Jeux et Apprentissage
Matthieu Boyer
Planche A

Khôlle MPI
04/11/2024

Variations sur MinMax

On considère le jeu à deux joueurs suivant :

Définition 0.1 — Belote en moins bien. On dispose d'un ensemble E de $2n$ cartes, et d'une fonction de valeur $\omega : E \to \mathbb{R}^+$. On va distribuer n cartes à chaque joueur, ce qui forme deux mains $a_1, \dots, a_n$ et $b_1, \dots, b_n$. À chaque tour, le joueur A va jouer une carte x et le joueur B va jouer une carte y, puis, le joueur dont la carte à la plus forte valeur remporte $\omega(x) + \omega(y) = \omega(x, y)$ points.

Remarquons que connaître la main d'un des deux joueurs et les coups précédents permet de connaître la main de l'autre joueur.

1. On suppose d'abord que A et B coopèrent pour maximiser le score de A. Comment modifier l'algorithme minmax dans ce cas ?
2. On suppose désormais que A et B coopèrent pour maximiser le score d'un des deux joueurs. Comment modifier l'algorithme minmax dans ce cas ?
3. Décrivez un algorithme permettant à B de maximiser son score, sans la coopération de A.
4. On suppose maintenant que B ne sait pas ce qu'a joué A au moment de poser sa carte. Comment modifier l'algorithme minmax dans ce cas ?

Collisions et Élagage

On s'intéresse au problème de détecter dynamiquement les collisions entre des objets dans l'espace en au moins deux dimensions. On définit la différence de Minkowski de deux parties de l'espace A et B par :

$$A - B = \{a - b \mid a \in A, b \in B\}$$

1. Montrer qu'on peut vérifier si A et B s'intersectent en résolvant un problème de minimisation sur leur différence de Minkowski.
2. En déduire un algorithme naïf pour calculer vérifier si un ensemble d'objets entre en collision. Donner sa complexité en fonction du nombre d'objets et du coût du calcul de collisions.

Ici, on va commencer par étudier l'état initial. On va chercher à réduire le nombre de vérifications à faire, de sorte à obtenir un algorithme nécessitant un temps de précalcul permettant d'effectuer en moyenne $\mathcal{O}(m)$ vérifications, où m est le nombre de collisions sur un seul axe, qu'on fixe au début. On définit :

$$\begin{aligned} A_\alpha &= \mathbb{R}_{\leq \alpha} \times \mathbb{R} \\ B_\alpha &= \mathbb{R}_{\geq \alpha} \times \mathbb{R} \\ C_\alpha &= [-\alpha, \alpha] \times \mathbb{R} \end{aligned}$$

3. Montrer que si $x \leq -y$ et $z \geq y$, alors $A_x \cap C_y = \varnothing \Longrightarrow A_x \cap B_z = \varnothing$
4. En déduire un algorithme en $\mathcal{O}(n \log n + m)$ vérifiant s'il y a des collisions dans un ensemble d'objets.

Dans le cas dynamique, on suppose qu'un objet ne peut dépasser qu'au plus un autre à la fois sur l'axe considéré. On va optimiser l'algorithme dans le cas dynamique en considérant deux phases : le balayage et l'élagage. Pour la phase de balayage, on va parcourir la liste triée des bords (gauche et droits) en maintenant l'ensemble des objets pouvant potentiellement être en collision à une abscisse donnée. Pour la phase d'élagage, on va utiliser le tri par insertion pour améliorer notre complexité.

5. Dans quel(s) cas y a-t-il un échange de deux arêtes dans la liste ? Que cela implique-t-il dans l'ensemble des collisions ? En déduire un algorithme en $\mathcal{O}(n + m)$ réalisant la modification dynamique.

Jeux et Apprentissage
Matthieu Boyer
Planche A : Corrigés

Khôlle MPI
04/11/2024

Solution: Variations sur MinMax

1. Ici, on applique simplement minimax avec pour étiquetage le score de A. En effet, l'ordre dans lequel on joue les paires n'importe pas. La complexité ne change pas.
2. Ici, on regarde les paires à jouer pour obtenir le score maximal pour chacun des joueurs. La complexité ne change pas, on fait simplement 2 fois chaque opération.
3. C'est minimax avec le score de B en un peu plus compliqué !
4. Dans l'implémentation, on regarde simplement tous les coups possibles pour A, ceci rend la complexité exponentielle toutefois.

Solution: Collisions et Élagage

Dans la suite, on dit que deux objets sont en superposition dans le cas où ils sont en collision sur l'axe privilégié.

1. On a $A \cap B \neq \varnothing$ si et seulement si $\min_{x \in A-B} \|x\| = 0$.
2. On calcule pour toute paire d'objets le minimum des normes de la différence de Minkowski.
3. Faire un dessin.
4. On trie les arêtes et on ne regarde que les collisions possibles en considérant que les paires où il peut y avoir intersection.
5. On a un échange d'arêtes L-R, R-L, L-L ou R-R. Dans les deux derniers cas, si les objets étaient déjà en superposition, ils le restent. Dans le cas R-L, ils entrent en superposition et dans le cas L-R ils ne sont plus en superposition.
6. En remarquant qu'il ne peut y avoir plus de $\frac{n}{2}$ échanges, on a le résultat en changeant de tri dans l'algorithme initial.

On pourrait améliorer encore l'algorithme en effectuant le calcul sur plusieurs axes à la fois. Ceci demanderait de multiplier par d le nombre d'axes le coût de pré-traitement.

Jeux et Apprentissage
Matthieu Boyer
Planche B

Khôlle MPI
04/11/2024

Algorithme Max^n

On donne les règles d'un jeu :

Définition 0.2 — Dames Chinoises. Sur les intersections d'un damier en étoile, chacun des six joueurs a 10 pions de sa couleur, chacun dans l'une des pointes de l'étoile. Le but du jeu est de tous les déplacer dans la pointe opposée. Chacun à son tour déplace un pion, de l'une des deux façons suivantes : Vers une case libre contiguë, ou bien en faisant un nombre quelconque de sauts avec ce pion. Chaque saut se fait vers une case libre, par dessus une case occupée.

1. En reprenant le principe de l'algorithme minimax, proposer un algorithme donnant une stratégie optimale pour le jeu des dames chinoises. Le but n'est pas d'obtenir un algorithme optimal (je n'en connais pas), on ne cherchera pas donc une heuristique efficace. On pourra dessiner un arbre de jeu à 3 joueurs pour observer la manière dont un algorithme similaire pourrait fonctionner.
2. Comment pourrait-on adapter les optimisations de l'algorithme $\alpha - \beta$ à notre nouvel algorithme ?

En pratique ces optimisations, ne sont pas très efficaces, puisqu'il n'y a que peu d'améliorations.

3. En supposant qu'on veut maximiser le score d'un seul joueur, montrer qu'on peut se ramener au cas d'un jeu à deux joueurs. Dans le meilleur cas, si le joueur à profondeur d peut jouer b coups, combien de noeuds devra-t-on développer ?

Solution: Algorithme Max^n

L'idée principale est de comprendre que chaque joueur va chercher à maximiser son score en minimisant celui des 5 autres. On cherche donc des heurisiques h_i à valeurs dans $(\mathbb{R}^+)^6$. Ici, on peut par exemple prendre les distances des billes d'un joueur à l'objectif (ou plutôt la distance maximale moins la distance actuelle, pour avoir une maximisation et rester dans le cadre de l'algorithme vu en cours). Les améliorations de $\alpha - \beta$ se traduisent directement à l'algorithme, en les appliquant sur chaque sous-arbre. On peut également considérer que $n-1$ joueurs s'allient contre notre joueur favori, et qu'ils cherchent à maximiser le score que l'un d'entre eux peut obtenir. Voir `https://cdn.aaai.org/AAAI/2000/AAAI00-031.pdf`

Jeux et Apprentissage
Matthieu Boyer
Planche C

Khôlle MPI
04/11/2024

Classification par Arbres

On va chercher à classifier un ensemble de points de l'espace à 2 dimensions.

1. Rappeler l'algorithme des k-moyennes.

Observons que le plan peut être vu comme $\mathbb{R}^+ \times \mathbb{R}^+ \sqcup \mathbb{R}^- \times \mathbb{R}^+ \sqcup \mathbb{R}^+ \times \mathbb{R}^- \sqcup \mathbb{R}^- \times \mathbb{R}^-$.

2. Proposer une structure de données basée sur un arbre permettant de représenter l'ensemble des points dans un arbre.
3. Quel est le nombre maximal de noeuds de l'arbre pour n points ?
4. Cette structure est-elle utilisable seule pour de la classification ? Proposer une méthode de classification utilisant plusieurs quadtrees.
5. Expliquer comment généraliser les quadtrees à plus de dimensions.
6. Proposer une méthode de séparation de l'espace en clusters à partir de quadtrees.

Solution: Classification par Arbres

1. Pour trouver P, on compare par rapport au point de dichotomie. Pour insérer un point P, on trouve la cellule où devrait aller P, et si celle-ci est vide, on la remplit, et si elle a un point, on découpe de manière répétée.
2. Si on note c la plus petite distance entre deux points, et si s est la norme de Manhattan maximale d'un point, alors la profondeur de l'arbre vérifie :
$$d \leq \log\left(\frac{s}{c}\right) + \frac{3}{2}$$
En effet, pour que deux noeuds ne soit pas séparés à profondeur i on doit avoir $s\sqrt{2}^{2^i} \geq c$ et donc $i \leq \log s\sqrt{2}/c = \log s/c + 1/2$. Puisque chaque noeud interne représente un carré avec au moins 2 noeuds, chaque niveau a au plus n noeuds, et donc un arbre représentant n points a $\mathcal{O}\left((d+1)n\right)$ noeuds.
3. Non, $(0, -2^{-i})$ et $(0, 2^{-i})$ seront toujours dans des parties de l'arbre radicalement différentes (puisqu'il faut revenir à la racine pour passer de l'un à l'autre). On peut par exemple utiliser un quadtree normal et un quadtree sur le plan après une rotation de 45°. On classifie ensuite en observant la densité.
4. On divise en 8 cubes en dimension 3 et plus généralement en 2^i hypercubes en dimension i. Il faudra alors utiliser i hyper-quadtrees pour classifier.

On peut utiliser les quadtrees pour générer des centres pour l'algorithme des k-moyennes `https://www.researchgate.net/publication/7211789_A_genetic_algorithm_using_hyper-quadtrees_for_low-dimensional_K-means_clustering`