UCLouvain

linfo2266

2026

Advanced Algorithms for Optimization

5.00 crédits	30.0 h + 15.0 h	Q1

Enseignants	Schaus Pierre ;
Langue d'enseignement	Anglais > Facilités pour suivre le cours en français
Lieu du cours	Louvain-la-Neuve
Préalables	Requis#: notions approfondies d'algorithmique et de programmation orientée objets telles que visées par le cours LEPL1402 Requis#: notions approfondies d'algorithmique et structures de données telles que visées par le cours LINFO1121
Thèmes abordés	• exploration d'arbres de recherche • branch and bound • relaxation (lagrangienne) et calcul de bornes • la recherche locale • la programmation mathématique • la programmation par contrainte • algorithmes de graphes, • les recherches à voisinage large • la programmation dynamique • les algorithmes gloutons et algorithmes approchés • l'optimisation multicritères • l'optimisation sans dérivée • comparaison d'algorithmes Ces méthodes seront appliquées sur des problèmes réels de type routing de véhicules, rostering et confection d' horaires, design de réseau, ordonnancement et scheduling, etc.
Acquis d'apprentissage	**A la fin de cette unité d'enseignement, l'étudiant est capable de :** Eu égard au référentiel AA du programme « Master ingénieur civil en informatique », ce cours contribue au développement, à l'acquisition et à l'évaluation des acquis d'apprentissage suivants : • AA1.1: Confronté à un problème informatique, il identifie les concepts, algorithmes et structures de données applicables pour le résoudre; et il en tire parti pour décomposer le problème en sous-problèmes et élaborer des méthodes de résolution informatique de ces derniers. • AA1.3: Confronté aux résultats obtenus par le raisonnement et la mise en oeuvre des outils et concepts qu'il a mobilisés, il prend le recul nécessaire pour en vérifier la pertinence, en ce qui concerne les fonctionnalités et la qualité de la solution recherchée. Dans ce contexte, il développera des tests et des vérifications pertinents qui peuvent garantir la qualité de la solution développée. • AA2.4: Dans la phase d'implémentation de la solution, il démontre sa maîtrise des principes, techniques et outils de développement à sa disposition. Il crée un prototype du logiciel à concevoir pour vérifier que le logiciel correspond bien aux attentes du client. Il crée un environnement et une batterie de tests pour s'assurer que la solution développée répond aux fonctionnalités du cahier des charges. En appliquant les techniques de validation et de vérification de programme, il identifie, localise les bugs (bogues) et y remédie. • AA5.4: Il sait se servir d'un ouvrage de référence ou d'un manuel relatif à l'emploi d'un langage informatique ou d'un logiciel, tant en anglais qu'en français. Il comprend un exposé technique fait en anglais. • AA5.5: Au cours du développement d'une application informatique, il en assure la traçabilité et la documentation dans un langage concis et précis : cahier des charges, structure du logiciel et des données qui y sont liées, mode opératoire. Il fait de même quand il s'agit de rédiger un rapport de synthèse décrivant et argumentant les choix (design et technologie) opérés dans le développement d'un projet. Eu égard au référentiel AA du programme « Master [120] en sciences informatiques », ce cours contribue au développement, à l'acquisition et à l'évaluation des acquis d'apprentissage suivants : • AA1: démontrer la maîtrise d'un solide corpus de connaissances en informatique, lui permettant de résoudre les problèmes qui relèvent de sa discipline • AA2.1: Analyser le problème à résoudre ou les besoins fonctionnels à rencontrer et formuler le cahier des charges correspondant. • AA2.2: Modéliser le problème et concevoir une ou plusieurs solutions techniques originales répondant à ce cahier des charges. • AA2.4: Implémenter et tester la solution retenue. • AA3.2: Proposer une modélisation et/ou un dispositif expérimental permettant de simuler et de tester des hypothèses relatives au problème étudié dans toute sa complexité. • AA6.5: S'autoévaluer et développer de manière autonome les connaissances nécessaires pour rester compétent dans son domaine.

	Les étudiants ayant suivi avec fruit ce cours seront capables de: • expliquer les algorithmes de résolution des problèmes d'optimisation discrets en les décrivant précisément, en précisant les problèmes qu'ils permettent de résoudre, en indiquant leurs avantages, inconvénients et limites (temps de calcul, exactitude, problèmes de passage à l'échelle, etc); • identifier les algorithmes qui s'appliquent à un problème d'optimisation discret auquel on est confronté et faire un choix argumenté parmi ceux-ci; • implémenter les algorithmes permettant de résoudre des problèmes d'optimisation discrets; • évaluer et comparer les différents algorihtmes d'optimisation.

Modes d'évaluation des acquis des étudiants	**Janvier:**
	Pour la première session, la note globale du cours est pondérée comme suit: 45% les projets et rapports, 45% l'examen écrit et 10% la participation au cours.
	Août:
	Pour la deuxième session, les ne pourront pas être resoumis. Seul l'examen sur 45% pourra être refait et celui-ci sera soit un oral, soit un examen écrit.
	Bonus
	Un concours d'optimisation sera proposé durant le quadrimestre. Selon le classement de l'étudiant, celui-ci pourra obtenir 0, 1, 2, 3, 4 ou 5 points. Un étudiant qui obtient X points aura donc la note finale suivante : (score_sur_20 / 20 * (20 - X) + X). Une note de zéro au concours n'entraîne donc aucune perte de points sur la note finale, mais une note de 5 peut vous permettre d'améliorer votre score. En effet, en supposant que vous obteniez 12/20 aux projets + examen + participation, et 5 points au concours, vous atteindrez la note finale de 12/20 * 15 + 5 = 14/20.
	Utilisation de l'IA générative dans les devoirs du cours (# Source : Rédigé avec l'aide de ChatGPT et des Prof. Schaus & Dupont pour la formulation)
	Les projets sont individuels (aucun échange de code n'est toléré) . Néanmoins, dans ce cours, nous reconnaissons l'évolution de la technologie et les avantages potentiels de l'utilisation des outils d'IA générative dans le processus de programmation. Cependant, l'honnêteté académique et l'originalité restent primordiales. À cet effet :
	• **Utilisation de l'IA générative** : Les étudiants sont autorisés à utiliser les outils d'IA générative pour les aider dans leurs devoirs. Ces outils peuvent fournir de l'inspiration, suggérer des approches de codage ou aider à résoudre des problèmes.
	• **Travail original** : Bien que l'IA puisse être un outil, elle ne doit pas être le seul auteur de votre devoir. Votre soumission doit être principalement votre propre travail. Copier et coller directement des solutions issues des résultats de l'IA sans compréhension ni modification est déconseillé. De même, collaborer avec d'autres étudiants est une partie précieuse du processus d'apprentissage, mais copier directement le travail d'un autre étudiant est considéré comme du plagiat.
	• **Indication de la source** : Chaque fois que vous utilisez l'IA générative pour aider dans votre devoir, vous devez l'indiquer en fournissant un bref commentaire dans votre code sur la manière dont l'IA a été utilisée. Par exemple:
	`# IA utilisée pour suggérer une optimisation pour cette boucle.`
	`for i in range(10): ...`
	Le non-respect de ces directives peut entraîner une réduction des notes ou d'autres sanctions académiques.
	Les mêmes conséquences s'appliqueront à un étudiant qui partage volontairement son code ou le rend disponible à d'autres étudiants (y compris en partageant votre code sur un dépôt public ou privé). Si le professeur le juge nécessaire, un entretien sur les projets pourra également être organisé.

Méthodes d'enseignement	La présentation des algorithmes sera soit proposée sous forme de cours magistraux, de vidéos ou de lecture et sera accompagnée de travaux pratiques (devoirs/micro-projets) sollicitant la mise en œuvre d'algorithmes pour résoudre un problème pratique d'optimisation et la rédaction de rapports.
Contenu	Ce cours présente une série de techniques pour des solutions exactes et approximatives aux problèmes d'optimisation combinatoire. Parmi les techniques : • programmation dynamique • branch and bound • programmation linéaire • relaxation Lagrangienne • génération de colonnes • recherche locale • programmation par contrainte • algo de graphes: problème de max flow • A* et ses variantes pour l'optimisation • comparaison d'algoritthmes d'optimisation Utilisation de ces méthodes sur des problèmes réels: tournées de véhicules, ordonnancement, confection d'horaire, design de réseau, etc.

Ressources en ligne	https://moodle.uclouvain.be/course/view.php?id=1474
Autres infos	Préalables: un background solide en algorithmique et structure de données (par exemple acquis via le cours LINFO1121) et un bonne maitrise du langage Java
Faculté ou entité en charge:	INFO

Programmes / formations proposant cette unité d'enseignement (UE)

Intitulé du programme	Sigle	Crédits	Prérequis	Acquis d'apprentissage
Master [120] : ingénieur civil en informatique	INFO2M	5		
Master [120] en sciences informatiques	SINF2M	5		
Master [120] : ingénieur civil en science des données	DATE2M	5		
Master [120] en science des données, orientation technologies de l'information	DATI2M	5		