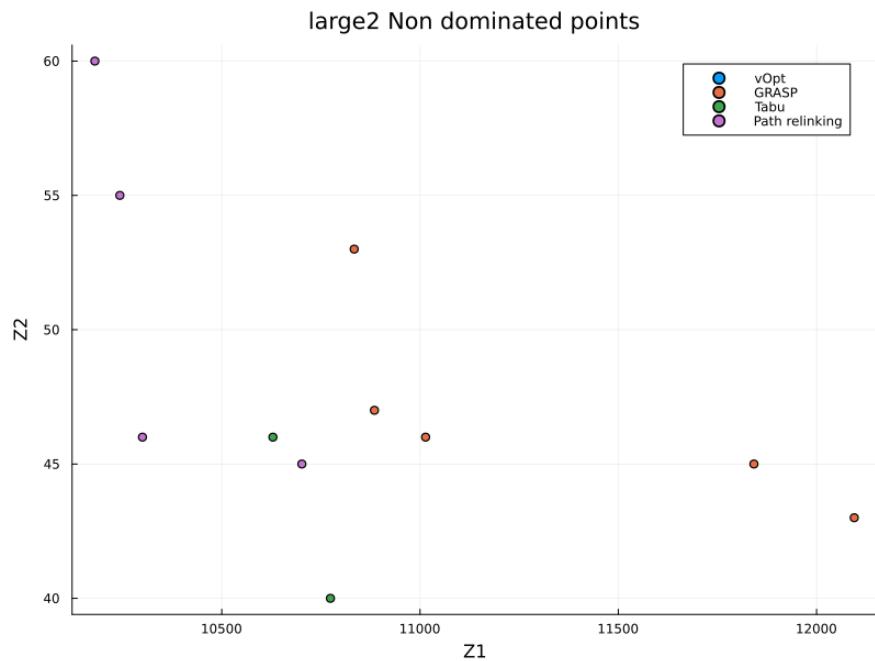


FIGURE 46 – large2 Y_N

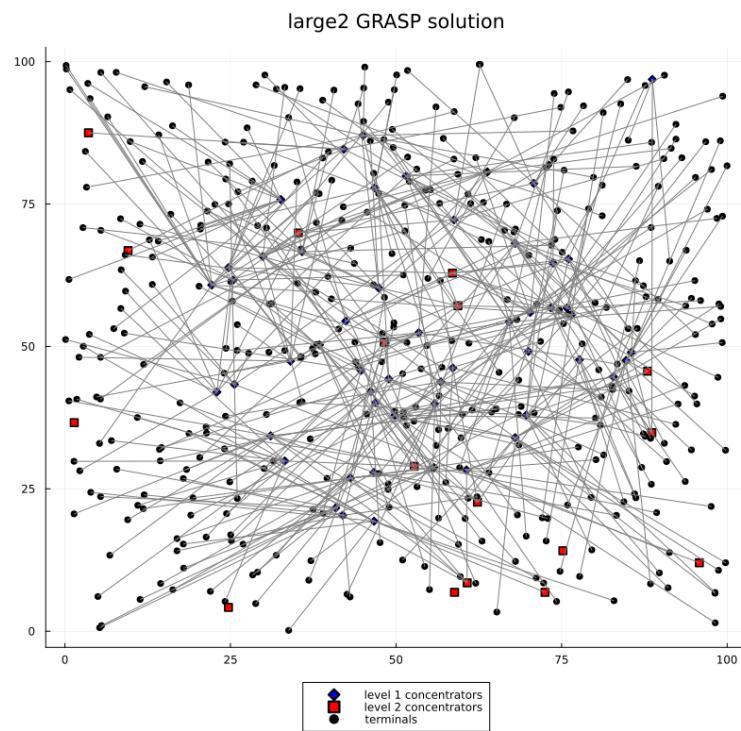


FIGURE 47 – Exemple : large2 GRASP solution

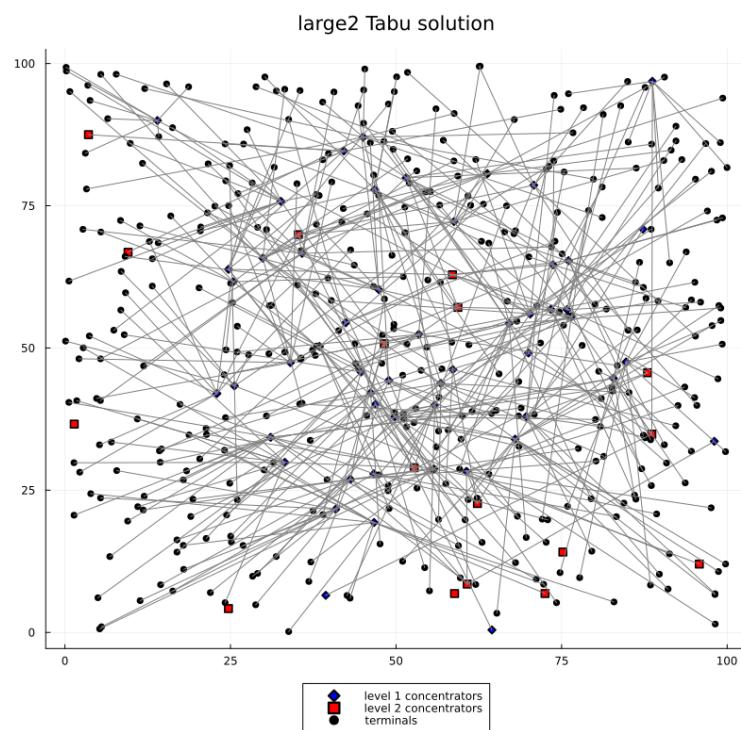


FIGURE 48 – Exemple : large2 Tabu solution

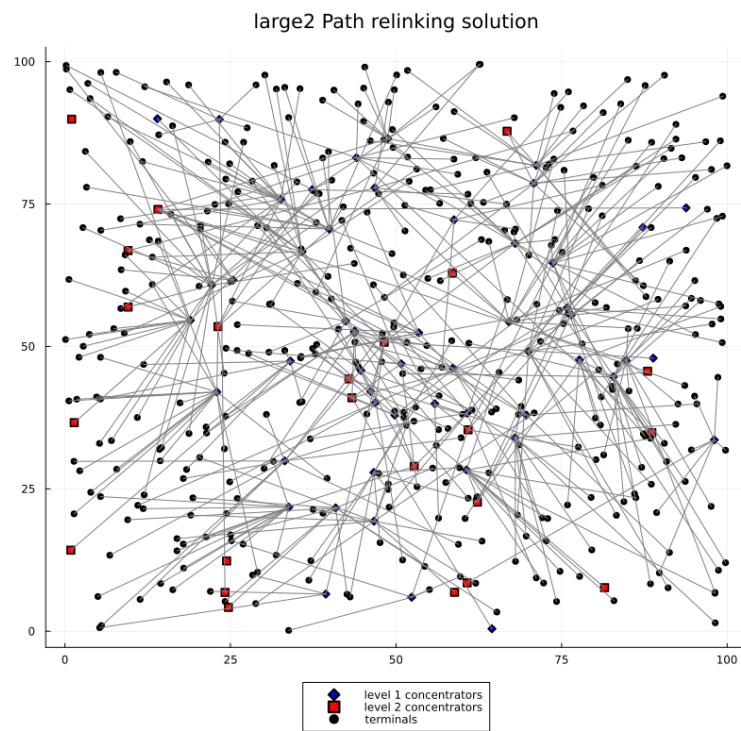


FIGURE 49 – Exemple : large2 Path Relinking solution