TD13 (éléments de réponse)

Exercice 1

Question 1.
□ **1.1.** La mise en œuvre de l'algorithme du point le plus proche conduit à de nombreuses tournées. On en représente deux ci-dessous.

Si V comporte $n = 4k$ points, la solution de gauche renvoie une longueur de tournée égale à $6k - 2$. La deuxième La solution de droite, optimale et obtenue en parcourant l'enveloppe convexe du nuage de points, renvoie une longueur de tournée égale à $4k$. L'écart relatif entre ces deux solutions est $(1 - 1/k)/2$, proche de 50% dès que k est grand.
□ **1.2.** Cet algorithme est de type *glouton*.
□ **1.3.** Sa complexité temporelle est quadratique en le nombre de points puisque chaque point ajouté nécessite de comparer $O(n)$ distance.
□ **1.4.** La construction d'une tournée aléatoire serait plus efficace avec une complexité temporelle optimale en $\Theta(n)$. En contrepartie, la longueur pourrait être beaucoup plus grande que celle produite par l'algorithme glouton.

Question 2.

Question 3. Partant du point A, la tournée effectuée par un algorithme glouton est : $A \to B \to C \to D \to A$. Sa longueur est : $\textsc{Greedy}(V) = 6 + 6\delta$.

Question 4. La tournée $A \to B \to D \to C \to A$ est une tournée optimale, de longueur $\textsc{Opt}(V) = 6$.

Question 5. Le facteur d'approximation de l'algorithme glouton est donc :

$$\frac{\textsc{Greedy}(V)}{\textsc{Opt}(V)} = \frac{6 + 6\delta}{6} = 1 + \delta$$

résultat non constant qui dépend de la valeur de δ. L'algorithme glouton n'est donc pas une α-approximation.

Exercice 2

Question 1. Si H possède un cycle hamiltonien, l'algorithme d'approximation doit renvoyer une tournée de longueur au plus αn puisque la longueur optimale est dans ce cas n. Cette longueur $\alpha n < n^2$ par hypothèse sur α. Et si H ne possède pas de cycle hamiltonien, la tournée renvoyée par A contiendra au moins une paire de points visités consécutivement v_i, v_{i+1} ne correspondant pas à une arête de H, donc avec $d_H(v_i, v_{i+1}) = n^2 > \alpha n$.

Question 2. Si TSP admet une α-approximation, on peut proposer l'algorithme suivant pour résoudre HamCycle.

Algorithme 1 : algorithme $\textsc{HamCycle}(H)$

```
1 tranformer H en une instance (V_H, d_H) de TPS
2 renvoyer le booléen (A(V_H, d_H) < n^2) où n = |V_H|.
```

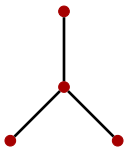
Question 3. On a ainsi réduit le problème HamCycle à celui de l'approximation de TSP : on peut résoudre HamCycle à l'aide d'une α-approximation de TSP, en temps polynomial puisque chaque étape de HamCycle prend un temps polynomial :

- $O(n^2)$ pour la première étape de transformation ;
- polynomial par définition de A pour la deuxième étape.

Or HamCycle est NP-complet. Par conséquent, sauf P = NP, A n'est pas une α-approximation. Ce qui implique que TSP est inapproximable.

Exercice 3

Question 1. Pour pouvoir apparier tous ses sommets par des arêtes indépendantes, un graphe doit posséder un nombre pair de sommets. Mais une telle condition ne garantit pas l'existence d'un couplage parfait. Par exemple, le graphe représenté ci-dessous, dit en étoile à trois branches, n'en admet pas.



Question 2.

□ **2.1.** La validité de l'algorithme peut être égablie à l'aide des arguments suivants.

- Le couplage parfait existe bien car $|I|$ est pair (rappelons que dans tout graphe, il existe un nombre pair de sommet de degré impair) et que le graphe induit par I est une clique.
- L'ajout du couplage parfait F à T produit un multi-graphe où tous les sommets sont de degré pairs, puisqu'on ajoute exactement une arête incidente à chaque sommet de degré impair de T.
- Le circuit eulérien de $T \cup F$ visite au moins une fois chacun des sommets de V puisque toutes les arêtes sont visitées et que T couvre V.

Par ailleurs, dans l'algorithme, l'étape la plus coûteuse et la plus délicate à mettre en œuvre est celle du calcul du couplage parfait, de complexité $O(n^3)$. Par conséquent, la complexité totale de l'algorithme est $O(n^3)$.

□ **2.2.** La longueur de la tournée renvoyée par l'algorithme est au plus la somme des poids des arêtes du circuit eulérien de $T \cup F$. Cela peut être moins car on saute les sommets déjà visités, ce qui grâce à l'inégalité triangulaire produit un raccourcis. On a donc $\textsc{ChristofidesApprox}(V,d) \leqslant d(T \cup F) = d(T) + d(F)$.
On a vu (en cours) que $d(T) < d(C^*) = \textsc{Opt}(V,d)$.
Notons C_I la tournée optimale pour l'instance (I,d) restreinte aux sommets I. Clairement $d(C_I) \leqslant d(C^*)$ puisque $I \subseteq V$. À partir de la tournée C_I, on peut construire deux couplages parfaits pour I : l'un obtenu en prenant une arête sur deux, l'autre en prenant son complémentaire. Le plus léger d'entre eux a un poids inférieur ou égal à $d(C_I)/2$ puisque leur somme fait $d(C_I)$. Il suit que le couplage parfait F de poids minimum pour I est de poids $d(F) \leqslant d(C_I)/2$.
En combinant ces inégalités, on obtient :

$$\textsc{ChristofidesApprox}(V,d) \leqslant d(T) + d(F) < d(C^*) + \frac{1}{2}d(C_I)$$

Puis :

$$\textsc{ChristofidesApprox}(V,d) = d(C^*) + \frac{1}{2}d(C^*) = \frac{3}{2}d(C^*)$$

et enfin :

$$\textsc{ChristofidesApprox}(V,d) \leqslant \frac{1}{2}\textsc{Opt}(V,d)$$

Ce qui montre que le facteur d'approximation est de 1.5.

Exercice 4

Question 1. Pour $n \geqslant 2$, notons I_n l'instance de SubsetSum suivante : $C = n$ et $T = [1,n]$. L'algorithme proposé renvoie sur cette instance le sous ensemble $\{1\}$ de somme $S_n = 1$ alors qu'une solution optimale consisterait en le sous ensemble $\{n\}$ de somme $S_n^* = n$.
Si cet algorithme était un algorithme d'approximation à facteur constant, il existerait $\alpha > 0$ tel que pour tout $n \geqslant 2$ on ait $\alpha S_n^* \leqslant S_n$. Mais en passant à la limite, ce qui précède montre que $\alpha \leqslant 0$!

Question 2.

□ **2.1.** Le tri peut se faire en $O(n \log n)$ qui est aussi la complexité globale car la boucle se fait en $O(n)$.

□ **2.2.** Notons S la somme obtenue avec l'algoritme et S^* la somme optimale pour une instance donnée.

- Si la somme de tous les t_i est inférieure à C alors $S = S^* \geqslant S^*/2$.
- Si aucun des t_i n'est inférieur à C alors $0 = S = S^* \geqslant S^*/2$.
- Sinon, notons S_{courant} la somme courante au moment où, dans l'algorithme, on a déjà ajouté un t_j à la varibale S puis rencontré un t_i qui ne peut pas y être ajouté sans dépasser C.
  Montrons alors que $S_{\text{courant}} \geqslant C/2$. Dans le cas contraire, on aurait $S_{\text{courant}} < C/2$. Vu l'ordre dans lequel sont considérés les t_k, tous ceux qui interviennent dans S_{courant} sont supérieurs à t_i. Ainsi $C/2 > S_{\text{courant}} \geqslant t_k \geqslant t_i$.
  Mais dans ces conditions, on peut ajouter t_i à S_{courant} sans dépasser C; ce qui est impossible.
  On en déduit que $S \geqslant S_{\text{courant}} \geqslant C/2 \geqslant S^*/2$ car au mieux $S^* = C$.

Ce qui permet de conclure dans tous les cas.

□ **2.3.** $1/2$ est le meilleur facteur d'approximation pour cet algorithme. Pour tout $n \geqslant 1$, notons I_n l'instance donnée par $C = 2n$ et $T = [n+1, n, n]$. La somme trouvée par l'algorithme glouton est $S_n = n+1$; la somme optimale est $S_n^* = 2n$. Si A admet un facteur d'approximation α, on a en particulier, pour tout $n \geqslant 1$:

$$\alpha \leqslant \frac{S_n}{S_n^*} = \frac{n+1}{2n}$$

et en passant à la limite on a $\alpha \leqslant 1/2$.