Exercices du cours d'optimisation combinatoire

Claire Hanen

20 décembre 2019

Table des matières

Chapitre 1

Modélisation

Exercice 1.1 Binômes

Modéliser le problème suivant à l'aide d'un programme mathématique. On considère un ensemble de personnes $P = \{1, \ldots, n\}$ qui doivent travailler en binôme sur des projets (on supposera que n est pair). Chaque personne possède une capacité c_i connue à l'avance et qui mesure son efficacité au travail : plus c_i est petit, plus la personne est rapide dans l'exécution de sa part du projet. Si on constitue un binôme entre deux personnes i et j alors le temps que mettra le binôme pour terminer son projet est $c_i + c_j$.

On dispose de plus d'un graphe d'incompatibilité G, dont les sommets sont les personnes. L'existence d'une arête $\{i, j\}$ signifie que les personnes i et j ne peuvent pas être en binôme.

Le but est de répartir les personnes en binômes compatibles de sorte que tous les projets soient terminés en un minimum de temps (Les binômes travaillent en parallèle).

Exercice 1.2 Performance d'une installation reseau

Un installateur d'équipements réseau cherche à tester les performances de son réseau d'antennes GSM. Il y a n antennes numérotées de 1 à nr, positionnées géographiquement.

Les antennes proches (et selon les éléments présents, immeubles, etc) ne peuvent émettre en même temps sans créer des interférences et fausser les signaux. On dispose d'un graphe G non orienté, dit graphe d'interférence, dont les sommets représentent les antennes et dont chaque arête $\{i, j\}$ exprime le fait que les antennes i et j ne peuvent émettre en même temps sous peine d'interférences.

Un sous-ensemble X des antennes est dit compatible si les antennes qui

le composent peuvent émettre en même temps sans interférence.

L'entreprise cherche à déterminer un ensemble compatible de cardinalité maximale. Modéliser ce problème.

Exercice 1.3 Gestion de stock

Une usine automobile cherche à plannifier sa production sur les H prochaines semaines. Elle dispose de ses prévisions, qui donnent pour chaque semaine i un nombre de véhicules D_i à livrer en fin de semaine.

Selon les aléa de la production (notamment congés, coût des transports, etc), le coût de production des véhicules varie selon les semaines. On note c_i le coût de production d'un véhicule durant la semaine i. Produire x véhicules pendant la semaine i coût donc $c_i x$. L'usine a, de plus, une capacité de production fixe par semaine, notée Q (c'est à dire ne peut produire plus de Q véhicules par semaine).

Ces deux éléments peuvent conduire à produire des véhicules en avance pour une commande d'une semaine ultérieure. Les véhicules doivent donc être stockés. Chaque véhicule stocké à la fin de la semaine i a un coût unitaire de stockage noté t_i.(stocker x véhicules en fin de semaine i coûte donc $t_i x$).

La production se déroule donc selon le schéma suivant : chaque semaine i, le stock restant de la semaine précédente est complété avec la production de la semaine, le client est livré en fin de semaine, et il peut éventuellement rester un stock pour la semaine suivante. L'entreprise cherche à calculer un plan de production, c'est à dire savoir combien de véhicules doivent être produits chaque semaine, de sorte que le coût total de production et de stockage soit minimum et que tous les clients soient livrés. Modéliser ce problème.

Exercice 1.4 Jeux Olympiques

Modéliser le problème suivant à l'aide d'un programme linéaire en nombres entiers. On considère un complexe sportif utilisé pour les JO , avec des installations fixes et des emplacements prévus pour différentes compétitions. On note $S = \{1, \ldots, n\}$ l'ensemble des n emplacements de compétitions.

Pour pouvoir suivre les compétitions, on installe par avance, à des points clés, des caméras fixes. On dispose de m emplacements possibles pour les caméras. L'ensemble de ces emplacements est noté E.

Chaque emplacement de compétition $i \in S$ doit pouvoir être filmé par n_i caméras (n_i est un nombre entier). On note $E_i \subset E$ l'ensemble des emplacements qui permettent, si l'on y installe une caméra, de filmer les compétitions placées en i.

On souhaite disposer un nombre total minimum de caméras à certains

emplacements de E de sorte que chaque emplacement de compétition i dispose de n_i caméras.

On supposera qu'il existe une solution réalisable, c'est à dire qu'en plaçant une caméra à chaque emplacement possible, toutes les compétitions sont bien couvertes.

Exercice 1.5 Réseau de communication

Modéliser le problème suivant : On considère un graphe d'interférence non orienté G, comportant un ensemble de n sommets $S = \{1, \ldots, n\}$ et un ensemble d'arêtes A. Les sommets représentent des stations d'émission, les arêtes représentent des interférences entre les émissions de ses deux noeuds extrémités : une arête $\{i, j\}$ indique que le noeud i et le noeud j ne peuvent émettre en même temps. Chaque noeud i a de plus une capacité d'émission (en termes de nombre de bits par seconde) notée c_i. On souhaite mesurer la capacité maximale du réseau c'est à dire le nombre maximal de bits pouvant être émis en même temps compte tenu des capacités et du graphe d'interférence.

Exercice 1.6 Ordonnancement à deux machines de pénalité minimale

Modéliser le problème suivant : On considère un ensemble de n tâches à faire. Les tâches peuvent être faites sur l'une ou l'autre de deux machines différentes. Chaque tâche i a une durée u_i si elle est faite sur la première machine, v_i sur la seconde. Chaque machine n'exécute qu'une seule tâche à la fois et réalise donc toutes les tâches qui lui sont affectées en séquence (dans n'importe quel ordre) sans s'arrêter, dans une durée qui ne doit pas dépasser la durée ouvrable de la journée O. Les deux machines fonctionnent en parallèle.

Malheureusement, il n'est pas toujours possible de faire toutes les tâches dans la durée impartie. Chaque tâche i , si on ne l'exécute pas engendre une pénalité notée p_i. On cherche donc à déterminer pour chaque tâche si on l'exécute ou non, et si oui sur quelle machine, de sorte que chaque machine fonctionne sur une durée qui n'excède O et que la somme des pénalités liées aux taches non-exécutées soit minimale.

Exercice 1.7 Planning de football

Une fédération régionale de football cherche à organiser les matches de sa saison, entre ses n équipes.

Chaque équipe rencontre toutes les autreséquipes deux fois, une fois à domicile, l'autre fois chez l'adversaire. On notera pour simplifier (i, j) la rencontre entre l'équipe i et l'équipe j au domicile de i.

Pour plannifier les rencontres, on dispose d'une matrice C dont les lignes correspondent aux clubs et les colonnes aux dates possibles (numérotées de 0 à $T-1$). $C[i][t]$ correspond au coût de l'organisation d'une rencontre dans le stade de i à la date t.

Il y a plusieurs contraintes à prendre en compte pour organiser la saison :

1. Certaines rencontres sont dites importantes, car elle sont très attendues du public et font l'objet d'une retransmission à la télévision. On note I l'ensemble des rencontres importantes. Il faut s'arranger pour qu'il n'y ait pas deux rencontres importantes en même temps.
2. Il ne peut y avoir plus de 4 rencontres programmées le même jour.
3. Une équipe i ne peut pas jouer deux jours de suite.

Modéliser le problème de la recherche d'une plannification des rencontres qui minimise le coût total du planning. On précisera les inconnues et leur sens par rapport au problème, et l'on indiquera comment se traduisent les contraintes exprimées ci-dessus.

Exercice 1.8 Compétences

Un groupe de n personnes possèdent des compétences. c_i est le niveau de compétence de l'individu i. On cherche à répartir ces personnes en deux groupes. Pour chaque groupe, la compétence du groupe est la somme des compétences des personnes qui le composent. L'objectif est que ces deux groupes soient équilibrés, et pour cela, si $C1$ est la compétence du premier groupe et $C2$ la compétence du second, on va chercher à minimiser $(C1 - C2)^2$.

Modéliser ce problème d'optimisation.

Exercice 1.9 Dépôts de carburant

Une compagnie pétroli'ere cherche à implanter dans un territoire des dépôts de carburant. Elle a identifié n sites possibles (ensemble$D = \{1, \ldots, n\}$. Un site i est caractérisé par sa capacité maximale de stockage q_i, un coût fixe d'ouverture du dépôt f_i . Les dépôts ouverts délivreront de l'essence à m stations services (ensemble $S = \{1, \ldots, m\}$), avec pour chaque station service j une demande de carburant estimée à d_j. Lorsqu'une station service j commande x litres de carburant à un dépôt i, le coût de la livraison est $x * c_{ij}$ euros, où c_{ij} est un coût donné dépendant de la distance et des facilités d'accès entre les deux lieux. Les stations services, pour satisfaire leur demande de carburant, sont susceptibles de commander à plusieurs dépôts à condition qu'il soient ouverts. Afin de satisfaire à la fois les clients (stations services) et de minimiser ses coûts d'exploitation,l'entreprise doit déterminer les dépôts qui seront ouverts et comment répartir la demande des stations services (quelles quantités doivent être livrées des dépôts ouverts vers les stations) de sorte que le coût total (ouverture des dépôts et livraison du carburant) soit minimal. Modéliser ce problème. vous indiquerez apès chaque contrainte une phrase en français décrivant ce qui est modélisé par la contrainte.

Exercice 1.10 Consommation d'énergie en fonction du temps

Modéliser le problème suivant : On considère un ensemble de n tâches de durée 1 à réaliser. Chaque tâche i doit être faite dans un intervalle $[r_i, d_i]$ (elle doit commencer au plus tôt à la date r_i et se terminer au plus tard à la date d_i. Chaque tâche i nécessite une machine pour s'exécuter (pendant une unité de temps). Dans le sytème il y a m machines identiques qui peuvent fonctionner en parallèle, chaque machine ne faisant qu'une tâche à la fois.

A chaque unité de temps t correspond un coût en énergie unitaire c_t pour chaque machine fonctionnant à cette date. Les coûts peuvent être différents à chaque date en fonction par exemple des conditions climatiques prévues. Si p machines sont en train d'exécuter une tâche à la date t, alors le coût en énergie de cet instant est $c_t * p$

On cherche à construire un planning (savoir à quelle date exécuter chaque tâche) de sorte que la somme des coûts en énergie de chaque instant soit le plus faible possible.

Exercice 1.11 Ordonnancement à vitesse variable

On se donne une suite de n tâches. On connaît pour chaque tâche i sa durée p_i lorsqu'elle est effectuée par une machine de vitesse 1. Ces tâches doivent être effectuées dans l'ordre $1, 2, \ldots n$ par une machine qui peut être plus où moins rapide selon la quantité d'énergie qu'on dépense.

On dispose de K vitesses différentes et entières $1 \leq v_1 \leq \ldots \leq v_K$. A chaque vitesse v_j correspond deux coefficents b_j (coût fixe) et c_j (coût variable), de sorte que chaque tâche i, lorsqu'elle est effectuée à vitesse v_j dure $\frac{p_i}{v_j}$ et coûte en énergie $c_j p_i + b_j$.

On cherche à associer à chaque tâche une vitesse de la machine, de sorte que la durée totale de l'ensemble des tâches ne dépasse pas une valeur donnée D (on supposera que $\frac{\sum_{i=1}^{n} p_i}{v_K} \leq D \leq \sum_{i=1}^{n} p_i$), et la dépense totale d'énergie soit minimale.

Question 1 : Modéliser ce problème à l'aide d'un programme mathématique.

Question 2 :Décrire le modèle associé aux données numériques suivantes :

Tâches :

i	1	2	3	4	5	6
p_i	3	1	2	2	4	3

vitesses :

j	1	2	3
v_j	1	2	4
b_j	1	2	3
c_j	1	3	5

$D = 7$

Chapitre 2

Méthodes arborescentes

Exercice 2.1 Etat d'une arborescence

Considérons une arborescence valide pour un problème de minimisation, en supposant qu'en chaque noeud, la fonction d'évaluation par excès est définie par une heuristique qui fournit une solution particulière associée au noeud. La numérotation des noeuds correspond à l'ordre de création de ces noeuds.

— la racine S_0 possède trois fils S_1, S_2, S_3.

— S_2 possède deux fils S_4, S_5.

noeuds	S0	S1	S2	S3	S4	S5
excès	210	200	200	205	190	170
défaut	135	200	135	140	180	150

Question 1 : Indiquer la valeur de la meilleure solution rencontrée.

Question 2 : Quelles opérations de troncature peut-on effectuer sur cette arborescence à chaque étape de sa création ?

Question 3 : Quel est le prochain sommet à séparer selon la stratégie choisie (profondeur d'abord, largeur d'abord ou meilleur d'abord) ?

Exercice 2.2 Planning hospitalier

On considère un hôpital disposant de plusieurs chirurgiens généralistes, mais chacun ayant des compétences plus ou moins reconnues dans les différents types d'opération. Cet hôpital a décidé d'optimiser son fonctionnement pour une meilleure efficacité des soins en planifiant les opérations de chaque semaine de la manière suivante. Si $P = \{1, \ldots, n\}$ est l'ensemble des patients

susceptibles d'être opéré cette semaine là, chaque patient i se voit affecté d'une durée prévue p_i de son opération,, d'une valeur u_i mesurant l'urgence de son opération (plus la valeur est grande, plus l'opération est urgente), et d'une pathologie m_i (un entier entre 1 et M le nombre de pathologies).

Par ailleurs le staff de chirurgiens disponibles cette semaine là est noté $C = \{1, \ldots, K\}$. Chaque chirurgien j est caractérisé par une durée maximale de présence dans l'hopital cette semaine là notée r_j, et pour chaque type de pathologie $m \in \{1, \ldots, M\}$ on connait une valeur entière v_{jm} qui mesure l'efficacité du chirurgien j pour opérer la pathologie m. Les normes de qualité de l'hôpital que chaque patient soit opéré par un chirurgien ayant une valeur au moins égale à un entier V par rapport à la pathologie du patient.

On cherche à affecter chaque patient à un chirurgien cette semaine là, sachant qu'il y aura peut-être des patients qui seront reportés la semaine suivante (donc pas affectés), et que le chirurgien j n'opère pas plus de r_j unités de temps ; tout en respectant la norme de qualité de l'hôpital.

Le critère d'optimisation est alors la minimisation de la somme des valeurs d'urgence des patients qui seront reportés à la semaine suivante.

Question 1 : Modéliser ce problème à l'aide d'un programme linéaire en nombres entiers.

Question 2 : Considérons le problème précédent, mais avec un seul chirurgien omni-compétent, qui a donc la valeur attendue V sur l'ensemble des pathologies. Donner une expression simplifiée de ce problème.

Question 3 : Proposer une méthode arborescente de résolution de ce problème, en vous inspirant de celles étudiées en cours.

Question 4 : Appliquez votre méthode arborescente au problème numérique suivant, décrivant les patients du chirurgien, celui-ci ayant une durée opératoire maximale de 6 :

patient i	1	2	3	4	5	6	7
urgence u_i	20	15	20	12	10	3	15
durée p_i	3	1	2	4	1	2	3

Exercice 2.3 Equipe sportive

On considère un ensemble de n joueurs. Chaque joueur j possède un indice de performance, résultant de ses activités antérieures, noté u_j et appelé l'utilité du joueur. Un sélectionneur souhaite constituer une équipe de 23 joueurs d'utilité maximale (l'utilité d'une équipe étant la somme des utilités de ses joueurs).

Mais pour faire son choix il doit se soumettre à certaines obligations. Tout d'abord il dispose d'une enveloppe financière maximale F, sachant que la sélection d'un joueur j nécessite de payer une somme c_j à son club d'origine.

D'autre part le sélectionneur dispose d'un graphe d'inimitié : les sommets représentent les joueurs, et une arête entre deux sommets i et j indique que les deux joueurs ne peuvent pas jouer dans la même équipe.

Question 1 : Modéliser le problème de l'entraîneur, et notons le $P1$. Notons $Z1$ l'utilité maximale d'une équipe pour ce problème.

Question 2 : Considérons maintenant le problème obtenu en supposant que l'équipe n'a pas un nombre de joueurs donné à l'avance, mais qu'elle peut-être de taille variable. Que faut-il changer au modèle ? Notons le $P2$, et $Z2$ l'utilité maximale d'une équipe. Quelle relation y-a-t-il entre $Z1$ et $Z2$ pour les mêmes données (nombre de joueurs, graphe d'inimitié, coûts et utilités des joueurs) ?

Question 3 : Considérons maintenant le problème $P3$ obtenu à partir de $P2$ en remplaçant le graphe d'inimitié par un graphe vide (en supposant que tous les joueurs s'aiment). Notons $Z3$ l'utilité maximale d'une équipe pour le problème $P3$. Quelle relation y-a-t-il entre $Z3$ et $Z2$ pour les mêmes données (nombre de joueurs, coûts et utilités des joueurs) ?

Question 4 : Quel algorithme pouvez-vous proposer pour majorer $Z3$?

Question 5 : Proposer un algorithme permettant de construire une équipe solution du problème $P2$ (pas nécessairement optimale).

Dans la suite on s'intéresse à une méthode arborescente pour résoudre le problème $P2$. Un noeud de l'arborescence S est caractérisé par un sous-ensemble de joueurs $C(S)$ choisis et un sous-ensemble de joueurs $R(S)$ rejetés. Il est associé à l'ensemble des équipes (qui rentrent dans l'enveloppe financière et respectent les inimitiés) contenant les joueurs de $C(S)$ et pas ceux de $R(S)$.

La séparation d'un noeud S se fait en déterminant un joueur j ni choisi

ni rejeté, ni ennemi (selon le graphe) d'un joueur de $C(S)$ et en construisant deux fils S', S''. Dans S' on ajoute j aux joueurs choisis, dans S'' on l'ajoute aux joueurs rejetés.

Question 6 : Quel algorithme pouvez-vous proposer obtenir une évaluation par excès du noeud S ?

Question 7 : Quel algorithme pouvez-vous proposer obtenir une évaluation par défaut du noeud S ?

Exercice 2.4 Stockage de fichiers

On considère un ensemble de fichiers $\mathcal{F} = \{1, \ldots, n\}$ de tailles respectives (en octets) $l_1, \ldots, l_n$. On dispose de deux disques durs de taille M (en octets).

On cherche à stocker un nombre maximum de fichiers, sachant qu'un fichier ne peut pas être découpé entre les deux disques, et qu'il est possible qu'il ne soit pas faisable de les stocker tous.

Les applications numériques se feront sur l'exemple suivant : $n = 5$, $M = 4$, $l_2 = l_5 = 1$, $l_1 = l_4 = 2$,$l_3 = 3$.

Question 1 : En notant x_i la variable de décision qui vaut 1 si le fichier est stocké sur le premier disque, 0 sinon, et y_i la variable de décision qui vaut 1 si le fichier est stocké sur le second disque, et 0 sinon, modéliser le problème à l'aide d'un programme linéaire à variables bivalentes.

On s'intéresse maintenant à la définition d'une méthode arborescente pour résoudre ce problème.

Question 2 : On considère le problème suivant : étant donné un seul disque de taille C, quel est le nombre maximum de fichiers de $\mathcal{F}$ pouvant être stockés sur ce disque ? Indiquer un algorithme permettant de résoudre ce problème.

Question 3 : A chaque noeud S de l'arborescence seront associés quatre sous-ensembles de fichiers disjoints : $R(S)$ l'ensemble des fichiers rejetés, $D_1(S)$ l'ensemble des fichiers affectés au disque 1, $D_2(S)$ l'ensemble des fichiers affectés au disque 2, et $F(S)$ l'ensemble des autres fichiers : $F(S) = \mathcal{F} \setminus (R(S) \cup D_1(S) \cup D_2(S))$. On supposera que $M_1(S) = M - \sum_{i \in D_1(S)} l_i \geq 0$ et $M_2(S) = M - \sum_{i \in D_2(S)} l_i \geq 0$.

Notons $X(S)$ le nombre maximum de fichiers parmi $F(S)$ pouvant être stockés sur un disque de taille $M_1(S) + M_2(S)$.

Montrer que $H(S) = X(S) + |D_1(S) \cup D_2(S)|$ est une évaluation par

excès de la fonction objectif au noeud S.

Question 4 : On considère l'algorithme suivant, permettant de calculer une solution associée au noeud S :

- classer les fichers de $F(S)$ par ordre de taille croissante.
- Affecter les premiers au disque 1, jusqu'à hauteur de $M_1(S)$ octets maximum.
- Affecter les suivants au disque 2 jusqu'à hauteur de $M_2(S)$ maximum.

Calculer pour l'exemple numérique la solution ci-dessus et l'évaluation par excès de la question précédente associée à la racine de l'arbre (noeud correspondant à l'ensemble de toutes les solutions).

Question 5 : La séparation d'un noeud S consiste à choisir un fichier $j \in F(S)$, et à générer au plus trois fils S', S'', S''' avec :

- $R(S') = R(S) \cup \{j\}$, $D_1(S') = D_1(S)$, $D_2(S') = D_1(S)$, $F(S') = F(S) \setminus \{j\}$
- lorsque $l_j \leq M_1(S) : R(S'') = R(S), D_1(S'') = D_1(S) \cup \{j\}, D_2(S'') = D_1(S)$, $F(S'') = F(S) \setminus \{j\}$
- lorsque $l_j \leq M_2(S)$:$R(S''') = R(S)$, $D_1(S''') = D_1(S)$, $D_2(S''') = D_1(S) \cup \{j\}$, $F(S''') = F(S) \setminus \{j\}$

Appliquer la méthode arborescente ainsi définie à l'exemple, en précisant les règles de parcours de l'arborescence choisies, les évaluations des noeuds, et les troncatures effectuées.

Exercice 2.5 Problème d'ordonnancement sur une machine

Dans la suite de l'exercice, on s'intéresse à la définition d'une méthode arborescente pour résoudre le problème d'ordonnancement suivant :

On a n tâches. Chaque tâche i possède deux paramètres : sa durée p_i et sa date d'échéance d_i. Ces tâches doivent être effectuées par une machine, sans temps d'arrêt. Etant donné un ordonnancement, si l'on note C_i la date de fin de la tâche i, le retard de i est noté $T_i = \max(C_i - d_i, 0)$. On cherche un ordonnancement (c'est à dire une permutation σ des tâches) qui minimise $\sum_{i=1}^{n} T_i^2$.

Dans cet exercice on définit une méthode arborescente pour ce problème.

Un noeud S sera associé à une sous-suite de tâches $J_S = \{i_1, \ldots, i_r\}$, et comprend l'ensemble des ordonnancements σ qui se terminent par la séquence J_S : $\sigma(n) = i_r, \ldots, \sigma(n - r + 1) = i_1$. Par exemple, si $J_S = 4, 3$ alors l'ensemble des ordonnancements associés au noeud S est l'ensemble des séquences qui se terminent par $4, 3$. Sur 5 tâches $2, 1, 5, 4, 3$ est l'une de ces séquences.

Notons $P_S = \sum_{i \notin J_S} p_i$.

Question 1 : Calculez α_S la somme des carrés des retards des tâches de la séquence J_S dans tout ordonnancement associé a noeud S en fonction des données du problème (durées des tâches, échéances).

Question 2 :Montrer que dans tout ordonnancement associé au noeud S, une tâche qui n'appartient pas à J_S aura un retard au moins égal à

$$\beta_S = \min_{i \notin J_S}(\max(0, P_S - d_i))$$

Question 3 : En déduire que $h(S) = \alpha_S + \beta_S^2$ est une évaluation par défaut du noeud S.

Question 4 :Proposer un algorithme glouton permettant de construire une solution particulière associée au noeud S.

Question 5 : Un noeud S est séparé en autant de fils qu'il y a de tâches $j \notin J_S$, correspondant aux sous-séquences $j, i_1, \ldots, i_r$. Appliquez la méthode arborescente (en fonction du temps en développant l'arbre sur un ou deux niveaux) à l'exemple suivant en précisant, pour les noeuds que vous développerez, le choix du noeud à séparer, les évaluations de chacun des noeuds développés, les mises à jour et les troncatures effectuées.

i	1	2	3	4	5
p_i	4	3	7	2	5
d_i	5	6	8	8	12

Exercice 2.6 Sac à dos bi-dimensionnel

Des étudiants ont réalisé le projet suivant : proposer et programmer une méthode arborescente pour le problème de sac à dos bidimensionnel : chaque objet i a un poids p_i, un volume v_i et une utilité w_i. Le sac a un poids maximum P et un volume maximum V, et l'on cherche à construire un sous-ensemble d'objets de poids total inférieur ou égal à P, de volume inférieur ou égal à V qui soit d'utilité maximale.

Question 1 : Rappeler la modélisation de ce problème comme un programme linéaire en nombres entiers.

Les étudiantsse sont tous accordés sur le principe de séparation suivant :

Chaque noeud S de l'arbre est associé à un sous-ensemble d'objets $C(S)$ et un sous-ensemble d'objets $R(S)$. On notera $N(S)$ l'ensemble des objets

qui ne sont ni dans $C(S)$ ni dans $R(S)$. Les solutions associées au noeud S sont l'ensemble des sacs qui contiennent tous les objets de $C(S)$ et aucun objet de $R(S)$. Pour séparer un noeud S, on choisit un objet i dans $N(S)$ et on construit les fils S' et S'', définis comme suit :

$$C(S') = C(S) \cup \{i\}, R(S') = R(S), \quad C(S'') = C(S), R(S'') = R(S) \cup \{i\}$$

Question 2 : Justifiez le choix de ce principe de séparation dans une méthode arborescente.

Les étudiants ont proposé plusieurs fonctions d'évaluation par excès :

- $h1(S)$: trier les objets de $N(S)$ par $\frac{w_i}{p_i}$ décroissant et les placer un à un dans le sac contenant déjà les objets de $C(S)$: si l'objet ne rentre pas dans l'une des dimensions (poids ou volume) on ne le met pas et on passe au suivant. $h1(S)$ est alors égal à la somme des utilités des objets dans le sac.
- $h2(S)$: trier les objets de $N(S)$ par $\frac{w_i}{p_i}$ décroissant et les placer un à un dans le sac contenant déjà les objets de $C(S)$ sans tenir compte du volume. Si un objet dépasse, on le découpe de sorte qu'il rentre en poids, et on s'arrête. $h2(S)$ est alors égal à la somme des utilités des objets dans le sac et à la fraction éventuelle d'utilité de l'objet découpé.
- $h3(S)$: trier les objets de $N(S)$ par $\frac{w_i}{p_i+v_i}$ décroissant et les placer un à un dans le sac contenant déjà les objets de $C(S)$: si un objet ne rentre pas dans l'une des dimensions, on le découpe de sorte qu'il rentre dans les deux dimensions et on s'arrête. $h3(S)$ est alors égal à la somme des utilités des objets dans le sac et à la fraction éventuelle d'utilité de l'objet découpé.
- $h4(S)$: trier les objets de $N(S)$ par $\frac{w_i}{p_i+v_i}$ décroissant et considérer un problème une dimension où chaque objet i a un poids p_i+v_i et un sac avec un poids maximum $P+V$. Ajouter les objets un à un dans le sac contenant déjà les objets de $C(S)$ jusqu'à ce que l'un d'eux dépasse (par rapport à ces poids fictifs). Dans ce cas, le découper pour qu'il rentre dans le sac à une dimension. $h4(S)$ est alors égal à la somme des utilités des objets dans le sac et à la fraction éventuelle d'utilité de l'objet découpé.

Question 3 : Discuter pour chaque fonction proposée sa validité : est-ce une fonction d'évaluation par excès et pourquoi ?

Question 4 : Proposer (et justifiez) une évaluation par défaut d'un noeud.

Question 5 : Appliquer votre méthode arborescente à un problème à 5 objets de votre choix. Vous préciserez les évaluations de chaque noeud et indiquerez, le cas échéant vos décisions de troncature.

Exercice 2.7 Un problème de publicité

Un site Internet cherche à se développer en autorisant la publicité sur son site. Chaque annonceur i souhaite être présent pendant une durée minimale de p_i minutes par 24 heures, et rétribue le site Internet au prix de $w_i + v_i * s_i$ où s_i est la durée supplémentaire au delà des p_i minutes où la publicité est présente et v_i le prix de chaque minute supplémentaire.

Les annonceurs exigent également d'être seuls présents sur le site à chaque instant (pas deux publicités d'annonceurs différents en même temps).

Toutefois, le site Internet ne peut placer une publicité trop longtemps, sous peine de faire fuir ses visiteurs, et par conséquent décide de limiter la durée d'une publicité à Q minutes par 24h.

Le site doit donc choisir parmi les n annonceurs les publicités qui passeront durant les prochaines 24h, et pour chacune d'entre elles les durées de passage, afin de maximiser son profit.

Question 1 : Modéliser le problème du site Internet.

Question 2 : En repartant de la définition initale du problème (avec durées variables), supposons que l'on connaisse les publicités choisies, comment résoudre le problème du choix de leur durée ?

Question 3 : Supposons que le site finalement n'autorise pas de dépassement de durée, c'est à dire que chaque publicité est choisie avec sa durée minimale p_i. Comment le problème s'exprime-t-il ?

Question 4 : Proposer une méthode arborescente pour résoudre ce problème (à durées fixes).

Question 5 : Appliquez votre méthode aux données suivantes (où les prix sont estimés en milliers d'euros et les durées en minutes).

i	1	2	3	4	5
w_i	18	5	10	8	12
p_i	700	40	1000	700	400

Exercice 2.8 Méthode exacte et approchée du problème de recouvrement

Cet exercice a pour objectif d'étudier des solutions exactes et approchées pour le problème du recouvrement minimum d'un graphe par un ensemble de sommets. Soit $G = (S, E)$ un graphe non orienté. Un recouvrement de G est un sous-ensemble X de S tel que toute arête de E a au moins l'une de ses extrémités dans X. Etant donné un graphe G, on cherche un recouvrement de cardinalité minimale.

Question 1 : Associons à chaque sommet i une variable bivalente x_i qui vaut 1 si X contient i et 0 sinon. Modéliser le problème à l'aide d'un programme linéaire en nombres entiers.

Question 2 : : Notons $\{u_i\}, \quad 1 \leq i \leq n$ la solution optimale de la relaxation continue de ce programme linéaire. On définit $x_i = 1$ si $u_i \geq 1/2$, $x*_i = 0$ sinon. Montrer que $\{x_i\}, \quad 1 \leq i \leq n$ est une solution réalisable du programme linéaire en nombre entiers.

Question 3 : Déduire de l'inégalité $x_i \leq 2u_i$ une garantie de performance relative de 2 de la solution ainsi construite.

Question 4 : pour tout sommet i, notons $d(i)$ le degré de ce sommet, c'est à dire le nombre d'arêtes ayant i pour extrémité. Montrer que dans tout recouvrement X, la somme des degrés des sommets de X est supérieure ou égale au nombre d'arêtes (noté m) du graphe G.

Question 5 : Supposons que les sommets du graphe soient numérotés par ordre décroissant de degrés, c'est à dire que $d(1) \geq \ldots \geq d(n)$. Soit k le plus petit indice tel que la somme des degrés des sommets de numéro inférieur ou égal à k soit supérieure ou égale à m. Montrer que pour tout recouvrement X, le cardinal de X est supérieur ou égal à k. Qu'en déduit-on si $\{1, \ldots, k\}$ est un recouvrement ?

Question 6 : Soit X un recouvrement de G, et soit i un sommet du graphe. Montrer que si i n'est pas dans X, alors X contient tous les sommets voisins de i dans G (notés $V(i)$). De plus, $X - V(i)$ est un recouvrement du graphe G' construit en retirant de G le sommet i, les sommets de $V(i)$ et toutes les arêtes qui leur sont adjacentes. Réciproquement, montrer que si X' est un recouvrement de G', alors $X' - V(i)$ est un recouvrement de G.

Question 7 : Soit X un recouvrement de G, et soit i un sommet du graphe. Montrer que si i est dans X, alors $X - i$ est un recouvrement du graphe G' construit en retirant de G le sommet i et toutes ses arêtes adjacentes.

Réciproquement, montrer que si X''' est un recouvrement de G'', alors $X'' - \{i\}$ est un recouvrement de G.

Les questions précédentes ont pour objectif de définir les différents éléments d'une méthode arborescente pour résoudre le problème du recouvrement minimal. A un noeud s de l'arborescence seront associés deux sous-ensembles disjoints de sommets du graphe : C(s) l'ensemble des sommets choisis, et R(s) l'ensemble des sommets rejetés. Ainsi, le noeud s correspondra à l'ensemble des recouvrements X contenant $C(s)$ et ne contenant pas $R(s)$. En appliquant les règles décrites dans les questions 6 et 7, on associe également à s un sous graphe $G(s)$ de sorte que : si Y est un recouvrement de $G(s)$, alors $Y \cup C(s)$ est un recouvrement de G ne contenant pas $R(s)$. Réciproquement, si X est un recouvrement de G contenant $C(s)$ et pas $R(s)$, alors $G - C(s)$ est un recouvrement de $G(s)$.

Question 8 : Montrer comment utiliser la question 5 pour construire une évaluation par défaut du meilleur recouvrement associé à un noeud s.

Question 9 :le principe de séparation d'un noeud s consiste à choisir dans $G(s)$ un sommet i de degré maximum, et de construire deux noeuds s' et s''. s' contient les recouvrements appartenant au noeud s qui ne contiennent pas i, et s'' les recouvrements appartenant au noeud s qui contiennent i. En utilisant au mieux les règles de simplification induites par les questions 6 et 7, indiquez comment construire $C(s'), R(s'), G(s')etC(s''), R(s''), G(s'')$.

Question 10 : : Appliquez la méthode arborescente à un exemple de votre choix. On précisera pour chaque noeud s de l'arbre - son évaluation, les sous ensembles C(s) et R(s), et l'on dessinera le graphe G(s).

Chapitre 3

Programmation dynamique

Exercice 3.1 Programmation dynamique

Développez l'algorithme de programmation dynamique vu en cours pour le problème de sac à dos ci-dessous avec $P = 7$. On rappellera l'algorithme de construction d'une solution optimale.

i	1	2	3	4	5	6
w_i	27	32	12	5	16	18
p_i	3	4	2	1	4	5

Exercice 3.2 Sac à dos inversé

On considère le problème suivant : n objets, de poids respectifs p_i et de regret r_i. On se donne également un entier R. On appelle regret d'un sac la somme des regrets des objets **qui ne sont pas** dans le sac. On cherche à construire un sac de poids minimal et de regret inférieur ou égal à R.

Question 1 : Modéliser ce problème à l'aide d'un programme mathématique.

Question 2 : On cherche à définir un schéma de programmation dynamique pour ce problème. Pour cela on définit la phase i comme étant celle du choix pour l'objet i, et l'état du système comme le regret des objets rejetés du sac dans les phases antérieures. Indiquer pour un couple (i, E) les décisions possibles à partir de cet état à la phase i, leur cout immédiat et les états résultants de ces décisions pour la phase suivante.

Question 3 : Notons $F_k(E)$ le poids minimum d'un sac de regret inférieur ou égal $R - E$ parmi les objets $\{k, \ldots, n\}$. Comment se définit la solution

du problème initial par rapport à cette notation ?

Question 4 : Que vaut $F_n(E)$?

Question 5 : Etablir l'équation de récurrence satisfaite par les F_k.

Question 6 : En déduire un algorithme pour résoudre le problème dont vous donnerez la complexité.

Question 7 : Appliquez l'algorithme au problème suivant :

$R = 45$

i	1	2	3	4	5
p_i	3	4	5	4	2
r_i	15	18	20	12	6

Exercice 3.3 Stockage d'objets

On considère un ensemble d'objets $\mathcal{F} = \{1, \ldots, n\}$ de tailles respectives (en octets) $p_1, \ldots, p_n$. On dispose de deux sacs de taille P. Chaque objet i, s'il est placé dans le contenant 1, rapporte a_i, alors qu'il rapporte b_i s'il est placé dans le sac 2. Les $a_1, \ldots, a_n$ et $b_1, \ldots, b_n$ sont des données du problème. On cherche à choisir des objets de sorte qu'ils rentrent dans les sacs et qu'ils rapportent le plus possible.

On remarquera qu'il est possible que la taille des sacs ne soit pas suffisante pour stocker tous les objets. Ainsi, chaque objet peut se trouver soit dans le sac 1, soit dans le sac 2, soit n'être pas emporté.

Question 1 : En notant x_i la variable de décision qui vaut 1 si l'objet est stocké dans le premier contentant, 0 sinon, et y_i la variable de décision qui vaut 1 si l'objet est stocké dans le sac 2, et 0 sinon, modéliser le problème à l'aide d'un programme linéaire à variables bivalentes.

Question 2 : Soient $m_1, m_2 \leq P$. Appelons $F_k(m_1, m_2)$ le profit maximum pouvant être espéré d'un choix d'objets parmi $\{k, \ldots, n\}$ stockés dans les deux sacs supposés de tailles respectives m_1, m_2. Exprimer la solution du problème initial par rapport à cette notation.

Question 3 : Calculez $F_n(m_1, m_2)$, et établir l'équation de récurrence permettant de calculer $F_k(m_1, m_2)$ pour tout $1 \leq k < n$ et toutes les valeurs admissibles de m_1, m_2.

Question 4 : Quelle est la complexité de l'algorithme de programmation

dynamique qui en résulte ?

Exercice 3.4 Régime alimentaire

Afin d'anticiper l'arrivée de l'été et son cortège de régimes amincissants, une entreprise agro-alimentaire cherche à lancer une nouvelle gamme de "'paniers repas hyperprotéinés"' à partir de produits de base en portions de 50g. Il dispose de n produits de base, sélectionnés pour leur succès auprès des consommateurs et leur prix relativement bas. Chaque produit i est caractérisé par une quantité p_i de protéines, une quantité g_i de glucides et une quantité l_i de lipides, supposés des nombres entiers.

Il s'agit de sélectionner des produits de sorte que la quantité totale de lipides soit limitée par une quantité L et la quantité de glucides par une quantité G, tout en cherchant à avoir un maximum de protéines.

L'entreprise s'autorise à avoir, dans son panier repas, un ou deux exemplaires des produits de base. Exemple : si les produits de base sont l'oeuf, la pomme, la tomate, le yaourt, et l'artichaut, un panier pourra comporter deux oeufs, une pomme, un artichaut.

Question 1 : Modéliser le problème de l'entreprise agro-alimentaire avec un programme mathématique.

Définissons $F_k(a, b)$ la quantité maximale de protéines d'un panier composé à partir des produits du numéro k au numéro n, qui contienne au maximum a glucides b lipides.

Question 2 : Décrire le schéma de programmation dynamique sous-tendu par cette définition : Quelles sont les phases, les états, les décisions, les transitions, l'état initial, et les profits immédiats ?

Question 3 : Que vaut $F_n(a, b)$?

Question 4 : Etablir l'équation de récurrence permettant de calculer $F_k(a, b)$

Question 5 : En déduire un algorithme permettant de calculer un panier optimal. Vous décrirez l'algorithme et calculerez sa complexité.

Exercice 3.5 Ordonnancement sur deux machines de profit maximum

Un atelier de production dispose de deux machines qui peuvent fonctionner en parallèle. En début de mois un ensemble de commandes (tâches) est proposé à l'atelier pour réalisation. La plupart du temps, il y en a trop pour être réalisées. Le responsable d'atelier doit donc décider quelles sont les commandes qui sont acceptées par son atelier (celles qui peuvent être réalisées

dans le mois), et sur quelle machine elles s'effectuent. Pour une tâche i la durée n'est pas la même selon la machine qu'on utilise (durées respectives a_i et b_i). De plus, le fait d'accepter une tâche i induit un profit w_i pour l'entreprise qui peut la facturer au client. On cherche donc à ordonnancer les tâches sur les deux machines de sorte que la durée des tâches sur chacune des machine soit inférieure à la durée ouvrable du mois M en maximisant le profit de l'atelier.

Question 1 : Modéliser ce problème à l'aide d'un programme mathématique.

On définit $F_k(A, B)$ le profit maximal d'un ordonnancement des tâches k à n pour lequel la durée des tâches sur la machine 1 est inférieure ou égale à A, celle sur la machine 2 est inférieure ou égale à B.

Question 2 : Décrire le schéma de programmation dynamique induit par cette définition (phases, états, décisions, transitions, etc)

Question 3 :Donner la valeur de $F_n(A, B)$ en fonction des données du problème et de A, B

Question 4 : Indiquez l'équation de récurrence satisfaite par $F_k(A, B)$.

Question 5 : Ecrire l'algorithme qui découle de ces égalités permettant de trouver un ordonnancement optimal. On précisera sa complexité.

Exercice 3.6 Un problème d'énergie

Afin de satisfaire la demande croissante en énergie dans les prochaines années, une entreprise dans le secteur de l'énergie se pose des questions sur son plan d'investissement.

Elle dispose des prévision de pics de consommation sur les 15 ans qui viennent. On note D_i la demande en énergie (puissance pic) pour l'année i, supplémentaire par rapport au parc existant de centrales en tout genre.

Deux types d'investissements nouveaux sont envisagés :

1. Des centrales nucléaires. Chaque centrale nécessite un investissement initial en euros important, noté I_N mais a une puissance de production très importante P_N pour un coût de fonctionnement annuel de c_N euros : chaque fois qu'une centrale est construite, elle coûte donc c_N pour toutes les années suivantes.
2. Des éoliennes. Chaque éolienne nécessite un investissement I_E, avec une puissance assez faible P_E et un coût de fonctionnement estimé négligeable.

En supposant ici qu'une centrale (ou une éolienne) dont on décide la mise en production l'année i peut bien produire un pic de P_N(resp. P_E) pour

l'année i en cours, comment choisir, pour chaque année, les investissements à réaliser pour permettre de satisfaire la demande, sachant qu'on ne peut pas investir dans plus d'une centrale nucléaire chaque année, et en tâchant d'optimiser le coût total (investissements + fonctionnement) ?

Question 1 : Modéliser ce problème à l'aide d'un programme mathématique.

On définit $F_k(A, B)$ le coût minimum nécessaire pour satisfaire la demande des années $k, k+1, ..15$, sachant qu'on a déjà investi dans A centrales nucléaires et B éoliennes les années précédentes.

Question 2 : Décrire le schéma de programmation dynamique induit par cette définition (phases, états, décisions, transitions, etc)

Question 3 :Donner la valeur de $F_{15}(A, B)$

Question 4 : Indiquez l'équation de récurrence satisfaite par $F_k(A, B)$.

Question 5 : Ecrire l'algorithme qui découle de ces égalités. On précisera sa complexité.

Exercice 3.7 Ordonnancement à vitesse variable et consommation d'énergie

On se donne une suite de n tâches à effectuer sur une machine. Cette machine possède deux modes de fonctionnement différents. Lorsqu'une tâche i est effectuée selon le mode 1, elle a une durée a_i, et selon le mode 2 elle dure b_i. Le choix d'un mode pour effectuer une tâche i engendre un coût énergétique : le choix du mode 1 coûte $\frac{c_1}{a_i}$, et le choix du mode 2 $\frac{c_2}{b_i}$.

On cherche à associer à chaque tâche un mode de la machine selon lequel elle sera exécutée, de sorte que la durée totale de l'ordonnancement des tâches (on supposera que l'on fait les tâches en séquence, une fois qu'on a décidé du mode de chacune) ne dépasse pas une valeur donnée D, afin que la dépense totale d'énergie soit minimale. S'il n'existe aucune solution réalisable, on dira par convention que la dépense minimale d'énergie vaut $+\infty$.

Question 1 : Modéliser ce problème à l'aide d'un programme mathématique.

On souhaite décrire un schéma de programmation dynamique pour résoudre ce problème. Soit $F_k(E)$ le coût minimal en énergie d'un ordonnancement de durée inférieure ou égale à E pour les tâches $\{k, \ldots, n\}$.

Question 2 : Comment calculer $F_n(E)$?

Question 3 : Etablir l'équation de récurrence satisfaite par les F_k.

Question 4 : En déduire un algorithme de programmation dynamique qui

construit une solution optimale dont on précisera la complexité.

Question 5 : Appliquez cet algorithme aux données suivantes :

Tâches :

i	1	2	3	4	5	6
a_i	3	1	2	2	4	3
b_i	2	1	1	1	3	2

machine : $c_1 = 1, c_2 = 2, D = 12$

Exercice 3.8 Stockage d'objets

On considère un ensemble d'objets $\mathcal{F} = \{1, \ldots, n\}$ de tailles respectives $p_1, \ldots, p_n$. On dispose de deux sacs de taille P. Chaque objet i, s'il est placé dans le contenant 1, rapporte a_i, alors qu'il rapporte b_i s'il est placé dans le sac 2. Les $a_1, \ldots, a_n$ et $b_1, \ldots, b_n$ sont des données du problème. On cherche à choisir des objets de sorte qu'ils rentrent dans les sacs et qu'ils rapportent le plus possible.

On remarquera qu'il est possible que la taille des sacs ne soit pas suffisante pour stocker tous les objets. Ainsi, chaque objet peut se trouver soit dans le sac 1, soit dans le sac 2, soit n'être pas emporté.

Question 1 : En notant x_i la variable de décision qui vaut 1 si l'objet est stocké dans le premier contentant, 0 sinon, et y_i la variable de décision qui vaut 1 si l'objet est stocké dans le sac 2, et 0 sinon, modéliser le problème à l'aide d'un programme linéaire à variables bivalentes.

Question 2 : Soient $m_1, m_2 \leq P$. Appelons $F_k(m_1, m_2)$ le profit maximum pouvant être espéré d'un choix d'objets parmi $\{k, \ldots, n\}$ stockés dans les deux sacs supposés de tailles respectives m_1, m_2. Exprimer la solution du problème initial par rapport à cette notation.

Question 3 : Calculez $F_n(m_1, m_2)$, et établir l'équation de récurrence permettant de calculer $F_k(m_1, m_2)$ pour tout $1 \leq k < n$ et toutes les valeurs admissibles de m_1, m_2.

Question 4 : Quelle est la complexité de l'algorithme de programmation dynamique qui en résulte ?

Exercice 3.9 Séquence à l'heure sur une machine de coût minimal

On considère un ensemble $\mathcal{T}$ de n tâches à faire sur une machine. Chaque tâche i a une durée p_i et une date d'échéance (c'est à dire une date à laquelle on souhaite que la tâche se termine) notée d_i, et un coût noté c_i. Le coût correspond à la pénalité que doit payer l'entreprise lorsque la tâche i est en

retard.

On suppose que les tâches sont numérotées de sorte que $d_1 \leq d_2 \leq \ldots \leq d_n$.

Pour illustrer le problème de cet exercice on s'appuie sur l'exemple numérique suivant :

i	1	2	3	4	5	6	7	8
p_i	3	4	6	5	2	8	10	4
d_i	5	5	9	12	14	18	24	24
c_i	3	2	1	2	3	1	4	1

On appelle séquence à l'heure par rapport à une date origine t_0 une suite de k tâches (k est un entier) notée $S = (i_1, \ldots, i_k)$ avec $i_1 \leq \ldots \leq i_k$ telle que si l'on exécute les tâches $i_1, \ldots, i_k$ sur la machine dans cet ordre à partir de la date t_0, elles sont toutes à l'heure (c'est à dire se terminent avant ou à leur date d'échéance).

A titre d'exemple, $S = (1, 3, 5, 7)$ est une séquence à l'heure par rapport à la date origine 0. en effet, en ordonnançant ces tâches dans cet ordre, 1 se termine à 3, 3 se termine à 9, 5 se termine à 11, 7 se termine à 21. Par contre $S = (1, 2)$ n'est pas une séquence à l'heure, puisqu'en faisant la tâche 1 puis la tâche 2, cette dernière est en retard (elle se termine à 7 alors que son échéance est 5).

Soit S une séquence à l'heure parmi l'ensemble des tâches $\mathcal{T}$. On note $P(S)$ la durée de S c'est à dire la somme des durées des tâches de la séquence. On note $C(S)$ le coût de la séquence S, qui correspond à la somme des coûts des tâches qui ne sont pas dans la séquence S :

$$C(S) = \sum_{i \in \mathcal{T} - S} c_i$$

Pour la séquence précédente, $P(S) = 21$ et $C(S) = c_2 + c_4 + c_6 + c_8 = 6$.

On s'intéresse à la constitution d'une séquence a l'heure de coût minimal pour la date origine 0.

Question 1 : soit E un entier, Notons $F_k(E)$ le coût minimal d'une séquence à l'heure parmi les tâches de l'ensemble $\mathcal{T}_k = \{k, \ldots, n\}$ pour la date origine E. Indiquer les valeurs de k et de E pour lesquelles $F_k(E)$ est la solution du problème de départ.

Question 2 : Exprimer un schéma de programmation dynamique (etats,

décisions, transitions, coûts immédiats) qui corresponde au $F_k(E)$ défini plus haut.

Question 3 : Que vaut $F_n(E)$?

Question 4 : Exprimer la relation de récurrence entre les F_k.

Question 5 : En déduire un algorithme de programmation dynamique. On précisera sa complexité.

Question 6 : Calculer la solution du problème numérique.

Exercice 3.10 Ordonnancement à deux machines de coût minimal

On considère un ensemble de n tâches à faire sur deux machines différentes. Chaque tâche i a une durée a_i sur la première machine, b_i sur la seconde. Malheureusement, il n'est pas toujours possible de faire toutes les tâches. Chaque tâche i , si on ne l'exécute pas coûte c_i. On cherche à déterminer pour chaque tâche si on l'exécute et sur quelle machine, de sorte que l'usage de chaque machine ne dépasse pas la durée de la journée J et que le coût pour l'entreprise soit minimum.

Question 1 : Modéliser ce problème à l'aide d'un programme linéaire en nombres entiers.

Appelons $F_k(A, B)$ le coût minimal d'un ordonnancement des tâches choisies parmi $\{k, \ldots, n\}$ qui dure au plus A sur la machine 1 et B sur la machine 2.

Question 2 : Pour quelles valeurs de k, A, B la notation $F_k(A, B)$ désigne-t-elle la solution optimale du problème de la question 1 ?

Question 3 : Que vaut $F_n(A, B)$?

Question 4 : En vous appuyant sur la définition de $F_k(A, B)$ définissez un schéma de programmation dynamique (phases, états, transitions, décisions, état inital, coûts immédiats) pour le problème.

Question 5 : Rappeler l'équation de récurrence générale de la programmation dynamique arrière vue en cours et spécialisez là pour le schéma de la question précédente.

Question 6 : En déduire un algorithme de programmation dynamique pour le problème ci-dessus.

Exercice 3.11 Ordonnancement juste à temps

On considère le problème d'ordonnancement suivant : on a n tâches à

effectuer dans l'ordre $1, 2, \ldots, n$ sur une machine qui ne peut traiter qu'une tâche à la fois. Chaque tâche i est caractérsée par plusieurs paramètres entiers : sa durée, notée p_i, sa date d'échéance, notée d_i, une pénalité d'avance notée a_i et une pénalité de retard notée r_i.

Un ordonnancement permet de définir les dates de fin f_i des tâches i (en garantissant qu'elles s'effectuent bien dans l'ordre de leur numérotation). Le coût d'un ordonnancement est défini en calculant la somme des coûts d'avance et de retard des tâches. Pour une tâche i, si $f_i < d_i$ alors la tâche est en avance et coûte a_i. Si $f_i > d_i$ la tâche est en retard et coûte r_i. Sinon $f_i = d_i$, la tâche i est à l'heure et ne coûte rien.

On cherche à ordonnancer les tâches de sorte que la durée totale (c'est à dire la date de fin de la tâche n) soit inférieure à une valeur T donnée, en minimisant le coût de l'ordonnancement.

L'ordonnancement d'une telle suite de tâches peut conduire a des temps morts : par exemple, si l'on considère les deux tâches 1 et 2 avec

$T = 10$

i	p_i	d_i	a_i	r_i
1	3	4	2	3
2	4	10	3	4

Dans cet exemple l'ordonnancement optimal consistera à commencer 1 à la date 1 pour qu'elle se finisse exactement à 4 et 2 à la date 6 pour qu'elle se finisse exactement à 10. L'ordonnancement qui consiste à commencer la tâche 1 à la date 2 et la tâche 2 à la date 5 conduit à un coût de 3 pour la tâche 1 qui est en retard, et 3 pour la tâche 2 qui est en avance (se termine à 9). D'où un coût total de 6.

Question 1 : Définissons une variable $x_i \in \{0, 1\}$. Supposons que la date de fin f_i et la variable x_i vérifie les inégalités suivantes :

$$x_i - T(1 - x_i) \leq d_i - f_i \leq T x_i$$

Montrer que la tâche i est en avance si et seulement si $x_i = 1$.

Question 2 : Définissons similairement une variable $y_i \in \{0, 1\}$ qui vérifie les inégalités :

$$y_i - T(1 - y_i) \leq f_i - d_i \leq T y_i$$

. Que modélise la variable y_i ?

Question 3 : Etablir un programme mathématique qui modélise le problème de l'ordonnancement juste à temps de la suite de tâches $\{1, \ldots, n\}$.

On cherche maintenant à définir un algorithme de programmation dy-

namique qui puisse résoudre le problème. Définissons $F_k(s)$ le coût minimal d'un ordonnancement de la suite de tâches $\{k, \ldots, n\}$ qui commence au moins à la date s et se termine au plus tard à la date T.

Question 4 : Quelle est la valeur de k et de s de sorte que le calcul de $F_k(s)$ corresponde à la définition du problème initial ?

Question 5 : Comment calculer $F_n(s)$?

Question 6 : Montrer que si $s + p_k > d_k$ alors $F_k(s) = r_k + F_{k+1}(s + p_k)$.

Question 7 : Supposons que $s + p_k \leq d_k$ et considérons un ordonnancement solution de $F_k(s)$.

- Montrer que dans cet ordonnancement la tâche k ne peut pas être en retard
- Montrer que si dans cet ordonnancement, la tâche k est en avance, on peut supposer qu'elle commence à la date s (si elle commençait plus tard, on pourrait l'avancer sans modifier le coût de l'ordonnancement).
- indiquer la date à laquelle k commence si elle est faite à l'heure.

Question 8 : Déduire de la question précédente l'équation de récurrence satisfaite par $F_k(s)$.

Question 9 : Ecrire l'algorithme permettant de trouver l'ordonnancement optimal et calculer sa complexité.

Chapitre 4

Algorithmes approchés

Exercice 4.1 Algorithmes approchés

Considérons un problème de minimisation sous la forme :

$$\begin{cases} \min f(X) \\ X \in D \end{cases}$$

On dispose d'un algorithme $\mathcal{A}$ qui résoud ce problème en fournissant une solution réalisable mais pas toujours optimale.

Question 1 : Que signifie l'assertion suivante : "'l'algorithme $\mathcal{A}$ est 2-approché ?"'

Question 2 : Indiquez un algorithme vu en cours qui a cette propriété.

Exercice 4.2 Algorithmes approchés et méta-heuristiques

Considérons un problème de minimisation sous la forme :

$$\begin{cases} \min f(X) \\ X \in D \end{cases}$$

On dispose d'un algorithme $\mathcal{A}$ qui résoud ce problème en fournissant une solution réalisable mais pas toujours optimale.

Question 1 : Que signifie l'assertion suivante : "'l'algorithme $\mathcal{A}$ a une approximation relative de 3 ?"'

Question 2 : Quelles sont les principes de fonctionnement d'un algorithme génétique ?

Exercice 4.3 Bin Packing

Dans le problème de bin Packing, on dispose d'un ensemble de n objets de poids respectifs $p_1, \ldots, p_n$. Les objets doivent être placés dans un nombre minimum de boites de capacité P (poids maximum qu'elles peuvent contenir).

Question 1 : Rappelez le fonctionnement de l'algorithme First Fit vu en cours, et appliquez le à l'exemple suivant, pour $P = 5$:

i	1	2	3	4	5	6	7	8	9	10
p_i	1	1	1	2	3	3	2	4	4	4

Question 2 : L'algorithme appelé First Fit Decreasing fonctionne comme l'algorithme First Fit, à ceci près que les objets sont préalablement triés par ordre de poids décroissant. Executer cet algorithme pour l'exemple.

Question 3 : Montrer que le nombre de boites générées par First Fit Decreasing est strictement inférieur à deux fois le nombre de boites optimales (on pourra utiliser le résultat vu en cours).

Exercice 4.4 Réseau de communication

On considère un réseau de communication donné par un graphe non orienté G, comportant un ensemble de n sommets $S = \{1, \ldots, n\}$ et un ensemble d'arêtes A. Ce réseau étant soumis à des attaques, l'entreprise qui le gère souhaite placer des équipements qui permettent d'analyser le trafic qui s'écoule dans le réseau. Ces équipements doivent être localisés sur certains sommets du graphe. Lorsqu'un équipement est placé en un sommet x, il peut analyser tout le trafic circulant sur les liens de communication reliés au sommet x.

Comme ces équipements sont coûteux, il s'agit pour l'entreprise de placer un nombre minimum d'équipements qui lui permette d'analyser le trafic dans chaque lien de communication.

Question 1 : Modéliser ce problème avec un programme mathématique. Quel est le nom vu en cours de ce problème ?

Question 2 : Soit T un ensemble d'arêtes deux à deux disjointes de G (c'est à dire avec aucun sommet commun). Montrer que le nombre d'équipements du graphe G sera nécessairement au moins égal à $|T|$.

Question 3 : Soit l'algorithme suivant $\mathcal{A}$: choisir une arête a_1, puis supprimer du graphe toutes les arêtes adjacentes aux extrémités de a_1, et recom-

mencer jusqu'à ce que le graphe soit vide. Notons $N^{\mathcal{A}}$ le nombre d'arêtes choisies par l'algorithme. Notons N^* le nombre optimal d'équipements (celui qu'on recherche). Montrer que $N^{\mathcal{A}}$ une évaluation par défaut de N^*.

Question 4 : Proposer une heuristique (qui vous semble la plus intelligente possible) permettant de construire une solution réalisable du problème.

Remarquons que dans ce problème, si l'on sait qu'un équipement est placé au sommet x, alors trouver la meilleure solution où un équipement est placé en x consiste à résoudre le problème pour le graphe G dont on a retiré les arêtes qui passent par x (puisque ces arêtes sont couvertes par l'équipement en x). A contrario, si l'on s'intéresse aux solutions où aucun équipement n'est placé au sommet x alors on sait qu'un équipement sera placé dans chaque voisin de x, et donc trouver la meilleure solution où aucun équipement n'est placé en x consiste à résoudre le problème sur le graphe G dont on a retiré le sommet x et ses voisins, et à y ajouter les voisins de x.

Question 5 : En déduire une méthode arborescente pour le problème. Vous préciserez : la définition des noeuds, le principe de séparation, et les évaluations par excés et par défaut.

Question 6 : Appliquer cette méthode à un graphe de votre choix avec 7 noeuds et une dizine d'arêtes. On précisera pour chaque noeud ses évaluations, et on argumentera les décisions de troncature éventuelles.

Question 7 : Considérons l'ensemble d'arêtes choisies par l'algorithme $\mathcal{A}$. Notons $X^{\mathcal{A}}$ l'ensemble des sommets extrémités des arêtes choisies. Montrer que $X^{\mathcal{A}}$ est une solution réalisable du problème de départ (c'est à dire qu'en choisissant de placer des équipements aux deux extrêmités de chaque arête choisie, on peut effectivement surveiller l'ensemble du réseau).

Question 8 : Construire $X^{\mathcal{A}}$ pour le graphe que vous avez choisi dans l'une des questions précédentes.

Question 9 : Montrer que $|X^{\mathcal{A}}| \leq 2N^*$

Question 10 : Qu'en déduit-on pour l'algorithme $\mathcal{A}$?

Exercice 4.5 Recouvrement de sommets

Etant donné un graphe non orienté $G = (S, E)$, un recouvrement est un sous-ensemble des sommets X de sorte que pour toute arête $\{x, y\} \in E$, $x \in X$ ou $y \in X$. On s'intéresse à la recherche d'un recouvrement de cardinalité minimale. On notera X^* un recouvrement optimal.

Question 1 : Soit U un sous-ensemble d'arêtes du graphe qui n'ont aucune extrémité commune. Montrer que pour tout recouvrement X de G, $|X| \geq |U|$

(le nombre de sommets de X est au moins égal au nombre d'arêtes de U).

Considérons l'algorithme suivant qui construit un ensemble de sommets X et un ensemble d'arêtes U :
$E' = E$
$X = \emptyset$
$U = \emptyset$
tant que $E' \neq \emptyset$

choisir $\{x, y\} \in E'$

ajouter x et y à X

ajouter l'arête $\{x, y\}$ à U

retirer de E' toutes les arêtes adjacentes à x et à y

Question 2 : Tracer un graphe à 6 sommets et 10 arêtes et appliquez cet algorithme à votre graphe.

Question 3 : Montrer que l'ensemble X construit par l'algorithme est bien un recouvrement de G, et que $|X| = 2 * |U|$

Question 4 : Montrer qu'à la fin de l'algorithme, les arêtes de U n'ont aucune extrémité commune

Question 5 : En déduire que l'algorithme ci-dessus est un algorithme approché avec un rapport d'approximation relative égal à 2.

Exercice 4.6 Stable

Soit G un graphe non orienté. On appelle stable de G un sous-ensemble X des sommets de ce graphe, qui ne sont reliés par aucune arête. On cherche à calculer le stable de cardinalité maximum d'un graphe. On notera X^* un stable optimal.

Question 1 : Modéliser ce problème.

Question 2 : Dessiner un graphe non orienté à 10 sommets et donnez-en un stable maximum.

On considère l'algorithme suivant, qui construit un stable de G : Choisir un sommet x et le mettre dans X. Retirer de G x et tous les voisins de x. Recommencer jusqu'à ce que le graphe G soit vide.

Question 3 : Appliquer cet algorithme à votre graphe.

Soit le graphe H_n suivant : il possède un sommet a et n sommets

$b_1, \ldots, b_n$. Il y a une arête b_i, b_{i+1} pour tout $i < n$ et une arête entre a et chaque sommet b_i.

Question 4 :Quel est cardinal maximum d'un stable de ce graphe, selon la valeur de n ?

Question 5 :En supposant que l'algorithme ci-dessus commence par mettre dans X le sommet a, quel est le stable construit ?

Question 6 :L'algorithme est-il un algorithme approché avec garantie de performance relative ?

Exercice 4.7 Expérimentation et Bin Packing

Considérons la comparaison expérimentale de deux heuristiques pour le problème de bin packing. Rappelons que ce problème est défini par n objets, chaque objet i de poids p_i, et que ces objets doivent être placés dans des boites, chaque boite étant de poids maximal P. Le problème est de déterminer le nombre minimum de boites nécessaires pour placer l'ensemble des objets.

Question 1 :Considérons

$$N^* = \left\lceil \frac{\sum_{i=1}^{n} p_i}{P} \right\rceil$$

. Que représente cette quantité par rapport au probème étudié ?

Question 2 : Rappeler le principe de l'algorithme first fit étudié en cours.

Question 3 :On compare le nombre de boites obtenu par First Fit et celui, que nous nommerone Random Fit, obtenu en plaçant itérativement dans la première boite possible un objet pris au hasard dans la liste les ob- jets non encore placés (quitte à ouvrir une nouvelle boite si celui-ci ne rentre dans aucune boite déjà ouverte). L'expérimentation consiste 'a générer aléatoirement des objets et une taille de boite P et à exécuter les deux heuristiques (First Fit et Random Fit) puis à comparer pour chacun des jeux de données, le nombre de boites obtenues. Cette comparaisona donne un résultat qui ne permet pas de distinguer l'efficacité des deux heuristiques. Pouvez-vous l'expliquer ?

Question 4 : Supposons qu'on effectue une comparaison entre First Fit (FF) et l'algorithme First Fit Decreasing (FFD) qui consiste à trier d'abord les objets par poids déecroissant avant d'appliquer First Fit. Quelle sont les mesures expérimentales qui permettent d'établir la supériorité de FFD sur FF, et de quantifier l'efficacité de FFD ?

Chapitre 5

Programmation

Exercice 5.1 Méthodes exactes pour sac à dos inversé

Etant donnés n objets, l'objet i étant caractérisé par un poids p_i et une utilité w_i. On se donne une valeur W, et l'on cherche à déterminer un sous ensemble d'objets de poids minimum dont la somme des utilités est au moins W.

Question 1 : Définir et programmer une méthode arborescente pour ce problème de sac à dos inversé :Indiquez ici les principaux éléments de la méthode arborescente que vous avez choisie (Définition des noeuds/solutions, évaluation par défaut, par excès, parcours de l'arbre).

Question 2 : Utilisez cette méthode arborescente pour résoudre l'exemple suivant :

$W = 45$

i	1	2	3	4	5	6	7
p_i	3	4	5	4	2	5	3
w_i	15	18	20	12	6	20	15

Question 3 : Programmer la résolution en programamtion dynamique du sac à dos inversé, et comparer les temps d'exécution entre les deux méthodes sur des jeux de données aléatoires.

Exercice 5.2 Plan d'expérimentation

Question 1 : On dispose d'un algorithme $\mathcal{A}$ qui permet de fournir une solution réalisable d'un problème de minimisation, mais pas toujours optimale. On peut également calculer avec un algorithme $\mathcal{B}$ une borne inférieure de

la solution optimale du problème. Décrire un plan d'expérimentation permettant de mettre en évidence la capacité de l'algorithme $\mathcal{A}$ à fournir de "bonnes solutions".

Exercice 5.3 Ordonnancement 1 machine

On s'intéresse à la résolution du problème d'ordonnancement suivant : On se donne n tâches caractérisées par des durées p_i, des dates d'échéance d_i et des profits w_i. Ces tâches doivent être ordonnancées sur une seule machine. Le profit d'un ordonnancement est la somme des profits des tâches qui se terminent avant ou à leur date d'échéance.

Question 1 : Proposez les éléments d'une méthode arborescente pour ce problème. Vous pourrez utiliser les propriétés qui ont permis de définir le schéma de programmation dynamique vu en cours pour ce même problème. Vous pourrez également vous inspirer des méthodes arborescentes présentées dans le document de cours.

Vous préciserez dans le document d'analyse vos choix en matière de :

- Définition des sous-ensembles de solutions associés aux noeuds.
- Définition de l'opérateur de séparation
- Définition de la fonction d'évaluation par excès.
- Définition d'un algorithme permettant d'obtenir en chaque noeud une solution particulière et donc une évaluation par défaut.

Il y a plusieurs solutions possibles. Vous pouvez faire plusieurs propositions, en tâchant de d'évaluer a-priori l'intérêt de chacune.

Question 2 : Implémentez votre méthode arborescente. Vous travaillerez à 2 groupes (ou plus) pour constituer un fichier commun de jeux d'essai (données particulières de problèmes). Vous pouvez si vous le souhaitez utiliser une génération aléatoire de données.

Il faudra des problèmes de taille différentes, quelques-un de l'ordre d'une dizaine, d'une trentaine et d'une cinquantaine de tâches. Une quinzaine de problème (5 de chaque sorte) est un minimum.

Les groupes expérimenteront différents paramètres sur leur méthode, en matière de règle de parcours (profondeur, largeur, meilleur), ou de fonctions d'évaluation, ou encore de séparation. Vous devez vous répartir le travail de manière équitable (et le groupe de 4 en fournira un peu plus que les autres, par exemple en implémentant également la méthode de programmation dynamique).

Pour chacun des jeux d'essai et des paramètres choisis, vous évaluerez (et comparerez) le nombre de noeuds développés par la méthode, et le temps de calcul, si vous pouvez.

Le dossier d'analyse (un par groupe) comprendra une explication succinte du choix des structures de données et d'algorithmique pour l'implémentation de la méthode, ainsi qu'une étude comparative des résultats des différents tests menés par au moins deux groupes. Il précisera également le rôle de chacun dans l'élaboration du projet.

L'objectif est de répondre à cette question : votre méthode est-elle efficace/inefficace ? Et à votre avis pourquoi ?