Programmation par contrainte: évaluation pratique
5-SDBD + mastère VALDOM, INSA Toulouse, janvier 2023

[Informations & cahier des charges]

— Chaque groupe est constitué de deux ou trois étudiants.
— Le code source doit être bien documenté. Chaque variable/contrainte doit être bien expliquée.
— Le travail doit être soumit sous format notebook en respectant l'ordre de questions. Le notebook doit être exécuté à l'avance.
— L'utilisation de ChatGPT est considérée comme une forme de plagiat.
— La date limite de dépôt sur moodle est fixée au **26 janvier à 23h59**.

Première partie

On utilise tout au long de ce sujet deux ensembles d'agents $H = \{h_1, \dots h_n\}$ et $F = \{f_1, \dots f_n\}$. Chaque agent $h_i \in H$ exprime ses préférences de couplage avec un agent de F à travers une liste $L(h_i)$ contenant les éléments de F sans duplication. Pour tout $k \in [1, n-1]$, h_i préfère être couplé avec le $k$ème agent dans $L(h_i)$ que le $k+1$ème agent. De la même façon, chaque agent $f_j \in F$ exprime ses préférences de couplage avec un agent de H à travers une liste $L(f_j)$.

Un tuple (h, f) est appelé un couple si $h \in H$ et $f \in F$. Soit M un ensemble de couples. Un agent a préfère un agent b à sa situation dans M si a n'appartient à aucun couple dans M ou si a préfère b à son partenaire dans M. Un couple $(h, f) \in M$ bloque M si h préfère f à sa situation dans M et f préfère h à sa situation dans M.

Un mariage stable est un ensemble de couples M tel que chaque agent est couplé avec un seul agent et M n'admet aucun couple bloquant. On note qu'une instance de ce problème peut admettre plusieurs solutions.

La table 1 présente une instance avec $n = 4$. La séquence $L[h_i]$ est la liste d'agents dans F ordonnée selon les préférences de h_i. Dans cet exemple, h_3 préfère f_2 à f_4, et f_4 à f_1, etc.
$M_1 = \{(h_1, f_2), (h_2, f_1), (h_3, f_3), (h_4, f_4)\}$ n'est pas stable car h_3 préfère f_2 à son partenaire et f_2 préfère h_3 à son partenaire. Dans ce cas (h_3, f_2) bloque M_1.
$M_2 = \{(h_1, f_3), (h_2, f_4), (h_3, f_2), (h_4, f_1)\}$ est un mariage stable car il n'existe aucun couple bloquant.

Préférences de $h_i \in H$					Préférences de $f_i \in F$				
$L(h_1) =$	f_2	f_3	f_1	f_4	$L(f_1) =$	h_2	h_1	h_3	h_4
$L(h_2) =$	f_4	f_1	f_3	f_2	$L(f_2) =$	h_3	h_4	h_1	h_2
$L(h_3) =$	f_2	f_4	f_1	f_3	$L(f_3) =$	h_1	h_3	h_4	h_2
$L(h_4) =$	f_3	f_1	f_4	f_2	$L(f_4) =$	h_2	h_1	h_3	h_4

TABLE 1 – Exemple

1. Proposez un modèle de programmation par contrainte pour trouver un mariage stable. Le modèle peut inclure des contraintes logiques (par exemple **if_then**). L'utilisation de ce genre de contraintes est accessible dans la documentation du solveur. Utilisez la recherche en profondeur par défaut (**('DepthFirst'**) sans préciser des heuristiques de branchements. N'affichez pas les traces d'exécutions internes du solveur.
2. Testez le modèle avec l'exemple ci-dessus (ou un autre exemple) en affichant toutes les solutions. Affichez chaque solution avec un format que vous jugez compréhensible.
3. Proposez **deux** stratégies de recherches différentes qui vous semblent très différentes (i.e., deux combinaisons de type <heuristique de choix de variables + heuristique de choix de valeurs>). On note ces deux stratégies par S_1 et S_2
4. On veut tester S_1 et S_2 avec une recherche en profondeur sans et avec redémarrage **('DepthFirst' vs 'Restart')**. Voici le protocole de l'étude expérimentale :
   — Générez différentes instances randomisées du problème avec différentes tailles (à partir de $n = 5$). Pour chaque taille, générez 5 instances.
   — Lancez les 4 expérimentations (S_1 et S_2 avec une recherche en profondeur sans et avec redémarrage) avec vos jeux de données. Fixez le temps limite à 200s pour chaque exécution [1].
   — Reportez le temps d'exécution et le nombre de nœuds dans des figures en fonction de la taille.
   — Quelles sont vos conclusions ?

Deuxième partie

On considère dans cette partie une version d'optimisation de ce problème. Étant donné un mariage M, on définit le degré de satisfaction de h_i dans M, noté par $d(M, h_i)$, comme le rang de son partenaire dans $L(h_i)$.

Dans l'exemple de la table 1, on a $d(M_2, h_4) = 2$ (car il est couplé avec f_1), $D(M_2, H) = 2$ (la pire satisfaction des agents dans H), $d(M_2, f_2) = 1$ (car f_2 est couplé à h_3) et $D(M_2, F) = 4$ (la pire satisfaction des agents dans H). Le problème de mariage stable équilibré est de trouver un mariage stable M tel que $|D(M, H) - D(M, F)|$ est minimale.

1. Proposez un modèle pour ce problème d'optimisation et testez le avec l'exemple précédant (ou autres exemples). N'affichez pas les traces d'exécutions internes du solveur.
2. En utilisant les instances générées dans la première partie, évaluez les deux stratégies S_1 et S_2 avec **uniquement** une recherche en profondeur sans redémarrage **(DepthFirst)**. Cette fois, on cherche la meilleure solution dans la limite de $200s$ par exécution.

   Présentez vos résultats sous forme de figures qui représentent le temps de résolution ainsi que la qualité des solutions (i.e., en se basant sur la meilleure valeur) en fonction de la taille de la donnée.
3. Quelles sont vos conclusions ?

1. On cherche à trouver une seule solution pour chaque exécution