INF8175 : Examen final (Hiver 2024) - A. Guichard, Q. Cappart	Note finale ↓
Date : 24/04/2024 Nombre de questions : **8** Total des points : **40**	
Matricule : Nom : Temps : 2h30 heures	

Question:	1	2	3	4	5	6	7	8	Total
Points:	0	5	10	5	5	5	5	5	40
Score:									

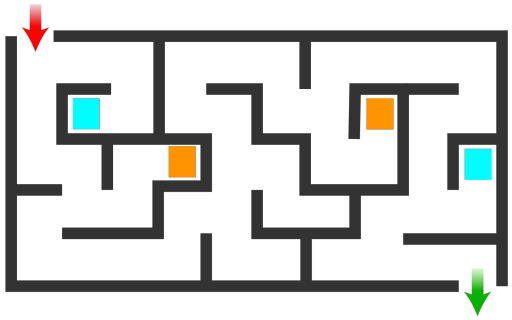
Instructions

1. La documentation papier et électronique sur clé USB est autorisée.
2. L'examen doit se réaliser de manière individuelle. Il est strictement interdit de collaborer.

Question 2 (5 points)

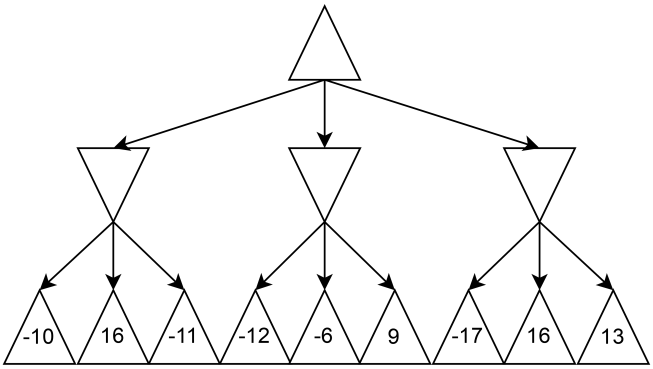
Répondez succinctement à chacune des questions en justifiant à chaque fois votre réponse. **Une bonne réponse sans justification (ou avec une une justification erronée) ne recevra aucun point.**

1. *(1 point)* Considérez la situation où un agent doit sortir d'un labyrinthe en effectuant le moins de déplacement possible. Un déplacement peut se faire soit en haut, en bas, à gauche, ou à droite. De plus, le labyrinthe comporte des *portails magiques*, permettant à l'agent de se téléporter directement à un autre endroit du labyrinthe. Par exemple, si l'agent va sur le portail orange, il sera téléporté à l'autre portail orange. Vous concevez une recherche $A^\star$ pour résoudre ce problème en utilisant la distance de Manhattan entre une position et la sortie comme heuristique. Est-ce que la recherche est optimale ?

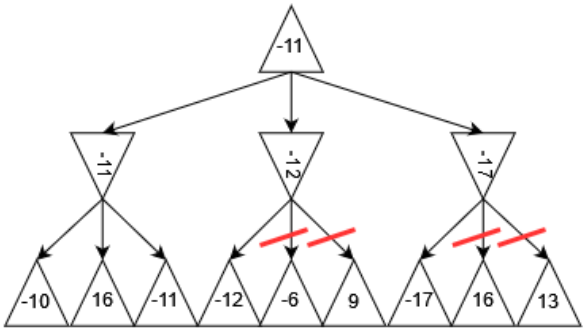


Solution: L'heuristique n'est pas admissible car elle peut surestimer le coût du trajet restant à parcourir à partir d'un point car elle ne prend pas les téléporteurs en compte. Par exemple : le valeur heuristique de la première case du labyrinthe vaut 15 alors qu'il est solvable en 11 actions en prenant le téléporteur bleu. La recherche n'est pas optimale.

2. *(1 point)* Complétez les valeurs minimax de l'arbre suivant puis indiquez les branches qui seront élaguées lors d'un élagage alpha-beta. Pour cette question, il n'est pas nécessaire de justifier votre réponse ni de préciser les bornes alpha et beta, barrez juste les branches de l'arbre.



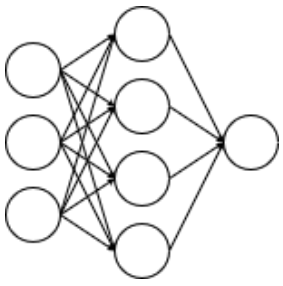
Solution:



3. *(1 point)* Le problème des n-reines consiste à placer n reines sur un échiquier de taille $n \times n$ de sorte à ce qu'aucune ne se retrouve sur la même ligne, colonne, diagonale qu'une autre. Vous souhaitez résoudre ce problème avec une recherche locale basée sur du *hill climbing*. Comme solution initiale, vous placez chaque reine aléatoire sur une colonne. Chaque colonne comporte ainsi une et une seule reine. Votre fonction de voisinage consiste à bouger une reine et à la placer à une autre case de sa colonne. Malheureusement, après l'exécution de votre recherche, vous convergez vers une solution non réalisable. Vous hésitez alors à modifier votre fonction de voisinage pour bouger deux reines au lieu d'une seule lors d'un mouvement local. Donnez un avantage et un inconvénient de cette idée.

Solution: Par exemple :
- **Avantage** : Un voisinage plus large peut permettre de "voir plus loin" et sortir d'un minimum local.
- **Inconvénient** : Agrandir le voisinage est considéré comme une méthode d'intensification qui amène dans de moins bons minimums locaux. Autre possibilité : un voisinage plus large est plus long à parcourir.

4. *(1 point)* Considérez le réseau de neurone ci-dessous. La première couche représentée indique le nombre de caractéristiques et la dernière couche indique une prédiction d'une seule valeur pour une classification binaire. De plus, la fonction *tanh* est utilisée comme activation non-linéaire de la couche cachée. Combien de paramètres à apprendre comporte ce réseau ?



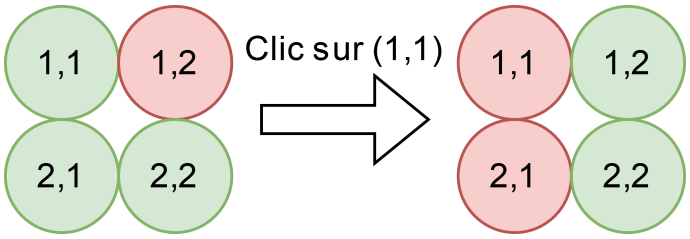
Solution: 16 poids et 5 biais soit 21 paramètres.

5. *(1 point)* Vous souhaitez identifier les tentatives de *phishing* dans vos mails via un apprentissage non-supervisé. Pour cela, vous relevez 2 caractéristiques : le nombre de fautes dans le mail ainsi que le nombre de liens présents dans le mail. Combien de paramètres devriez-vous apprendre si vous utilisez un modèle basé sur une loi normale multi-variée ?

Solution: 2 paramètres pour l'espérance, $2 + \frac{2^2-2}{2} = 3$ paramètres pour la matrice de variance-covariance soit 5 paramètres au total.

Question 3 (10 points)

Johnny est bon chanteur mais mauvais électricien, les lampes de ses 4 pièces sont reliées de sorte à ce que allumer ou éteindre une lampe, va basculer l'état des lampes adjacentes (en haut, en bas, à gauche, et à droite). Ainsi en éteignant la lampe en haut à gauche (1,1) celle en bas à gauche (2,1) va s'éteindre aussi, mais la lampe en haut à droite (1,2) va s'allumer ! Après un concert à s'en casser la voix, il aimerait bien se coucher sans avoir à courir dans toutes les pièces de la maison. Le problème à résoudre consiste à éteindre toutes les lumières en effectuant le moins d'actions.



Formellement, la maison est représentée par une grille de largeur n et de longueur n. Chaque case $c_{i,j}$ contient une lampe qui peut être soit allumée (1) ou éteinte (0). Basculer l'état d'une case basculera l'état des cases adjacentes. Ainsi, basculer la case $c_{i,j}$ donnera donc :

$$
\begin{aligned}
c_{i,j} &= 1 - c_{i,j} \\
c_{i+1,j} &= 1 - c_{i+1,j} \\
c_{i-1,j} &= 1 - c_{i-1,j} \\
c_{i,j+1} &= 1 - c_{i,j+1} \\
c_{i,j-1} &= 1 - c_{i,j-1}
\end{aligned}
$$

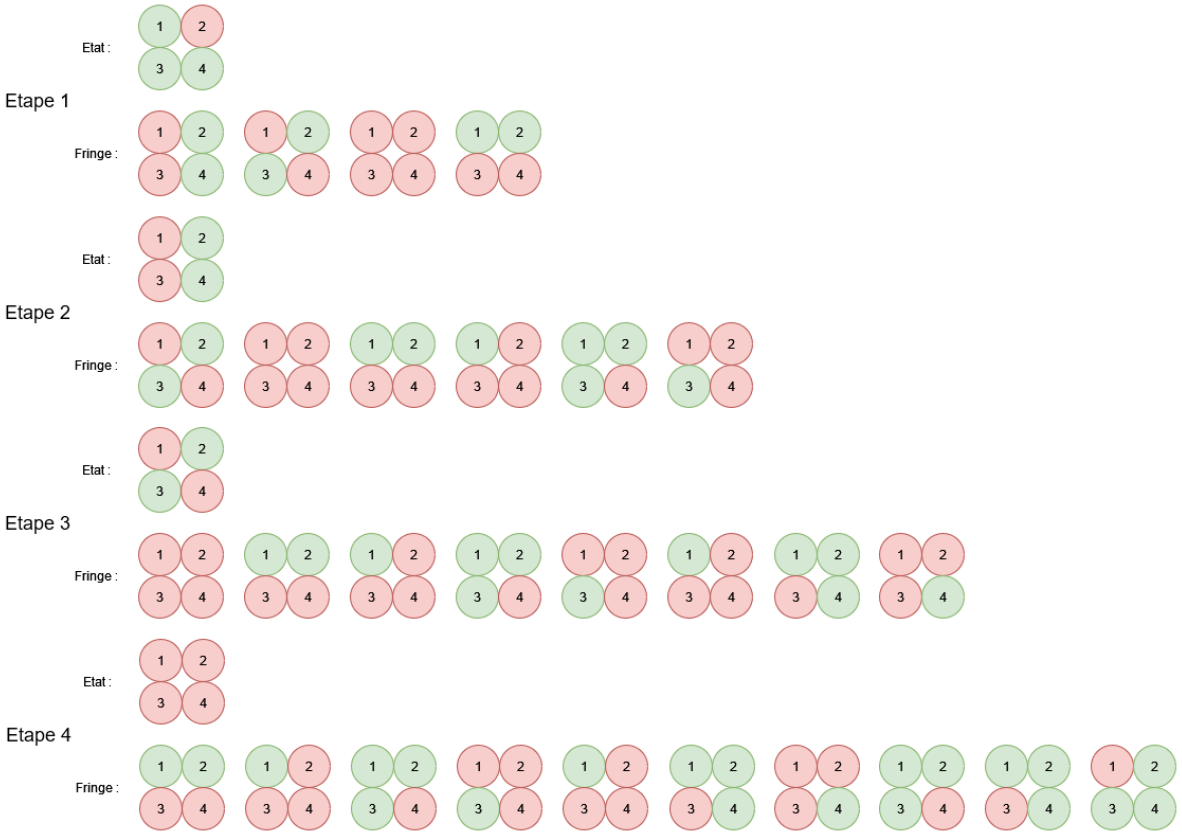
1. *(2 points)* Formalisez ce problème comme un *problème de recherche* en identifiant bien quels sont les états, les actions, la fonction de transition, et la fonction de coût de votre modèle. Il n'est pas nécessaire de donner une formalisation mathématique stricte. L'important est que le correcteur comprenne clairement vos différentes entités.
2. *(1 point)* Donnez le nombre d'états de votre formulation, ainsi que le nombre d'actions qui sont possibles à un état (*branching factor*). Exprimez votre solution en fonction de la taille de la grille ($n \times n$).
3. *(4 points)* Déterminez la séquence d'actions qui sera obtenue en exécutant une *recherche en largeur* à partir de l'état présenté plus haut : $\langle c_{1,1} = 1, c_{1,2} = 0, c_{2,1} = 1, c_{2,2} = 1\rangle$. Considérez que les états sont ajoutés dans la *fringe* dans l'ordre croissant de bascule (on ajoute d'abord l'état où on a basculé la cellule $c_{1,1}$, puis celui où on a basculé $c_{1,2}$, puis $c_{2,1}$ puis $c_{2,2}$.) Indiquez chaque étape de l'algorithme. À chaque étape, veillez à bien indiquer votre état actuel, ainsi que les nœuds présents dans la liste des états candidats (*fringe*). Considérez une *recherche en graphe* (et non en arbre). Précisément, un état ne sera pas ajouté dans la liste des états candidats (*fringe*) si et seulement s'il a déjà été visité. Afin d'éviter de vous faire dessiner un grand nombre de schémas, un patron d'exécution vous est mis disponible aux pages suivantes. Vous pouvez représenter un état seulement par une représentation visuelle. Barrez les cases qui ne sont pas nécessaires.
4. *(1 point)* Combien d'actions doit faire Johnny pour éteindre toutes les pièces avec cette solution ?
5. *(1 point)* Avez-vous la garantie que cette solution est optimale ? Justifiez.
6. *(1 point)* Proposez une heuristique pour ce problème. Indiquez si elle est admissible ou non (avec justification).

Solution:

1. Proposition d'une formulation (des variations sont possibles) :
   — **Etat** : un tuple représentant l'état de chaque cellule 0 ou 1.

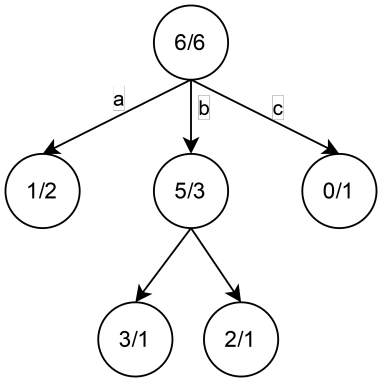
— **Action** : basculer l'etat d'une des cellule
— **Fonction de transition** : basculer l'état des cellules adjacentes
— **Fonction de coût** : incrément de 1 pour chaque mouvement.

2. Chaque cellule peut etre allumée ou éteinte. Il y a donc 2^{n*n} états. Chaque cellule peut être basculée, il y a $n * n$ cellules donc $n * n$ actions possibles.
3. Notez bien qu'on exécute une recherche en graphe, et que dès lors, on retient les états déjà vus. Attention à ne pas confondre le nombre d'états visités et le coût de la solution retournée. L'exécution est montrée plus bas.
4. Johnny doit faire 2 actions : basculer l'état de la pièce 3 puis celle de la pièce 4.
5. Une recherche en largeur à coût unitaire s'apparente à une exécution UCS. La solution trouvée est optimale.
6. Une heuristique possible est le nombre de pièce encore allumées. Celle-ci n'est pas admissible car une action peut éteindre plusieurs pièce du fait de la nature du problème. Pour trouver une heuristique admissible, il faudrait prendre en compte les lumières allumées adjacentes. Elle est plus difficile à définir.



Question 4 (5 points)

Soit l'arbre de recherche MCTS partiellement construit suivant :



Les valeurs (x/y) de chaque noeud n indiquent respectivement le score total cumulé du noeud via les simulations $(U(n))$ et le nombre total de simulation ayant impliqué le noeud $(N(n))$. La règle de sélection utilisée pour la phase de sélection est la suivante :

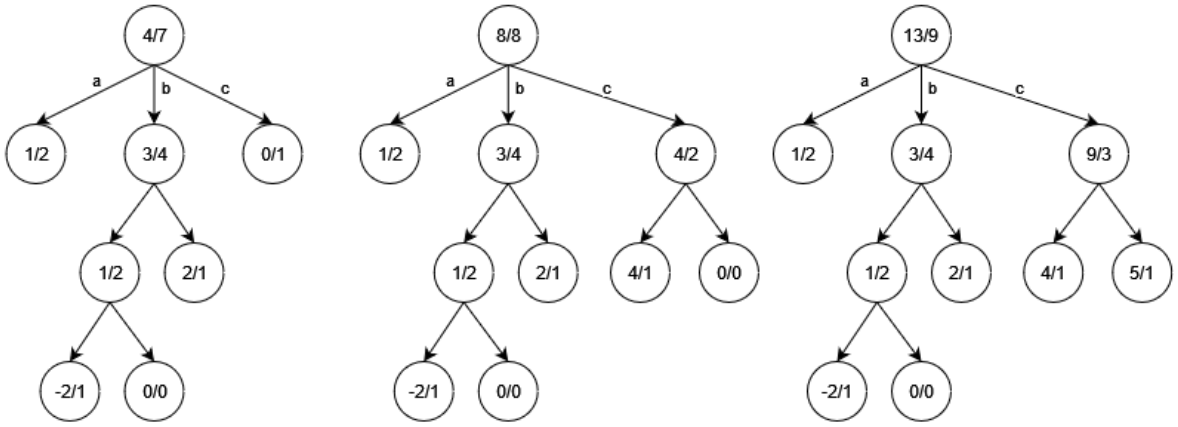
$$UCB1(n) = \frac{U(n)}{N(n)} + 2\sqrt{\frac{lnN(Parent(n))}{N(n)}} \tag{1}$$

1. *(3 points)* Réalisez 3 itérations complètes de l'algorithme MCTS sur cet arbre. C'est à dire que vous devez réaliser les étapes de sélection, d'expansion, de simulation, et de backpropagation 3 fois. Indiquez après chaque itération l'arbre mis-à-jour. Pour la phase d'expansion, supposez que deux nouveaux noeuds sont générés, et que celui utilisé pour la simulation est celui de gauche. Pour les simulations, considérez que les scores suivants sont obtenus : -2, 4, et puis 5. Des arbres partiels vous sont fournis pour accélérer la réalisation de cet exercice. Barrez ceux que vous n'utilisez pas. Pour vous aider dans cette tache, une table est fournie pour calculer les valeurs `UCB1` des noeuds plus rapidement. À titre d'exemple, la valeur `UCB1` d'un noeud $(-1/3)$ ayant un parent $(x/6)$ est 1.212.
2. *(1 point)* À l'issue de ces simulations, quelle action le joueur actuel devrait jouer (a, b, ou c) ?
3. *(1 point)* Donnez 2 raisons d'utiliser MCTS plutôt qu'un algorithme minimax avec élagage alpha-beta dans le cas d'une stratégie de recherche adversarielle.

$U(n)$ / $N(n)$ \ $N(Parent(n))$	1	2	3	4	5	6	7	8
-2 / 1	-2.0	-0.335	0.096	0.355	0.537	0.677	0.79	0.884
-2 / 2	-1.0	0.177	0.482	0.665	0.794	0.893	0.973	1.039
-1 / 3	-0.333	0.628	0.877	1.026	1.132	1.212	1.277	1.332
0 / 0	inf	inf	inf	inf	inf	inf	inf	inf
0 / 1	0.0	1.665	2.096	2.355	2.537	2.677	2.79	2.884
0 / 2	0.0	1.177	1.482	1.665	1.794	1.893	1.973	2.039
1 / 2	0.5	1.677	1.982	2.165	2.294	2.393	2.473	2.539
2 / 1	2.0	3.665	4.096	4.355	4.537	4.677	4.79	4.884
3 / 1	3.0	4.665	5.096	5.355	5.537	5.677	5.79	5.884
3 / 4	0.75	1.583	1.798	1.927	2.019	2.089	2.145	2.192
4 / 1	4.0	5.665	6.096	6.355	6.537	6.677	6.79	6.884
4 / 2	2.0	3.177	3.482	3.665	3.794	3.893	3.973	4.039
4 / 3	1.333	2.295	2.544	2.693	2.798	2.879	2.944	2.998
5 / 2	2.5	3.677	3.982	4.165	4.294	4.393	4.473	4.539
5 / 3	1.667	2.628	2.877	3.026	3.132	3.212	3.277	3.332
6 / 2	3.0	4.177	4.482	4.665	4.794	4.893	4.973	5.039
7 / 5	1.4	2.145	2.337	2.453	2.535	2.597	2.648	2.69

Solution:

1. Attention, la dernière itération n'a pas de phase d'extension !
2. Il choisira l'action b car c'est celle qui a été le plus simulé (N le plus grand)
3. MCTS peut être plus avantageux si une heuristique est difficile à définir ou si le facteur de branchement est grand (D'autres réponses sont possibles).

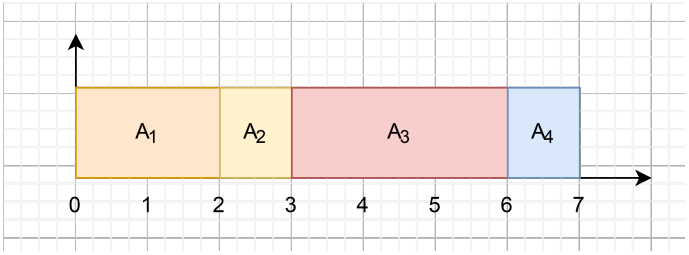


Question 5 (5 points)

Johnny est bon chanteur mais mauvais négociateur. Arnaqué par ses labels, il doit produire plus d'albums qu'il ne le peut. Chaque album A_i nécessite un temps d'écriture T_i et entraîne une pénalité de retard P_i pour chaque jour passé après une deadline D_i. Une situation avec 4 albums vous est présentée dans le tableau ci-dessous.

Album	A_1	A_2	A_3	A_4
Temps d'écriture	2	1	3	1
Deadline	4	1	2	2
Pénalité	1	3	5	3

Johnny ne peut pas écrire plusieurs albums en même temps. Une solution possible est de commencer l'écriture de l'album 1 au temps 0, de l'album 2 au temps 2, de l'album 3 au temps 3, et de l'album 4 au temps 6. Une représentation visuelle de cette solution vous est présentée ci-dessous.



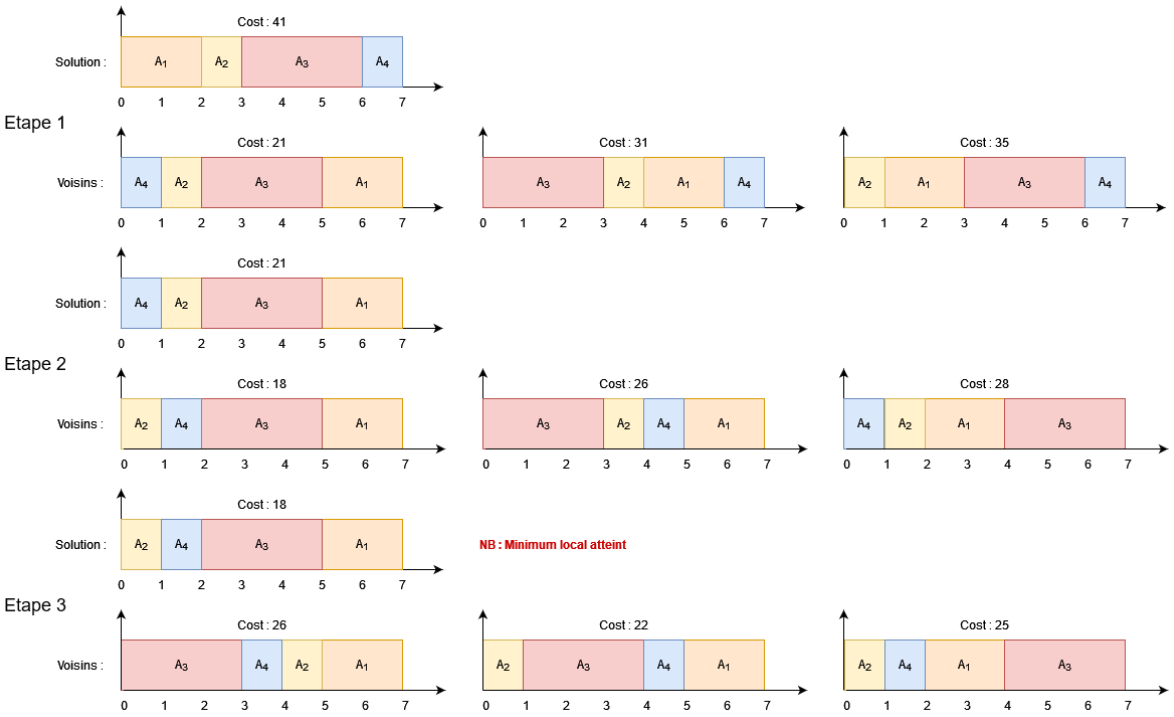
Dans cette solution, les albums 1, 2, 3 et 4 sont en retard de respectivement 0, 2, 4 et 5 jours donnant ainsi une somme des pénalités à $0 \times 1 + 2 \times 3 + 4 \times 5 + 5 \times 3 = 41$. Johnny aimerait trouver dans quel ordre écrire ses albums afin de minimiser les pénalités de retard à payer aux labels. Pour cette question, il ne vous est pas demandé de résoudre complètement ce problème, mais simplement de proposer une résolution basée sur la recherche locale et d'exécuter quelques étapes de l'algorithme.

1. *(1 point)* Proposez un modèle de recherche locale en vue de résoudre ce problème. En particulier, veillez à bien définir les éléments suivants :
   (a) Votre espace de recherche. Indiquez le plus précisément possible quelle est sa taille.
   (b) Une méthode pour obtenir une solution initiale.
   (c) Une fonction d'évaluation pour caractériser la qualité d'un état.
   (d) Une fonction de voisinage. Indiquez le plus précisément la taille de celui-ci.
2. *(3 points)* Exécutez 2 itérations de l'algorithme du *hill climbing* en utilisant comme solution initiale la solution présentée plus haut et comme fonction de voisinage et d'évaluation celles proposées en (1). Vous ne devez pas indiquer tous les voisins de votre voisinage, prenez en juste 3 au maximum (qui sont cohérents avec la définition de votre voisinage), et faites la sélection parmi ces 3 voisins. Des grilles vous sont fournies pour dessiner plus bas, n'oubliez pas d'indiquer la valeur de votre fonction d'évaluation à chaque itération.
3. *(1 point)* Quelle est le risque d'utiliser un algorithme de type *hill climbing* ? Comment mitiger ce risque ?

Solution:

1. Proposition d'une formulation (des variations sont possibles) :
   — **Espace de recherche** : L'ensemble des séquences (permutations des albums) possible. Il est de l'ordre de $n!$ où n est le nombre d'albums à écrire. Ici il est donc de taille $4! = 24$.
   — **Solution initiale** : aléatoire ou dans l'ordre croissant des deadlines ou tout autre fonction prenant en compte les deadlines, pénalités et/ou durée.
   — **Fonction d'évaluation** : La somme pondérée des pénalités.
   — **Fonction de voisinage** : 2-SWAP (inversion de la position de 2 albums) de taille 6 ($\sum_i^{n-1} i = \frac{n*(n-1)}{2}$). Ou encore insertion d'un album à un autre endroit de la séquence de taille 6 aussi (le calcul est le même).

2. Nous utiliserons le voisinage 2-SWAP pour cet exemple.
3. L'algorithme du *hill climbing* tombe facilement dans des minimums locaux, l'ajout de métaheuristiques (simulated annealing, tabu search, algorithmes génétiques, etc.) ou de restarts peut mitiger cela.



Question 6 (5 points)

Soit le modèle de problème combinatoire suivant. La contrainte $\texttt{Odd}(x)$ force la variable x à prendre une valeur impaire.

$$
\begin{array}{lll}
\text{satisfy} & & \\
\text{subject to} & x^2 = y & (c_1) \\
& y + z \leq 8 & (c_2) \\
& \texttt{Odd}(x) & (c_3) \\
& x, y, z \in \{1, \ldots, 10\} &
\end{array}
$$

1. *(0.5 point)* Indiquez la taille de l'espace de recherche si on souhaite réaliser une recherche exhaustive.
2. *(4 points)* Appliquez l'algorithme du point fixe pour réduire au maximum les domaines de chaque variable. Indiquez les domaines résultants après chaque propagation.
3. *(0.5 point)* Indiquez la taille restante de l'espace de recherche.

Solution:

1. La taille de l'espace de recherche est de $10 * 10 * 10 = 1000$
2. — file de propagation initiale : $\{c_1, c_2, c_3\}$
   — propagation de c_1 : $x = \{1, 2, 3\}, y = \{1, 4, 9\}, z = \{1, ..., 10\}$, file de propagation : $\{c_2, c_3\}$
   — propagation de c_2 : $x = \{1, 2, 3\}, y = \{1, 4\}, z = \{1, ..., 7\}$, file de propagation : $\{c_3, c_1\}$
   — propagation de c_3 : $x = \{1, 3\}, y = \{1, 4\}, z = \{1, ..., 7\}$, file de propagation : $\{c_1\}$
   — propagation de c_1 : $x = \{1\}, y = \{1\}, z = \{1, ..., 7\}$, file de propagation : $\{c_2\}$
   — propagation de c_2 : $x = \{1\}, y = \{1\}, z = \{1, ..., 7\}$, file de propagation : $\{\}$
3. Après l'exécution de l'algorithme du point fixe, la taille de l'espace de recherche est de $1 * 1 * 7 = 7$

Question 7 (5 points)

Passionné de *myrmécologie* (l'étude des fourmis), vous décidez d'implémenter un agent logique pour vous aider à identifier l'espèce des fourmis que vous étudiez. L'idée est simple, grâce à internet vous construisez une base de connaissance, ensuite, lorsque vous trouverez une fourmi à identifier, vous ajouterez temporairement les caractéristiques de celle-ci (opération TELL) puis questionneriez votre agent pour connaître l'espèce (opération ASK). La base de connaissances que vous bâtissez est la suivante.

$$\begin{aligned} TailleDiff &\Rightarrow Caste \\ (Caste \wedge Graine) &\Rightarrow Messor \\ (Caste \wedge Miellat) &\Rightarrow Camponotus \\ (TeteRouge \wedge Messor) &\Rightarrow MessorBarbarus \\ (CorpsNoir \wedge Messor) &\Rightarrow MessorCapitatus \\ (CorpsRouge \wedge Pique) &\Rightarrow Myrmica \end{aligned}$$

Pour information, les symboles *Graine* et *Miellat* réfèrent au régime alimentaire (le miellat étant un liquide sucré). Le symbole *Caste* réfère à la présence de différentes ouvrières spécialisées dans la colonie (Major, minor). Le symbole *Pique* réfère au fait que certaines fourmis sont urticantes. Les symboles *TeteRouge*, *CorpsRouge*, *CorpsNoir* réfèrent aux caractéristiques physiques des fourmis. Les autres symboles réfèrent à l'espèce spécifique d'une fourmi. Cette base de connaissance est évidemment simplifiée pour la question.

1. *(3 points)* Au détour d'un chemin, vous trouvez une colonie qui possède des fourmis de différentes tailles, avec une tête rouge et qui transportent des graines. Vous souhaitez identifier l'espèce. Pour cela, vous ajoutez les formules suivantes à votre base de connaissances : $TailleDiff$, $Graines$, et $TetesRouges$. Vous voulez vérifier si les fourmis observées sont des $MessorBarbarus$. Appliquez itérativement la règle d'inférence suivante pour confirmer cette hypothèse.

$$\frac{p_1, \ldots, p_n \quad (p_1, \wedge \cdots \wedge p_n) \Rightarrow q}{q}. \tag{2}$$

2. *(1 point)* Est-ce que cet algorithme d'inférence est *cohérent* pour cette base de connaissance ? Justifiez.
3. *(1 point)* Est-ce que cet algorithme d'inférence est *complet* pour cette base de connaissance ? Justifiez.

Solution:

1. —
$$\frac{TailleDiff \quad TailleDiff \Rightarrow Caste}{Caste} \tag{3}$$
—
$$\frac{Caste, Graine \quad (Caste \wedge Graine) \Rightarrow Messor}{Messor} \tag{4}$$
—
$$\frac{Messor, TetesRouges \quad (Messor \wedge TetesRouges) \Rightarrow MessorBarbarus}{MessorBarbarus} \tag{5}$$
— On génère le littéral $MessorBarbarus$, La fourmi est bien une Messor Barbarus.
2. Réponse à la question 2 et 3, On observe que l'on se place dans la logique des clauses de Horn. Dans cette logique, le modus ponens est cohérent et complet. Les questions peuvent être aussi répondu via model checking.

Question 8 (5 points)

Soit la fonction de coût suivante : $J(w_1, w_2, w_3) = (2w_1^2 + 5w_2w_3)^2$, avec w_1, w_2 et w_3 étant des paramètres à apprendre.

1. *(1 point)* Dessinez le graphe de dépendance de cette fonction.
2. *(3 point)* Les paramètres sont initialisés aux valeurs $w_1 = 1$, $w_2 = 2$, et $w_3 = 1$. Effectuez deux étapes de descente de gradient en utilisant un taux d'apprentissage de 0.1. Veillez à bien détailler vos calculs et à indiquer les valeurs des paramètres et du coût après chaque itération.
3. *(1 point)* Après ces 2 itérations, que pensez-vous de la valeur de ce taux d'apprentissage ? Devrait-on le diminuer, l'augmenter, ou garder sa valeur ?

Solution:

1. Un graphe plus simple que celui proposé est possible.
2. Les passes en avant sont disponible ci-dessous.
   — Première backpropagation

$$\frac{\delta J}{\delta w_1} = \frac{\delta J}{\delta e} * \frac{\delta e}{\delta c} * \frac{\delta c}{\delta a} * \frac{\delta a}{\delta w_1} = 2e * 1 * 2 * 2w_1 = 24 * 1 * 2 * 2 = 96 \quad (6)$$

$$\frac{\delta J}{\delta w_2} = \frac{\delta J}{\delta e} * \frac{\delta e}{\delta d} * \frac{\delta d}{\delta b} * \frac{\delta b}{\delta w_2} = 2e * 1 * 5 * w_3 = 24 * 1 * 5 * 1 = 120 \quad (7)$$

$$\frac{\delta J}{\delta w_3} = \frac{\delta J}{\delta e} * \frac{\delta e}{\delta d} * \frac{\delta d}{\delta b} * \frac{\delta b}{\delta w_3} = 2e * 1 * 5 * w_2 = 24 * 1 * 5 * 2 = 240 \quad (8)$$

$$w_1 = w_1 - 0.1 * \frac{\delta J}{\delta w_1} = -8.6 \quad (9)$$

$$w_2 = w_2 - 0.1 * \frac{\delta J}{\delta w_1} = -10 \quad (10)$$

$$w_3 = w_3 - 0.1 * \frac{\delta J}{\delta w_1} = -23 \quad (11)$$

   — Seconde backpropagation

$$\frac{\delta J}{\delta w_1} = \frac{\delta J}{\delta e} * \frac{\delta e}{\delta c} * \frac{\delta c}{\delta a} * \frac{\delta a}{\delta w_1} = 2e * 1 * 2 * 2w_1 = 2495.84 * 1 * 2 * -17.2 = -85856.896 \quad (12)$$

$$\frac{\delta J}{\delta w_2} = \frac{\delta J}{\delta e} * \frac{\delta e}{\delta d} * \frac{\delta d}{\delta b} * \frac{\delta b}{\delta w_2} = 2e * 1 * 5 * w_3 = 2495.84 * 1 * 5 * -22 = -274542.4 \quad (13)$$

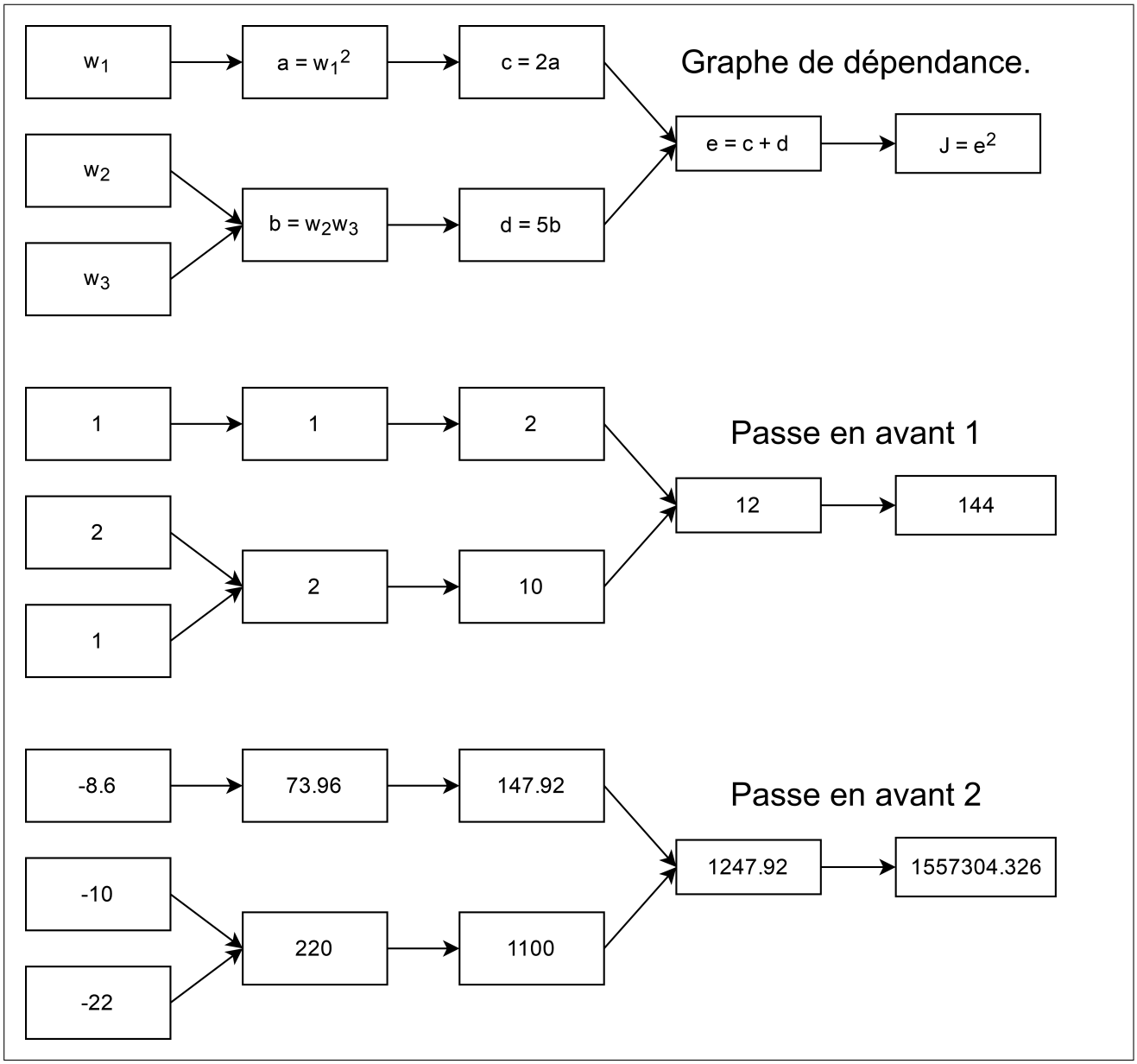
$$\frac{\delta J}{\delta w_3} = \frac{\delta J}{\delta e} * \frac{\delta e}{\delta d} * \frac{\delta d}{\delta b} * \frac{\delta b}{\delta w_3} = 2e * 1 * 5 * w_2 = 24 * 1 * 5 * -10 = -124792 \quad (14)$$

$$w_1 = w_1 - 0.1 * \frac{\delta J}{\delta w_1} = 8577.0896 \quad (15)$$

$$w_2 = w_2 - 0.1 * \frac{\delta J}{\delta w_1} = 27444.24 \quad (16)$$

$$w_2 = w_3 - 0.1 * \frac{\delta J}{\delta w_1} = 12457.2 \quad (17)$$

3. On observe que les poids oscillent de valeurs extrêmes positives à des valeurs extrêmes négatives. C'est symptomatique d'un taux d'apprentissage trop élevé. il faut réduire le taux d'apprentissage.


w_1
$a = w_1^2$
$c = 2a$
Graphe de dépendance.
w_2
$b = w_2 w_3$
$d = 5b$
$e = c + d$
$J = e^2$
w_3
1
1
2
Passe en avant 1
12
144
2
2
10
1
-8.6
73.96
147.92
Passe en avant 2
1247.92
1557304.326
-10
220
1100
-22