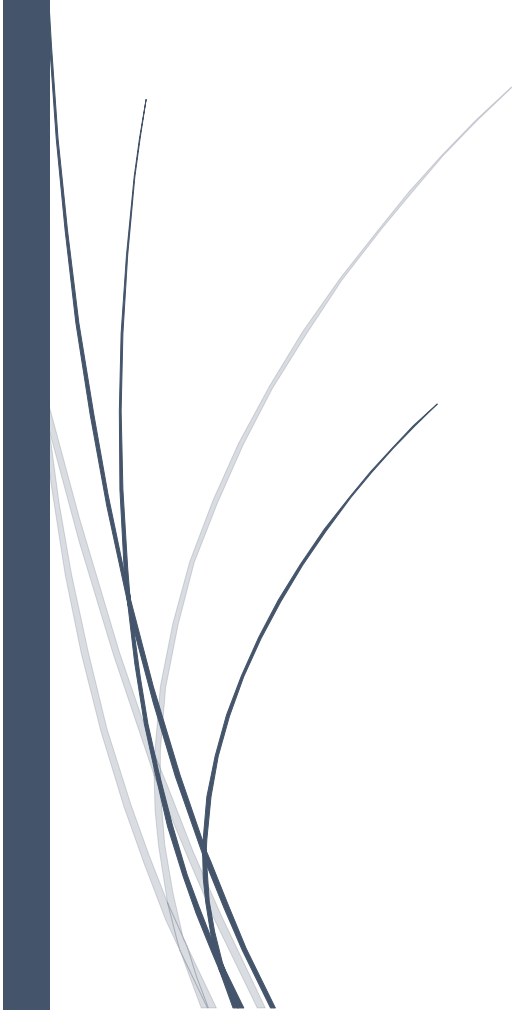
23/04/2021

DEVOIR

Cours d'algorithme et ordonnancement



Cédric Nguendap
UDM

Table des matières

Liste des figures

I. INTRODUTION

1. Complexité des algorithmes et classification des problèmes

a. Complexité

C'est la *mesure du nombre d'opérations fondamentales qu'un algorithme effectue sur un jeu de données*. On cherche à déterminer une mesure qui rende compte de la complexité intrinsèque (propre et essentiel) des algorithmes indépendamment de leur implémentation permettant ainsi de comparer entre eux plusieurs algorithmes. En général la complexité est évaluée sur deux (2) axes :

- La complexité en espace : il évalue l'espace mémoire utilisé par l'algorithme lors de son exécution
- La complexité en temps : il évalue le nombre d'opérations effectué par l'algorithme lors de son exécution

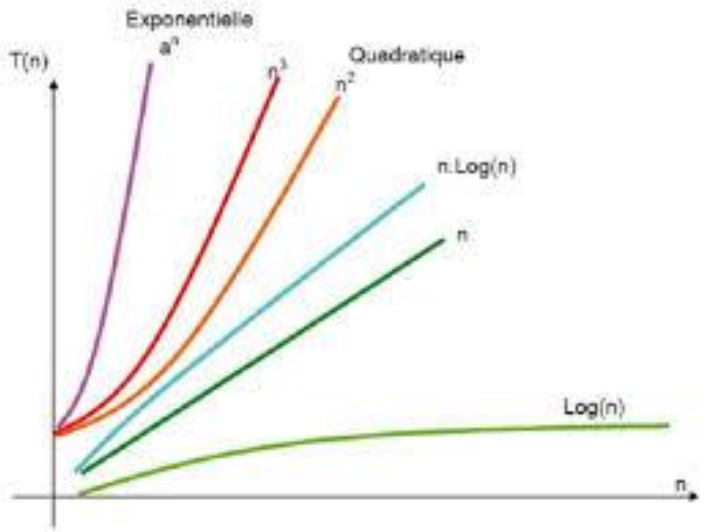


Figure 1: complexité

La figure 1 présente l'évolution du temps d'exécution d'un algorithme en fonction de la taille de son entré

b. Classification des problèmes

Afin de résoudre plus facilement les problèmes, les classes de problèmes permettent d'appliquer des méthodes de résolution bien connus. Dans cette classification, nous avons :

- Les problèmes de décisions : Ceux sont les problèmes dont le résultat est booléen (oui ou non, vrai ou faux, 0 ou 1)
- Les problèmes d'optimisation : Ceux sont les problèmes de minimisation (ou de maximisation). Ici il est question de minimiser ou d'optimiser une fonction objective
- Les problèmes de recherche : Ceux sont les problèmes de recherche d'élément.

Il existe une autre classification de problèmes basé sur leurs complexités. Nous avons principalement :

- La classe P se compose des problèmes qui sont résolubles en temps polynomial. Plus précisément, il existe des problèmes qui peuvent être résolus en temps $O(n^k)$ pour une certaine constante k (n est la taille de l'entrée).
- La classe NP se compose des problèmes qui sont « vérifiables » en temps polynomial. Cela signifie : si l'on nous donne, d'une façon ou d'une autre, une solution « certifiée », nous pouvons vérifier que cette solution est correcte en un temps qui est polynomial par rapport à la taille de l'entrée.

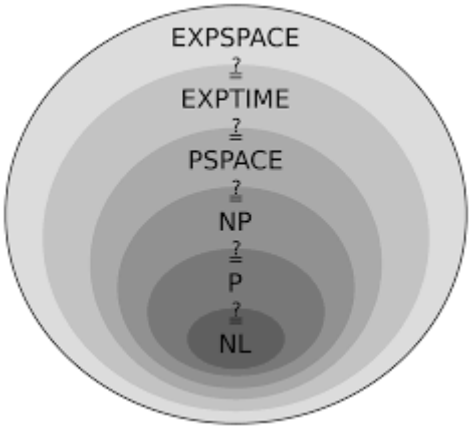


Figure 2: Les différentes classes de problèmes

La figure 2 présente les différentes classes de complexité d'un algorithme

2. Preuve de la NP-Complétude

Figure 3: La np-completude

Certains algorithme sont des ***algorithmes à temps polynomial*** : sur des entrées de taille n, leur temps d'exécution dans le pire des cas est : $O(n^k)$ pour une certaine constante k. Il est naturel de se demander si *tous* les problèmes peuvent être résolus en temps polynomial. La réponse est non. Par exemple, il existe des problèmes, tel le fameux « problème de l'arrêt de Turing d'une machine », qui ne peuvent être résolus par aucun ordinateur, quel que soit le temps qu'il y passe. De manière générale, on considère que les problèmes résolubles par des algorithmes à temps polynomial sont traitables (faciles) et que les problèmes nécessitant un temps supra polynomial sont intraitables (difficiles).

La méthode de démonstration de la np-complétude dépend de la classe de probléme.

- Problèmes de décision et problèmes d'optimisation : Nombre de problèmes intéressants sont des ***problèmes d'optimisation***, dans lesquels chaque solution réalisable (c'est à dire, «légale »)a une valeur associée et où l'on veut trouver la solution réalisable qui a la valeur optimale. Ainsi, dans un problème que nous appelons PLUS-COURT-CHEMIN, on a un graphe non orienté *G* et des sommets *u* et *v*, et l'on veut trouver le chemin de *u* à *v* qui utilise le moins d'arêtes. (En d'autres termes, PLUS-COURT-CHEMIN est le problème du plus court chemin à origine unique dans un graphe non orienté non pondéré.) La NP-complétude ne s'applique pas directement, toutefois, aux problèmes d'optimisation, mais aux ***problèmes de décision*** dans lesquels la réponse est tout simplement « oui » ou « non » (ou, plus formellement, « 1 » ou « 0 »)
- Réductions

  Cette notion de montrer qu'un problème n'est pas plus difficile ou plus facile qu'un autre s'applique même quand les deux problèmes sont des problèmes de décision. Cette idée est exploitée dans la quasi-totalité des démonstrations de NP-complétude, et ce de la façon suivante. Soit un problème de décision, disons *A*, que l'on voudrait résoudre en temps polynomial. L'entrée d'un problème particulier est dite ***instance*** de ce problème ; ainsi, dans CHEMIN, une instance serait un graphe particulier *G*, des sommets particuliers *u* et *v* de *G*, et un entier particulier *k*. Supposons maintenant qu'il y ait un autre problème de décision, disons *B*, que nous savons déjà comment résoudre en temps polynomial. Enfin, supposons que nous ayons une procédure qui transforme toute instance a de *A* en une certaine instance b de *B* et qui a les caractéristiques suivantes:
- 1)La transformation prend un temps polynomial.
- 2) Les réponses sont les mêmes. C'est à dire, la réponse pour a est « oui » si et seulement si la réponse pour b est « oui ». Une telle procédure porte le nom ***de réduction*** à temps polynomial et, elle donne un moyen de résoudre le problème *A* en temps polynomial : 1) Étant donnée une instance a de *A*, utiliser une réduction à temps polynomial pour la transformer en une instance b de *B*.
- 2) Exécuter l'algorithme de décision à temps polynomial de *B* sur l'instance b. 3) Utiliser la réponse pour b comme réponse pour a

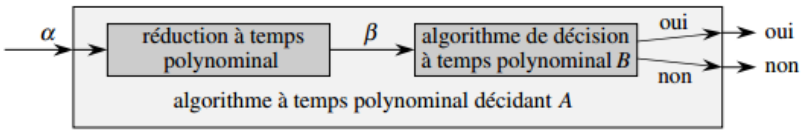


Figure 4 Principe de la réduction polynomiale

3. Méthode exacte et méthode approché

De nombreux problèmes d'intérêt pratique, bien que NP-difficiles, sont trop importants pour qu'on les abandonne pour la seule raison qu'il n'existe pas d'algorithme efficace pour obtenir une solution optimale. Si un problème est NP-difficile, il y a peu de chances de trouver un algorithme à temps polynomial capable de le résoudre exactement, mais cela n'implique pas que tout espoir soit perdu. Il existe au moins deux approches pour contourner la NP-difficulté. Primo, si les entrées réelles sont petites, un algorithme à temps d'exécution exponentiel peut parfaitement convenir. Secundo, il se peut que l'on puisse trouver des solutions *quasi optimales* à temps polynomial (dans le cas le plus défavorable, ou en moyenne). En pratique, on peut souvent se contenter d'une solution quasi optimale. Un algorithme qui retourne des solutions quasi optimales est appelé ***algorithme d'approximation***.

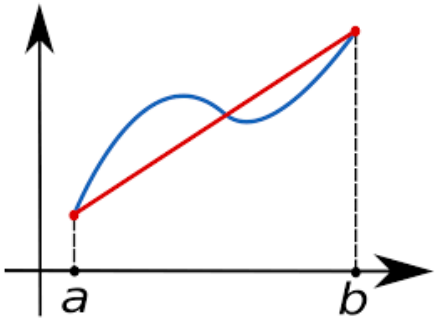


Figure 5 Méthode exacte VS méthode approché

La figure 5 présente la principale différence entre méthode exacte (en rouge) et méthode approché (en bleu)

2. Définition et formalisation des problèmes d'ordonnancement

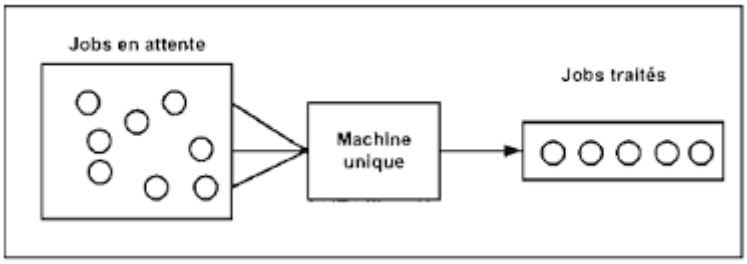


Figure 6: théorie de l'ordonnancement

a. Définition des problèmes d'ordonnancement

Ordonnancer un ensemble de *tâches*, c'est programmer leur exécution en leur allouant les *ressources* requises et en fixant leurs dates de début. La théorie de l'ordonnancement traite de modèles mathématiques mais analyse également des situations réelles fort complexes.

Dans un problème d'ordonnancement interviennent deux notions fondamentales : les tâches et les ressources. Une ressource est un moyen, technique ou humain, dont la disponibilité limitée ou non est connue *a priori.* Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource.

La résolution d'un problème d'ordonnancement doit concilier deux objectifs. L'aspect *statique* consiste à générer un plan de réalisation des travaux sur la base de données prévisionnelles. L'aspect *dynamique* consiste à prendre des décisions en temps réel compte tenu de l'état des ressources et de l'avancement dans le temps des différentes tâches.

b. Formalisation des problèmes d'ordonnancement

Dans un problème d'ordonnancement interviennent deux notions fondamentales : les tâches et les ressources. Une ressource est un moyen, technique ou humain, dont la disponibilité limitée ou non est connue a priori. Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource.

La résolution d'un problème d'ordonnancement doit concilier deux objectifs. L'aspect statique consiste à générer un plan de réalisation des travaux sur la base de données prévisionnelles. L'aspect dynamique consiste à prendre des décisions en temps réel compte tenu de l'état des ressources et de l'avancement dans le temps des différentes tâches.

Les données d'un problème d'ordonnancement sont les tâches et leurs Caractéristiques, les contraintes potentielles, les ressources, et la fonction économique.

On note en général :

- I = {1, 2, . . . , n) l'ensemble des tâches
- Pi : la durée de la tâche i
- r_i : Date de disponibilité
- d_i : date échue
- q_i : latence
- t_i : dates réelles de début
- C_i : dates réelles de fin

Les tâches sont souvent liées entre elles par des relations d'antériorité. Si ce n'est pas le cas on dit qu'elles sont indépendantes. La contrainte d'antériorité la plus générale entre deux tâches i et j, appelée contrainte potentielle, s'écrit sous la forme $t_j - t_i > a_y$, elle permet d'exprimer la succession simple ($a_{ij} = p_i$) et de nombreuses variantes.

Dans certains cas, une tâche, dite morcelable ou encore préemptive, peut être exécutée par morceaux. Des techniques d'optimisation issues des mathématiques du continu (par exemple la programmation linéaire) sont alors souvent utiles pour résoudre ce type de problèmes.

Dans certains cas, une tâche, dite morcelable ou encore préemptive, peut être exécutée par morceaux. Des techniques d'optimisation issues des mathématiques du continu (par exemple la programmation linéaire) sont alors souvent utiles pour résoudre ce type de problèmes.

On distingue deux types de ressources pouvant être requises par les tâches, les ressources renouvelables et les ressources consommables

3. Modélisation par le réseau de Pétri

a. Définition

Un **réseau de Petri**[1] (aussi connu comme un réseau de **Place/Transition** ou réseau de **P/T**) est un modèle mathématique servant à représenter divers systèmes (informatiques, industriels…) travaillant sur des variables discrètes. Les réseaux de Pétri sont des outils graphiques et mathématiques permettant de modéliser et de vérifier le comportement dynamique des systèmes à événements discrets comme les systèmes manufacturiers, les systèmes de télécommunications, les réseaux de transport.

b. Modélisation

Un réseau de Petri est un 6-uplet (S,T,F,M_0,W,K), où :

- S : définit une ou plusieurs *places*.
- T : définit une ou plusieurs *transitions*.
- F : définit un ou plusieurs *arcs* (flèches).

Un arc ne peut pas connecter deux places ni deux transitions, il ne peut connecter que des paires place-transition ; plus formellement : $F \subseteq (S \times T) \cup (T \times S)$

- M_0: S → N : appelé *marquage initial*, où, chaque place $\delta \in S$, il y a N jetons.
- W : F→ N^+appelé ensemble d'*arcs primaires* , assignant à chaque arc $f \in F$ un entier positif $n \in N^+$qui indique combien de jetons sont *consommés* depuis une place vers une transition, ou sinon, combien de jetons sont produits par une transition et arrivent pour chaque place.
- K : S→ N^+appelé *limite de capacité*, faisant correspondre à chaque place $\delta \in S$ un nombre positif $n \in N^+$ représentant le nombre maximum de jetons qui peuvent occuper une place.

De nombreuses définitions formelles existent. Cette définition concerne un réseau *place-transition* (ou *P-T*). D'autres définitions n'incluent pas la notion d'*arc primaire* ou la *limite de capacité*.

Un réseau de Petri se représente par un graphe biparti (composé de deux types de nœuds et dont aucun arc ne relie deux nœuds de même type) orienté (composé d'arc(s) ayant un sens) reliant des *places* et des *transitions* (les nœuds). Deux places ne peuvent pas être reliées entre elles, ni deux transitions. Les places peuvent contenir des *jetons*, représentant généralement des ressources disponibles. La distribution des jetons dans les places est appelée le *marquage* du réseau de Pétri.

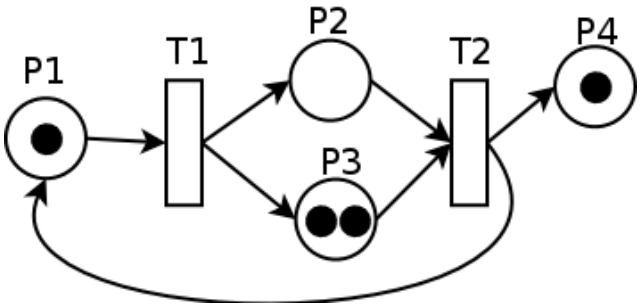


Figure 7: modélisation par réseau de Pétri