

Recherche

M1 IDD 2019–2020 *Représentation des connaissances et raisonnement*

Stéphane Airiau



Pourquoi avoir un cours d'IA

- Qu'est-ce que l'IA ?
question difficile, les contours de l'IA sont parfois assez flous
- Qu'est ce que l'IA fait pour vous ?
 - traduction automatique (par exemple Mandarin → Français)
par exemple, essayez `deepl.com`.
 - identification d'objet dans des images (ex : chaises, visages, etc)
essayez *Rasta* (stage réalisé au lamsade)
 - plier votre linge
 - démontrer ou aider à démontrer de nouveaux théorèmes
 - assistants (médical, légal, ex IBM Watson)
 - robots (aides aux personnes âgées, aides musées)
 - **voitures autonomes**
 - **jouer au jeu de Go**
 - **jouer au pocker** Brains Vs. AI

Terme créé par John McCarthy (aussi développeur du langage LISP) autour de 1956

Construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisantes par des êtres humains car elles demandent des processus mentaux de haut niveau tels que l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique.

Marvin Minsky

Un enjeu politique

- France IA
- The Administration's Report on the Future of Artificial Intelligence

Le test de Turing

- Turing (1950) “Computing machinery and intelligence”
- Le test est créé pour donner une définition opérationnelle satisfaisante de l’intelligence
- Un ordinateur passe ce test si un homme, après avoir posé des questions écrites ne sait pas s’il s’adresse à un autre humain ou à une machine.

International Joint Conference on Artificial Intelligence

Agent and Multiagent Systems

Constraint Optimization

Ontologies

Game Theory

Heuristic Search

Graphical Models

Robotics and Vision

Natural Language Processing

Planning

Relational Learning

Satisfiability

Social Choice Theory

Vision and Perception

Web Mining

Artificial Intelligence and Social Sciences

Auctions and Market-Based Systems

Distributed Search/CSP/Optimization

Knowledge Representation, Reasoning, and Logic

Knowledge Acquisition

Machine Learning

Multidisciplinary Topics and Applications

Constraints, Satisfiability, and Search

Recommender Systems

Model Verification / Model Checking

Sequential Decision Making

Social Networks

Web and Knowledge-Based Information Systems

Special Track on Artificial Intelligence and the Arts

Special Track on Computational Sustainability

Special Track on Knowledge Representation and Reasoning

Special Track on Machine Learning

Economie ?

Agent and Multiagent Systems

Constraint Optimization

Ontologies

Game Theory

Heuristic Search

Graphical Models

Robotics and Vision

Natural Language Processing

Planning

Relational Learning

Satisfiability

Social Choice Theory

Vision and Perception

Web Mining

Artificial Intelligence and **Social Sciences**

Auctions and Market-Based Systems

Distributed Search/CSP/Optimization

Knowledge Representation, Reasoning, and Logic

Knowledge Acquisition

Machine Learning

Multidisciplinary Topics and Applications

Constraints, Satisfiability, and Search

Recommender Systems

Model Verification / Model Checking

Sequential Decision Making

Social Networks

Web and Knowledge-Based Information Systems

Special Track on Artificial Intelligence and the Arts

Special Track on Computational Sustainability

Special Track on Knowledge Representation and Reasoning

Special Track on Machine Learning

Logique ?

Agent and Multiagent Systems

Constraint Optimization

Ontologies

Game Theory

Heuristic Search

Graphical Models

Robotics and Vision

Natural Language Processing

Planning

Relational Learning

Satisfiability

Social Choice Theory

Vision and Perception

Web Mining

Artificial Intelligence and Social Sciences

Auctions and Market-Based Systems

Distributed Search/CSP/Optimization

Knowledge Representation, Reasoning, and Logic

Knowledge Acquisition

Machine Learning

Multidisciplinary Topics and Applications

Constraints, Satisfiability, and Search

Recommender Systems

Model Verification / Model Checking

Sequential Decision Making

Social Networks

Web and Knowledge-Based Information Systems

Special Track on Artificial Intelligence and the Arts

Special Track on Computational Sustainability

Special Track on Knowledge Representation and Reasoning

Special Track on Machine Learning

Optimisation Combinatoire ?

Agent and Multiagent Systems

Constraint Optimization

Ontologies

Game Theory

Heuristic Search

Graphical Models

Robotics and Vision

Natural Language Processing

Planning

Relational Learning

Satisfiability

Social Choice Theory

Vision and Perception

Web Mining

Artificial Intelligence and Social Sciences

Auctions and Market-Based Systems

Distributed Search/**CSP/Optimization**

Knowledge Representation, Reasoning, and Logic

Knowledge Acquisition

Machine Learning

Multidisciplinary Topics and Applications

Constraints, Satisfiability, and Search

Recommender Systems

Model Verification / Model Checking

Sequential Decision Making

Social Networks

Web and Knowledge-Based Information Systems

Special Track on Artificial Intelligence and the Arts

Special Track on Computational Sustainability

Special Track on Knowledge Representation and Reasoning

Special Track on Machine Learning

Agent and Multiagent Systems

Constraint Optimization

Ontologies

Game Theory

Heuristic Search

Graphical Models

Robotics and Vision

Natural Language Processing

Planning

Relational Learning

Satisfiability

Social Choice Theory

Vision and Perception

Web Mining

Artificial Intelligence and Social Sciences

Auctions and Market-Based Systems

Distributed Search/CSP/Optimization

Knowledge Representation, Reasoning, and Logic

Knowledge Acquisition

Machine Learning

Multidisciplinary Topics and Applications

Constraints, Satisfiability, and Search

Recommender Systems

Model Verification / Model Checking

Sequential Decision Making

Social Networks

Web and Knowledge-Based Information Systems

Special Track on Artificial Intelligence and the Arts

Special Track on Computational Sustainability

Special Track on Knowledge Representation and Reasoning

Special Track on Machine Learning

- **Machine Learning** (Classification, Feature Selection, Data Mining, Learning Graphical Model, Active Learning, Relational learning, Time series and Data Stream, Classification, Kernel Methods, Deep Learning, Data Mining and Personalisation, Semi supervised Learning, Reinforcement learning)
- **Constraint Satisfaction** (Constraint Satisfaction 2, Solvers and tools,)
- **Uncertainty in AI** (Approximate Probabilistic inference 2, MDP)
- **Robotique** (Vision and Perception, motion and path planning, robotics and vision)
- **Search** (Search in planning and scheduling, Heuristic search, planning algorithms, Satisfiability, Activity and plan recognition)
- **Systemes multiagents** (Agent theories, Cooperative Games, Noncooperative games, Economic Paradigms, Agent based simulations, etc)
- **Traitement automatique des langues** (Natural Language semantics, Natural Language processing, applications and tools, Discourse, Sentiment analysis and Text mining, Machine Translation)
- **Knowledge representation** (Common sense reasoning, automated reasoning and theorem proving, Description Logics and Ontologies, Geometric spatial and temporal reasoning, Computational Complexity of Reasoning, Belief change)

Plan du cours (pas complètement final)

- Algorithmes de recherche
 - Algorithmes de recherche et heuristiques
 - Recherche locale
(algorithmes génétiques, hill climbing, ...)
 - Problème de satisfaction de contraintes
 - Jeux à deux joueurs
minimax et élagage α - β
Monte Carlo Tree Search
- } Good old fashion AI (GOFAI)
- Apprendre un arbre de décision
 - Réseaux de neurones
 - Intelligence Artificielle et Ethique ?
- } Apprentissage automatique

- un examen (semaine du 18 mai) : 50%
- un projet : 30% (travail effectué, rapport, soutenance)
- CC : 20% : implémentation des algorithmes des TDs et mini rapport.

But du cours

- présenter un panorama de problèmes et d'algorithmes pour résoudre ces problèmes
- *comprendre* le fonctionnement des algorithmes
- les séances de TPs TDs vous proposerons d'implémenter la plupart des algorithmes et de les tester.
- en Java ➡ pour ceux qui ne connaissent pas le langage, trouver un binôme qui peut vous aider !

Ce cours demande un travail d'implémentation pour bien comprendre les algorithmes !

Résolution d'un problème grâce à la recherche dans un graphe

M1 IDD 2019–2020 *Représentation des connaissances et raisonnement*

Stéphane Airiau



Exemple d'un jeu à résoudre : le taquin



taquin 2×2

Un taquin $n \times n$ est constitué de $n^2 - 1$ jetons carrés à l'intérieur d'un carré pouvant contenir n^2 jetons : on a donc une **case vide**.

Les coups permis sont ceux qui font glisser dans la case vide un des 2, 3 ou 4 jetons contigus.

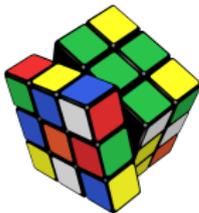
But : partir de l'état du taquin ci-dessus pour arriver à l'état du taquin ci-dessous.



Un programme peut-il résoudre ce type de problèmes pour $n=2,3,4,5... ?$

Jeux dans le même esprit

- poser n reines sur un échiquier de taille $n \times n$ sans qu'aucune reine n'attaque une autre reine.
- résoudre un "rubik's cube"



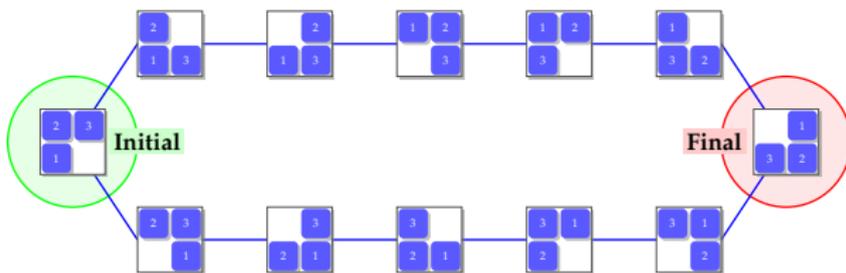
- résoudre un niveau de "Sokoban"
- jouer "aux chiffres" de l'émission des chiffres et des lettres.
- trouver un itinéraire d'un point A à un point B à l'aide d'une carte
- trouver un itinéraire qui passe exactement une fois par chaque ville

On peut aussi se poser des questions **d'optimisation** :

- résoudre le taquin en un minimum de coups
- trouver un itinéraire qui minimise le temps de parcours
- ...

Définition du problème : la modélisation

- 1^{ère} étape : décider ce que sont les états et les actions.
 - 2^{nde} étape : bien définir le problème avec
 - un état de **départ**.
 - les **actions** possibles pour chaque état
 - le modèle de **transition** : c'est une fonction qui donne l'état qui résulte d'avoir effectué une action depuis un état.
- ➔ on peut représenter l'état de départ, les actions et le modèle de transition par un **graphe**.
- une fonction qui teste si le but est **atteint**.
 - éventuellement une fonction de coût pour chaque action dans chaque état.



Graphe pour le taquin 2 × 2

Définition du problème : la modélisation

- un état de **départ**.
- les **actions** possibles pour chaque état
- le modèle de **transition**
- une fonction qui teste si le but est **atteint**.
- éventuellement une fonction de coût pour chaque action dans chaque état.

Une fois que le problème est modélisé

⇒ on peut utiliser des algorithmes génériques / solveurs indépendants

⇒ force brute de l'ordinateur

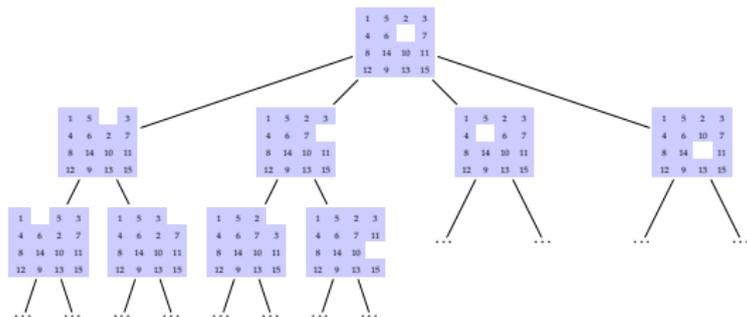
mais une certaine intelligence : avoir une méthode/ algorithme qui résout plein de problèmes

(quelques efforts faits par un humain pour bien modéliser le problème)

Résolution

La séquence d'actions possibles à partir de l'état initial forme un **graphe de recherche** :

- les noeuds représentent les états
- les branches représentent les différentes actions
- la racine est l'état initial

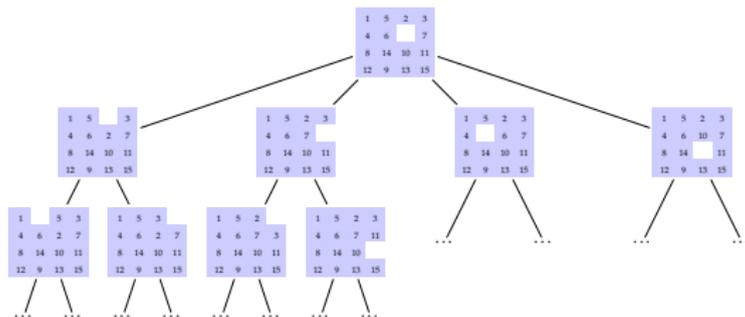


- Pour résoudre : on choisit un noeud à développer et on teste si les fils sont des états finaux.
- Quelle politique pour choisir le noeud à développer ?
- Attention : on peut répéter des noeuds → on peut tourner en boucle (sauf si on se rappelle des états visités) !

Résolution

La séquence d'actions possibles à partir de l'état initial forme un **graphe de recherche** :

- les noeuds représentent les états
- les branches représentent les différentes actions
- la racine est l'état initial



Attention : pour expliquer un algorithme, on dessine/parcourt un graphe **mais** lors de l'exécution de l'algorithme, tout le graphe n'est pas forcément stocké en mémoire.

Taquin $n \times n$

- résoudre un problème de taquin 2×2 à l'air facile !
- Pensons au taquin 4×4
 - Combien y a-t-il de sommets exactement ?
 - ?
 - la réponse est un entier proche de 2.10^{13}
 - Certains problèmes sont impossibles (il n'existe pas de chemin de l'état initial à l'état final).
 - en fait pour le 4×4 , le graphe possède deux composantes connexes
 - si on tire au sort l'état initial et l'état final on a 50% de chance qu'une solution existe.
 - il existe un test simple pour le savoir.
- Peut-on utiliser des algorithmes de la théorie des graphes ?
- suppose que le graphe puisse être stocké en mémoire peut être possible pour 4×4 , pas sûr pour 5×5 ou 6×6 !
- on peut utiliser des algorithmes enseignés en cours d'algorithmique.

Résoudre avec des algorithmes de base

- expansion d'un graphe en largeur d'abord ("**breadth-first search**" (BFS))
- expansion d'un graphe en profondeur d'abord ("**depth-first search**" (DFS))
- expansion d'un graphe avec coût uniforme ("**Uniform-cost search**") : développer le noeud qui a le coût le plus bas.
- recherche en profondeur limitée ("**depth-limited search**") : utilise DFS jusqu'à une profondeur l donnée
cela évite le problème du chemin infini, mais pose problème si la solution est plus profonde que l .
- recherche en profondeur itérative ("**iterative deepening DFS**") : combine DFS et BFS : itérativement utilise DFS jusqu'à une profondeur de $1, 2, \dots$ jusqu'à trouver le but.
les états sont générés de nombreuses fois
- recherche bidirectionnelle ("**Bidirectional search**") : effectue deux recherches : une de l'état initial vers l'état final, l'autre dans le sens inverse (donc depuis l'état final).

Répétition des états

On ajoutera à ces stratégies la faculté de se souvenir ou non des noeuds visités

- sans mémoire : **“tree-search”** risque que les états se répètent
⇒ risque de boucle
- avec mémoire : **“graph-search”** nécessite des ressources de stockage évite les boucles !
- ⇒ on ajoute une liste d'états visités (parfois appelée “explored set” ou “closed list”)

Critères pour comparer ces algorithmes

- **Complétude** : a-t-on une garantie de trouver une solution quand elle existe ?
- **Complexité du temps de calcul** : combien de temps a-t-on besoin (dans le pire des cas)
- **Complexité de l'espace mémoire** : combien d'espace mémoire a-t-on besoin (dans le pire des cas)
- **Optimalité** : est-ce que la solution trouvée est optimale ?

Comparaison

	BFS	Uniform-Cost	DFS	Depth-limited	Iterative-deepening	Bidirectional
complet ?	✓?✗	✓?✗	✓?✗	✓?✗	✓?✗	✓?✗
complexité temps	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$
complexité mémoire	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$	$\mathcal{O}(?)$
optimal ?	✓?✗	✓?✗	✓?✗	✓?✗	✓?✗	✓?✗

- b sera le facteur de branchement de l'arbre
- d est la profondeur de la solution la moins profonde
- m est la profondeur maximale de l'arbre de recherche
- l est la limite de profondeur