Chapitre 2

Arbre de plus court chemin

Dans ce chapitre, nous nous intéressons au problème de la construction d'un arbre de plus court chemin. Nous nous focalisons sur une variante naturelle qui consiste à construire des arbres de plus court chemin lorsque chacun des noeuds possède un intérêt propre.

Nous motivons le problème dans la première section. Dans la deuxième section, nous définissons formellement le problème de la construction d'un arbre stable de plus court chemin. Dans la troisième section, nous présentons quelques résultats connus sur le problème. Dans la quatrième section, nous présentons une variante de ce problème où nous considérons que les noeuds peuvent appartenir à des coalitions (couleurs). Dans la cinquième section, nous présentons quelques liens avec la théorie des jeux.

Ce chapitre vise à présenter d'une part quelques résultats personnels, essentiellement dans la quatrième section [10], [42]. Mes résultats personnels relatifs à ce chapitre consistent principalement en d'une part une preuve de la complexité du problème de plus court chemin avec couleurs, d'autre part en des conditions nécessaires pour l'existence d'un arbre stable et enfin un algorithme auto-stabilisant de construction d'arbres stables.

D'autre part, et surtout, ce chapitre vise à présenter par cet exemple introductif le type de problèmes et le type de questions algorithmiques que nous nous posons. En particulier, il nous sert à introduire quelques concepts de la théorie algorithmique des jeux sur lesquels nous reviendrons.

Les travaux personnels de ce chapitre ont partiellement été obtenus en collaboration avec A. Dasgupta, S. Delaët, S. Ghosh, et S. Tixeuil.

2.1 Motivation

Nous nous intéressons au problème classique de la construction d'arbre de plus court chemin. Ce problème est clairement relié au problème du routage dans les réseaux.

Vu les évolutions des réseaux (routage inter-domaine, système de peering, etc ...), l'algorithmique répartie doit maintenant prendre en compte le fait que les nœuds du réseau ont leur intérêt propre même s'ils participent à un processus global. La théorie des jeux est un outil naturel pour modéliser ces phénomènes. Comme nous allons le voir, elle n'apporte pas de solution algorithmique directe pour résoudre la concurrence entre les participants, mais plutôt un outil naturel de modélisation.

Voici une définition plus formelle du problème de l'arbre de plus court chemin :

`Problème :` `Arbre du plus court chemin`
`Instance :` un graphe $G = (V, E)$ où V est l'ensemble des sommets et E est l'ensemble des arêtes, une fonction de poids w sur les arêtes, et un sommet distingué r dans V
`Objectif :` Construire un arbre de plus court chemin enraciné en r.

Le problème de la construction d'arbre de plus court chemin enraciné en un sommet r dans un graphe $G = (V, E)$ a été étudié de façon intensive dans le cadre centralisé où l'algorithme a une vue complète du graphe. Historiquement, deux principaux algorithmes sont connus : celui de Bellman et Ford [Cormen et al., 2001] et celui de Dijkstra [Dijkstra, 1971]. Par la suite, des variantes réparties (chaque sommet possède une vue locale du graphe : son voisinage et son propre état) de ces algorithmes ont été réalisées, puis adaptées aux réseaux sous forme de protocoles de routage.

Les protocoles de routage construisent les chemins entre deux endroits du réseau pour faire transiter les données. Par exemple, le protocole *Routing Information Protocol* (RIP) correspond à une adaptation de l'algorithme de Bellman et Ford, et le protocole *Open Shortest Path First* ($OSPF$) à celui de Dijkstra : voir [Huitema, 2001]. En effet les protocoles de routage dynamiques des réseaux locaux (une seule entité gérante) tel que RIP et $OSPF$ sont essentiellement basés sur des algorithmes distribués pour la résolution du problème du plus court chemin.

Dans le cadre du réseau Internet, ce n'est plus le cas. En effet, Internet est composé de systèmes autonomes (*Autonomous System*) qui s'échangent des messages afin de pouvoir assurer le routage. Chaque système autonome correspond à un ensemble de réseaux et de routeurs sous une administration unique et possède sa propre politique de routage correspondant aux préférences d'administration de cette partie du réseau.

Actuellement, un routage adapté au fait que les nœud du réseau aient leur propre politique de routage est le protocole *Border gateway protocol BGP* (utilisé pour le routage inter-domaine dans l'Internet). Il permet à chacun des systèmes autonomes d'utiliser ses propres politiques de routage pour déterminer ses propres routes. Par conséquent, il peut diverger dans le sens où les différentes politiques de routage peuvent être en conflit et cela peut impliquer une certaine instabilité des routes. Dans [Griffin et al., 2002], la stabilité des routes est étudiée. Il y est prouvé qu'étant donné un ordre de préférence par sommet, le problème de décider la stabilité des routes est NP-complet. De plus, des propriétés sur les instances ayant des routes stables y sont extraites et, à partir de cela, un algorithme auto-stabilisant (simple) est donné. Nous revenons maintenant sur certaines de ces propriétés.

2.2 Arbre du plus court chemin stable

Nous considérons le problème du plus court chemin dans un graphe $G = (V, E)$. En algorithmique classique, des algorithmes centralisés et des algorithmes distribués voire auto-stabilisants (voir [Tel, 2000]) existent pour résoudre ce problème. Quel que soit le modèle retenu (centralisé ou distribué), les sommets collaborent pour construire une structure minimisant la même fonction objectif. Or ces algorithmes ne tolèrent pas que les acteurs impliqués suivent leur propre objectif. La suite de ce chapitre vise à adapter ces algorithmes pour cela.

En effet, nous supposerons ici que tous les sommets V de G ont d'abord un but commun : construire un arbre couvrant enraciné en un sommet distingué noté r. Par

ailleurs, ils ont aussi un autre but privé : minimiser le coût de leur chemin dans l'arbre à destination de r. Ceci peut entraîner des conflits entre les ensembles de sommets définis par une métrique commune.

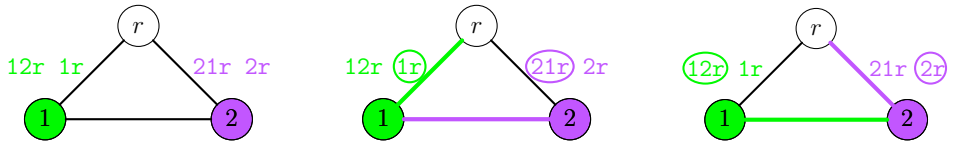
Plusieurs modélisations ont été proposées dans la littérature pour prendre en compte cette concurrence. Par exemple celles développées dans [Griffin et al., 2002] ou [10]. Dans ce document, nous allons présenter le premier travail motivé par le problème du routage inter-domaine introduit dans [Griffin et al., 2002].

Avant de définir formellement le problème, nous allons introduire quelques notations sur les chemins. Un chemin $P = v_0 v_1 \dots v_k$ est un chemin dirigé allant du sommet source v_0 vers un sommet destination v_k. De plus si P et Q sont deux chemins respectivement allant de v vers u et de u vers z, alors $P \cdot Q$ sera la notation du chemin obtenu par la concaténation de P avec Q.

Maintenant, le problème de l'arbre de routage est défini comme suit :

`Problème :` `Arbre de routage du plus court chemin stable`
`Instance :` $G = (V, E)$ un graphe simple non-orienté, un sommet distingué r ; chaque sommet v a une liste $\mathcal{P}_v$ de chemins autorisés de v vers r avec un ordre de préférence $\preceq_v$ sur les chemins.
`Objectif :` Construire un arbre tel que tous les sommets v établissent un chemin entre eux et r satisfaisant la propriété suivante : si v choisit $P = u \cdot Q$ sachant que u est un voisin de v et Q est un chemin de u vers r , alors u choisit le chemin Q.

Par exemple, dans la figure 2.1, le sommet `1` peut choisir entre deux chemins : celui qui passe par le sommet `2` que l'on notera `12r`, et celui qui va directement à `r` que l'on notera `1r`. De plus il préfère le premier chemin `12r` par rapport au second `1r` : `1r` $\preceq_1$ `12r`. Par contre le sommet `2` préfère le chemin `21r` à celui `2r` (c'est-à-dire `2r` $\preceq_2$ `21r`). Ici, les intérêts individuels de ces deux sommets ne sont pas compatibles.



Instance arbre stable T_1 arbre stable T_2

FIG. 2.1 – Instance ayant deux arbres stables T_1 et T_2

Notons qu'un arbre couvrant[1] T de G possède un unique chemin noté $T_{v \to u}$ entre n'importe quelle paire (v, u) de sommets. Le but est de construire un arbre couvrant *stable* c'est-à-dire non améliorable du point de vue de chaque sommet. En effet, l'objectif de chaque sommet v est de maximiser sa préférence de son chemin vers la racine r.

Nous formulons maintenant les définitions de sommets et d'arbres stables. Notons $\Gamma_G(v)$, le voisinage du sommet v dans le graphe G (c'est-à-dire l'ensemble des sommets voisins de v dans G).

Un sommet v est stable pour un arbre T s'il ne peut pas choisir un autre voisin $z \in \Gamma_G(v)$ qui lui permettrait d'augmenter sa préférence dans l'arbre T' engendré.

Définition 1 (Stabilité des sommets et des arbres) *Soit $G = (V, E)$ un graphe avec pour chaque sommet v une liste de chemins $\mathcal{P}_v$ avec leur préférence $\preceq_v$ et r un sommet*

[1] un arbre couvrant de $G = (V, E)$ est un arbre dont l'ensemble des sommets est V.

distingué de V. Soit T un arbre couvrant de G. Le sommet v est stable dans T *si et seulement s'il n'existe aucun chemin p dans $\mathcal{P}_v$ tel que $T_{v\to r} \preceq_v p$ et si $T_{v\to r} \in \mathcal{P}_v$. On dira qu'un arbre T est* stable *dans G si tous ses sommets sont stables.*

Considérons un graphe de trois sommets $\{r, 1, 2\}$ de la figure 2.1. Les deux autres dessins représentent des arbres stables T_1 et T_2 pour l'instance du problème. L'arbre T_1 est stable puisque le sommet 2 choisit le chemin de plus forte préférence tandis que le sommet 1 ne peut pas modifier son chemin sans perte de la propriété d'avoir un chemin entre lui et r.

Par contre, il existe des instances où aucun arbre stable existe. Par exemple, considérons l'exemple suivant repris de [Griffin et al., 2002] : le graphe de quatre sommets $\{r$, 0, 1, 2$\}$ est représenté par la figure 2.2. Remarquons, que les listes des préférences des chemins sont représentées dans l'ordre de croissant.

Pour chacun des sommets j avec $\mathtt{j} \in \{0, 1, 2\}$, le chemin de préférence la plus grande est celui passant par le voisin $(\mathtt{j}+\mathtt{1})$. Les indices doivent être interprétés modulo 3. Un arbre couvrant du graphe contient nécessairement au moins l'une des arêtes (j,r) pour atteindre le sommet r. Sous l'hypothèse que le sommet j soit sélectionné par le chemin (jr), le sommet $(\mathtt{j}-\mathtt{1})$ n'est stable que s'il sélectionne le chemin $(\mathtt{j}-\mathtt{1})\mathtt{jr}$ car c'est le chemin de plus grande préférence. Le sommet $(\mathtt{j}+\mathtt{1})$ ne peut que sélectionner le chemin $(\mathtt{j}+\mathtt{1})r$, Par conséquent, le sommet j n'est pas stable car en choisissant le chemin $\mathtt{j}(\mathtt{j}+\mathtt{1})\mathtt{r}$, il choisit un chemin de préférence la plus grande tout en conservant un chemin entre lui et r. Nous venons de démontrer qu'aucun arbre couvrant de ce graphe ne peut être stable.

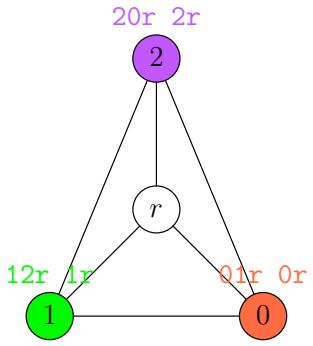


FIG. 2.2 – Instance n'admettant aucun arbre stable.

Dans la section suivante, une analyse plus fine est présentée sur les conditions d'existence et de construction d'arbres stables.

2.3 Analyse des arbres stables

L'exemple présenté dans la section 2.2 (page 14) peut se généraliser en un type de familles d'instances que l'on nommera instances de type *roue avec conflit*.

2.3.1 Conditions d'existence d'arbres stables

En effet, aucun arbre stable existe pour certains éléments de la famille d'instances décrite dans la définition 2 suivante :

Définition 2 (roue avec conflit) *Une instance* $\mathcal{I} = (G, (\mathcal{P}_v)_{v\in V}, (\preceq_v)_{v\in V})$ *est de type* roue avec conflit *(en anglais* dispute wheel*) ayant* k *rayons si et seulement s'il existe un sous-ensemble* $U = \{u_0, u_1 \ldots, u_{k-1}\}$ *de sommets de* V*, des ensembles de chemins* $\mathcal{R} = \{R_0, R_1 \ldots, R_{k-1}\}$ *et* $\mathcal{Q} = \{Q_0, Q_1 \ldots, Q_{k-1}\}$ *tels que pour* $j \in [0, \ldots, k-1]$*,*

1. *le chemin* R_j *(resp.* Q_j*) relie* u_j *à* u_{j+1} *(resp.* r*)*
2. $Q_j \in \mathcal{P}_{u_j}$
3. $R_j \cdot Q_{j+1} \in \mathcal{P}_{u_j}$
4. $Q_j \preceq_{u_i} R_i \cdot Q_{j+1}$

Les indices doivent être interprétés comme modulo k. La figure 2.3 correspond à une représentation graphique de graphe de type *roue avec conflit*.

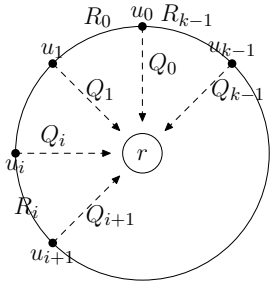


FIG. 2.3 – un graphe de type *roue* avec conflit.

Nous pouvons remarquer que l'instance représentée dans la figure 2.2 est une instance de type *roue avec conflit* de taille 3. En fait, pour les instances de ce type ayant un nombre impair de rayons, aucun arbre stable n'existe. A partir de cette constatation, il a été montré le théorème suivant :

Théorème 1 ([Griffin et al., 2002]) *Décider s'il existe un arbre de routage de plus court chemin stable est NP-complet.*

La réduction polynomiale utilisée dans la preuve de ce théorème se base sur la construction d'un graphe contenant des instances de type *roue avec conflit* à partir de 3-SAT. L'astuce de la preuve est d'associer une clause à un graphe de type roue de telle façon à ce que pour toute affectation qui ne satisfait pas une clause, alors, ce graphe de type *roue avec conflit* ne possède pas d'arbre stable (et réciproquement).

De plus,

Théorème 2 ([Griffin et al., 2002]) *Pour toute instance ne contenant pas une* roue avec conflit*, il existe un unique arbre de routage de plus court chemin stable.*

A partir de ce résultat, un algorithme réparti glouton, va pouvoir construire un arbre stable de plus court chemin. Avant de le décrire, nous allons introduire quelques notions classiques.

2.3.2 Algorithme réparti

L'algorithme localement exécuté sur chaque sommet est présenté au moyen de variables partagées et d'actions. Chaque sommet possède ses propres variables et communique avec les autres processeurs par passage de messages par des canaux de communication FIFO.

Un système de transitions est un couple $S = (C, \rightarrow)$, où C est un ensemble de configurations et $\rightarrow$ une relation binaire sur C. Une configuration est le produit cartésien à un instant donné de toutes les variables de tous les sommets du réseau. La relation $\rightarrow$ permet de modéliser les changements d'états locaux des sommets par le programme.

Une exécution de S est une suite maximale de configurations (finie ou infinie) $E = (\gamma_0, \gamma_1, \ldots, \gamma_i, \ldots)$ telle que pour tout $i \geq 0$, $\gamma_i \rightarrow \gamma_{i+1}$.

L'algorithme que l'on va considéré est simple. Chaque sommet u conserve son choix, son chemin entre lui et la destination dans une variable locale. Lors de la réception d'un message d'un sommet voisin v, le sommet u conserve cette information dans une table et il vérifie que son choix courant est bien le meilleur. Si ce n'est pas le cas il le réactualise et envoie son nouveau chemin à tous ses voisins. Le traitement local se réalise de façon atomique[2].

Théorème 3 ([Griffin et al., 2002]) *Cet algorithme glouton converge vers un arbre stable si l'instance ne contient pas de roue avec conflit.*

Pour cela, il a été prouvé qu'à partir de toute exécution débutant par une configuration arbitraire, un arbre stable est construit (propriété de sûreté[3]).

Il est à noter que les graphes contenant des *roues avec conflit* peuvent parfois avoir des arbres stables. Il suffit par exemple d'étendre l'exemple de 2.1 pour former *une roue avec conflit* de 4 rayons. Cet algorithme ne converge pas sur cet exemple, et cela implique qu'il peut diverger pour certains types d'exécutions même si l'instance du problème possède un arbre stable.

Un algorithme pour y remédier a été proposé par [Griffin and Wilfong, 2000]. Il détecte des cycles liés à des conflits de politiques grâce à l'historique des chemins. Plusieurs améliorations ont été proposées ultérieurement en utilisant des mécanismes de jetons [Ahronovitz et al., 2006], des algorithmes aléatoires [Ibrahim and Matta, 2004].

Dans [Griffin et al., 2002], la seule hypothèse sur l'ordre de préférence est qu'elle est fixée, connue dès le départ et surtout qu'elle ne change pas au cours du temps. Que se passe-t-il si l'ordre de préférence peut se modifier au cours du temps ? Cela revient à comprendre quelles sont les propriétés (s'il en existe) de l'ordre de préférence qui permettent de conserver la convergence de l'algorithme de [Griffin et al., 2002]. A ma connaissance, ces questions ne sont pas complètement fermées. Nous considérons dans la section suivante des réponses partielles en considérant des algorithmes auto-stabilisants.

Définition 3 (algorithme autostablisant) *Un système de transition $S = (C, \rightarrow)$ est* auto-stabilisant *pour la spécification P (un prédicat sur l'ensemble des exécutions) s'il existe un ensemble $L \subseteq C$ de configurations légitimes vérifiant les propriétés suivantes :*

1. Correction : *toute exécution débutant par une configuration dans L satisfait P ;*
2. Convergence : *toute exécution atteint une configuration dans L.*

[2] cette procédure ne peut pas être interrompue.

[3] toute exécution débutant par une configuration arbitraire atteint une configuration satisfaisant le prédicat P.

2.4 Arbres stables avec couleurs

Dans [10], [42], nous avons étendu le problème de la construction d'arbre stable de plus court chemin au cas où les sommets pourraient ne pas avoir les mêmes politiques de routage.

Pour cela, l'ensemble des sommets V est découpé en p parties disjointes $V_1, V_1, V_2, \ldots, V_p$ tel que $V = \bigcup_{i=1}^{p} V_i$: chaque élément de cette partition correspondant à une couleur est associée à un ordre de préférence.

Dans [10], [42] et [Dasgupta et al., 2006], l'ordre de préférence est défini comme l'opposé du coût d'un chemin : moins le coût d'un chemin est élevé, plus le joueur préfère le chemin. Pour modéliser le coût, le graphe G est pondéré par une fonction $w : E \to \mathbb{N}^{*p}$. Pour chaque entier $i \in [1 \ldots p]$, la fonction $w_i : E \to \mathbb{N}^*$ est définie comme

$$\forall e \in E,\ w_i(e) = x_i \text{ si et seulement si } w(e) = (x_1, \ldots, x_i, \ldots, x_p).$$

Le coût du chemin C pour la couleur i est $w_i(C) = \sum_{e \in C} w_i(e)$. Par rapport aux notations précédentes, pour un sommet v, l'ensemble des chemins autorisés $\mathcal{P}_v$ est l'ensemble de tous les chemins le reliant à r. Nous pouvons remarquer que dans certains graphes, la cardinalité de cet ensemble peut être exponentielle par rapport au nombre de sommets du graphe.

L'ordre $\preceq_v$ pour le sommet v se définit alors de la façon suivante : pour tout couple (P, Q) de deux chemins, nous avons

$$P \preceq_v Q \text{ si et seulement si } w_i(Q) \leq w_i(P)$$

sachant que le sommet v est de couleur i.

Plus formellement,

`Problème :`	`Arbre du plus court chemin stable`
`Instance :`	$G = (V, E)$ un graphe non-orienté, un sommet distingué r ; ayant une fonction poids sur les arêtes $w : E \to \mathbb{N}^{*p}$.
`Objectif :`	Construire un arbre T tel que tous les sommets v établissent un chemin entre eux et r et tel que T soit stable, i.e : $\forall z \in \Gamma_G(v)$, $w_i(z \cdot T_{z \to r}) \geq w_i(T_{z \to r})$ sachant que v a la couleur i.

Nous pouvons constater que l'exemple de la figure 2.2 correspond aussi à un problème d'arbre stable dans le cas où le nombre de couleurs est égal à trois. Par contre, puisque chaque sommet a son propre ordre de préférence, cet exemple ne correspond pas à une instance du problème d'arbre stable ayant deux couleurs.

Dans [10] nous avons construit des exemples d'instances ne possédant pas d'arbre stable et prouvé :

Théorème 4 ([10]) *Décider s'il existe un arbre de plus court chemin stable est NP-complet même si le nombre de couleurs est* 2.

De plus, nous avons décrit dans [42] des familles d'instances possédant au moins un arbre stable.

Contrairement à ce qui se passe dans la section précédente, il n'y a pas unicité de l'arbre. Pour cette raison, il a été prouvé dans [Dasgupta et al., 2006] que l'algorithme glouton est seulement faiblement auto-stabilisant.

Théorème 5 ([Dasgupta et al., 2006]) *L'algorithme glouton calcule un arbre stable et il est faiblement auto-stabilisant.*

Notons, qu'un système est *faiblement auto-stabilisant* s'il existe une exécution qui atteint une configuration légitime à partir de toute configuration de départ. En fait, par rapport à la notion d'auto-stabilisation classique, seule la propriété de convergence est modifiée : pour la notion d'auto-stabilisation faible, seule une exécution atteignant une configuration légitime suffit contrairement à la notion de (fortement) auto-stabilisant où toutes les exécutions doivent atteindre une configuration légitime.

Nous avons étendu le résultat précédent :

Théorème 6 ([42]) *L'algorithme glouton converge vers un arbre stable de façon auto-stabilisante si l'instance ne contient pas certains motifs (roues avec conflit généralisées).*

Mais cet algorithme est complexe en terme d'échanges de messages puisqu'il est obligé de construire pour chaque sommet tout le graphe avec sa métrique de coût.

Nous conjecturons que l'algorithme proposé dans [Dasgupta et al., 2006] est en fait auto-stabilisant pour les mêmes instances que le théorème 6. Cette conjecture s'inspire d'arguments liés aux travaux sur les jeux de potentiel ordinal (voir les chapitres 5, 7) dont nous parlerons plus longuement lors de l'étude de ces jeux.

A ce jour, nos travaux portent essentiellement sur l'ordre de préférence évoqué plus haut basé sur le coût (longueur) des chemins. Un de mes travaux en cours [35] porte sur la construction d'un arbre *de flot maximum* (en anglais *maximum flow tree*). Les résultats sont identiques à ceux obtenus pour le problème de la construction du plus court chemin pour cette métrique. Cependant, les techniques de preuve diffèrent et il nous paraît difficile de généraliser ces résultats pour un ordre de préférence générique ou pour de larges classes de métriques.

2.5 Analogie avec la théorie des jeux

Dans le contexte précédent, chaque sommet v a son propre objectif correspondant à l'ordre de préférence $\preceq_v$. Cette optique est proche de celle de la théorie des jeux permettant de modéliser la concurrence entre deux ou plusieurs joueurs via un jeu. C'est pour cette raison que nous introduisons ici certains concepts de la théorie des jeux.

Définition 4 (jeu) *Un* jeu *(non-répété) est composé d'un ensemble de joueurs J. Chaque joueur i possède un ensemble noté S_i de* stratégies pures. *Notons $\mathcal{S}$ le produit cartésien des ensembles de stratégies pures $\mathcal{S} = \times_{i \in J} S_i$. Un* profil pur *de stratégies pures $s = (s_1, \ldots, s_n)$ est un élément de $\mathcal{S}$.*

Pour chaque profil $s \in \mathcal{S}$, chaque joueur $i \in J$ possède une *fonction de coût* $c_i : \mathcal{S} \to \mathbb{R}$ qui représente l'ordre de préférence. Il est à noter, que traditionnellement dans la théorie des jeux, la notion de *fonction d'utilité* joue le rôle de modélisation de la préférence d'un joueur : chaque joueur cherche à maximiser son utilité.

Par contre, dans nombre de contextes liés à la théorie algorithmique des jeux, il est plus naturel de considérer des coûts : chaque joueur cherche à minimiser son coût. Toutes les définitions liées à la théorie des jeux seront exprimées dans ce document par rapport aux coûts. On peut toujours voir un coût comme l'opposé d'une utilité.

La représentation sous la *forme normale du jeu* est un triplet $(J, \mathcal{S}, (c_i)_{i\in J})$ avec J ensemble des joueurs, $\mathcal{S}$ ensemble des profils, et $(c_i)_{i\in J}$, l'ensemble des fonctions de coût.

Définition 5 (Equilibre de Nash pur) *Un* équilibre de Nash pur *est un profil pur* $s = (s_1, \ldots, s_n)$ *tel que*

$$\forall i \quad \forall s'_i \in S_i \quad c_i(s_1, \cdots, s_i, \cdots, s_n) \leq c_i(s_1, \cdots, s'_i, \cdots, s_n).$$

Par exemple, dans le problème de l'arbre de routage de plus court chemin stable, l'ensemble de joueurs J correspond à l'ensemble V des sommets du graphe. L'ensemble des stratégies de chaque joueur v est la liste $\mathcal{P}_v$ de chemins autorisés et la fonction de coût de v correspond à son ordre de préférence $\preceq_v$ sur les chemins pour [Griffin et al., 2002] et aux coût des chemins dans [10], [42] et [Dasgupta et al., 2006]. Dans le jeu représenté par la figure 2.1 , il existe deux joueurs : le joueur `1` ayant `1r`, `12r` comme stratégies pures et le joueur `2` ayant `2r`, `21r` comme stratégies pures.

Plus précisément, la fonction de coût quantifie l'ordre de préférence des joueurs correspondant en fait à une liste triée des différentes stratégies (chemins). Il n'existe pas de règle universelle pour quantifier la notion de préférence d'un joueur entre les stratégies et par rapport à l'extérieur (stratégies des autres joueurs). Par exemple, les tableaux de la table 2.1 du jeu défini à la page 13 représenté par la figure 2.1 représentent des coûts de différents joueurs qui respecte l'ordre de préférence du jeu . Bien sûr, il y en a d'autres.

		joueur 2	
		21r	2r
joueur 1	12r	2	0
	1r	1	1

Coût du joueur 1

		joueur 1	
		12r	1r
joueur 2	21r	2	0
	2r	1	1

Coût du joueur 2

TAB. 2.1 – Coût du jeu représenté dans la figure 2.1

La représentation sous la *forme stratégique du jeu* de 2 joueurs est un tableau tel que chaque élément du tableau représente un profil du jeu et tel qu'il contient les différents coûts des deux joueurs. La première colonne (resp. ligne) représente toutes les stratégies pures du joueur `1` (resp. `2`). Chaque case du tableau, contient un couple dont le premier (resp. deuxième) élément est le coût du joueur `1` (resp. `2`) si le jeu est dans ce profil. Voir la table 2.2 pour avoir la forme stratégique du jeu défini à la page 13 représenté par la figure 2.1 .

		joueur 2	
		21r	2r
joueur 1	12r	(2,2)	(0,1)
	1r	(1,0)	(1,1)

TAB. 2.2 – Coût du jeu représenté dans la figure 2.1

La table 2.3 représente la correspondance entre les différentes notions introduites dans la section 2.2 pour le problème de l'arbre de plus court chemin et la théorie des jeux.

Nous pouvons interpréter l'algorithme de [Griffin et al., 2002] comme un algorithme d'apprentissage d'équilibre de Nash pur. A chaque étape tous les joueurs ont choisi leur

Théorie des jeux	Problème de plus court chemin
Joueurs	Sommets du graphe
Ensemble de stratégies	Ensembles des chemins
Fonction d'utilité	Ordre de préférence
Fonction de coût	L'opposé de l'ordre de préférence
Équilibre de Nash	Arbre stable

TAB. 2.3 – Équivalence entre la théorie des jeux et l'arbre de plus court chemin stable

stratégie pure. A la fin de cette étape, un seul joueur réactualise sa stratégie pure en sélectionnant toujours sa meilleure réponse. S'il y a convergence, l'algorithme converge vers un arbre stable : un arbre stable correspond à un équilibre de Nash pur au sens de la théorie des jeux. Cet algorithme correspond à une approximation d'une dynamique de la meilleure réponse. Ces termes vont être définis et discutés beaucoup plus longuement dans le chapitre 8.