Optimisation dynamique des temps d'arrêts en station

4.4.1 Rappels des objectifs

Cette section a pour objectif d'adapter les travaux présentés au chapitre 3 sur l'optimisation offline des temps d'arrêt en station pour les rendre utilisables dans une application en temps réel.

Pour cela deux défis doivent être relevés. D'une part réduire le temps de calcul des méthodes d'optimisation et d'autre part intégrer les aléas d'exploitation qui peuvent survenir dans le processus décisionnel.

Deux options peuvent être envisagées : premièrement, une amélioration des caractéristiques de l'unité de calcul et deuxièmement un apprentissage des résultats issus de l'optimisation offline. Dans un contexte industriel, cette première solution serait envisageable mais non souhaitable pour des raisons de logistique et de coûts.

La deuxième solution est assez semblable à celle qui a été mise en œuvre dans la section précédente et consiste à déduire des résultats de l'optimisation une politique de décision robuste permettant de définir le temps de stationnement optimal de chaque train en station.

Généralement, les travaux portant sur la commande optimale adoptent une démarche qui consiste à calculer une trajectoire optimale puis à construire un contrôleur pour suivre la consigne du système. Cette approche est adaptée à certains domaines comme l'aéronautique ou l'animation 3D, où une trajectoire optimale peut être connue a-priori.

Cependant, dans de très nombreux domaines comme la robotique ou la finance, la dynamique du système n'est généralement pas prévisible, ce qui rend cette approche inefficace.

Pour palier à ces inconvénients, de nouvelles méthodes mathématiques et algorithmiques ont vu le jour pour calculer en ligne de nouvelles trajectoires optimales et ainsi adapter la stratégie suivie par le système.

[172] dresse ainsi une étude assez complète concernant les méthodes mathématiques qu'il est possible d'utiliser pour réaliser cet objectif.

4.4.2 Etat de l'art sur l'optimisation dynamique

Dans des conditions réelles d'exploitation, des perturbations de trafic influencent les temps de parcours des trains, ainsi que les horaires d'arrivées et de départs en stations. Une régulation de trafic est utilisée pour gérer ces perturbations de trafic et permettre aux trains de respecter au maximum la table horaire initiale. Ainsi, un aléa subi par un train peut être propagé à d'autres trains présents sur le réseau pour respecter les

contraintes techniques d'exploitation définies par l'exploitant.

Dans cette section, une table horaire est dite robuste si celle-ci a la capacité d'absorber les aléas de trafic qui se produisent en temps réel.

Un grand nombre d'études ont été dédiées à la conception de tables horaires robustes comme [173], [174],[175] ou encore [176]. Cependant, aucune planification hors-ligne ne peut être assez robuste pour absorber tous les types d'aléas sans compromettre les performances de la table horaire. De fait, il est nécessaire de modifier en temps réel le planning horaire initial pour s'adapter aux modifications imposées par les perturbations de trafic. Néanmoins, dans la pratique, la robustesse d'une table horaire est largement dépendante de la régulation de trafic utilisée.

En 2015, Corman [177] a établi un état de l'art très exhaustif des travaux concernant les différentes approches employées dans la littérature pour effectuer une replanification en temps réel des tables horaires de lignes ferroviaires. De cette étude, plusieurs enseignements peuvent être tirés.

D'une part, il est nécessaire d'avoir une bonne connaissance en temps réel du système ferroviaire à replanifier (position, profil de vitesse, aléas de trafic,...). Ce premier point est un pré-requis indispensable pour envisager une replanification efficace.

D'autre part, une vision probabiliste des états futurs du système est obligatoire pour obtenir une aide à la décision pertinente et précise. En effet, sans connaissance de l'évolution future du réseau ferroviaire, l'optimalité de la replanification ne peut être assurée et est même fortement compromise, puisque cela revient à effectuer une étude statique d'un système dynamique en connaissant uniquement les informations du système au pas de temps courant. Cette vision probabiliste est assimilable à la capacité d'anticiper l'évolution du système pour effectuer la replanification.

En outre, [177] met en lumière que la nature de la modélisation influence également les performances de la replanification, que ce soit pour l'atteinte des objectifs initiaux, la rapidité de la prise de décision ou le degré de précision de la planification (optimalité de la prise de décision sur un horizon temporel).

Enfin, cette étude démontre surtout qu'il existe une très grande quantité d'approches différentes pour réaliser une même macro-tâche. Chacun de ces travaux présente des spécificités, des avantages et désavantages qui dépendent avant tout des informations disponibles pour effectuer une replanification optimale en temps réel ainsi que des objectifs visés. De fait, aucune méthode universelle ne résulte de cet état de l'art, pour effectuer une optimisation dynamique des temps d'arrêt en station dans une ligne ferroviaire de type métro automatique.

L'optimisation temps réel des temps d'arrêt en station semble donc être liée à deux contraintes majeures : la rapidité d'exécution de la boucle d'optimisation et la connaissance probabiliste de l'évolution future de l'état du système.

Il a alors été choisi d'étudier la possibilité d'utiliser une approche par apprentissage par renforcement (AR) pour réaliser cette tâche d'optimisation en temps réel.

4.4.3 Définition de l'apprentissage par renforcement

De tous temps, l'homme a utilisé un système de récompense/punition (également appelés *renforceurs*) pour conditionner les actions des animaux qu'il souhaitait domestiquer.

En associant ces signaux de renforcement aux actions de l'animal, il est possible de lui faire adopter des comportements différents suivant la nature du renforceur : l'animal va chercher à maximiser les récompenses reçues et à minimiser ses punitions [178].

Le cerveau humain exploite ce principe de récompense/punition à travers la sécrétion de dopamine produite par les ganglions de la base afin de renforcer les connections synaptiques entre les neurones. Les travaux de Berridge et Robinson [179] constituent en ce sens une étude intéressante sur le fonctionnement neuro-chimique du système nerveux humain.

L'apprentissage par renforcement définit l'interaction entre un agent et son environnement : dans un état s_n de l'environnement, l'agent choisit et exécute une action a_n ce qui provoque une transition vers l'état s_{n+1}. L'agent reçoit alors le signal de renforcement r_n correspondant au bénéfice que l'action a_n a eu sur l'atteinte de l'objectif de l'agent.

Ce signal de renforcement est ensuite utilisé pour améliorer la politique de décision, à savoir, la séquence d'action optimale à effectuer pour maximiser les récompenses reçues et ainsi atteindre l'objectif fixé.

Dans cette section, un épisode est défini comme une série d'expériences permettant à un agent de partir d'un état initial et d'arriver à un état terminal. L'agent se déplace d'état en état en effectuant des actions et en observant les signaux de renforcement qui vont orienter sa recherche d'une politique optimale.

4.4.3.1 Processus de décision markovien

En 1957, Bellman pose les bases de l'apprentissage par renforcement en introduisant la notion de programmation dynamique, dans le but de concevoir des méthodes de contrôle optimal pour des systèmes dynamiques discrets et stochastiques. Dans [180], il explicite la notion de *processus de décision markovien* (PDM), afin de caractériser l'évolution d'un système suivant une succession d'états distincts, où les actions entreprises dépendent d'une fonction probabiliste de transition.

La figure 4.17 illustre le déroulement d'un PDM pour un agent qui part d'un état initial s_0 pour atteindre un état final s_n. Chacune des actions entreprises dans les différents états amène l'agent à recevoir une récompense.

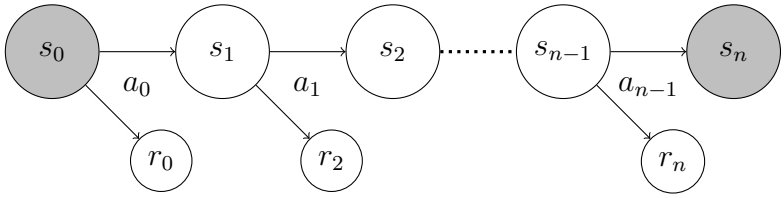


FIGURE 4.17 – Illustration d'un processus de décision markovien.

En 1988, Sutton et Barto effectuent une formalisation mathématique des algorithmes d'apprentissage par renforcement et généralisent le principe de commande optimale par l'utilisation de PDM [181], [182].

Un PDM se caractérise par le fait que l'agent doit résoudre séquentiellement des problèmes de décisions, où chaque décision courante influence la résolution des problèmes suivants et également, par l'incertitude des conséquences de chacune des actions possibles dans un état donné.

Ce type de problème est formalisé mathématiquement par le quadruplet $\{S, A, P, R\}$, où S est l'ensemble des états du système, A est l'ensemble des actions pouvant être accomplies par l'agent, P est une fonction de transition représentant la probabilité pour l'agent de se retrouver dans l'état suivant s' après avoir effectué l'action a dans un état s et R est la fonction de récompense du système représentant la probabilité pour l'agent de recevoir une récompense r en ayant effectué l'action a dans un état s. Une variable T représentant l'axe temporel des décisions peut également être introduite lorsque l'évolution du système ou les paramètres du PDM sont dépendants du temps [183].

Un problème dépendant du temps se caractérise notamment par le principe de causalité : il n'est pas possible de revenir en arrière une fois qu'une action a été effectuée.

La résolution d'un PDM consiste alors à contrôler l'agent pour qu'il effectue les actions lui permettant d'optimiser les récompenses reçues tout au long de son interaction avec l'environnement.

La notion de *politique* ou *stratégie*, notée π, est introduite pour qualifier la décision qui amène l'agent à effectuer une action dans un état donné. Un état du système est alors défini par (4.24), où $\pi(s_t)$ représente l'action effectuée dans l'état s_t sous la politique π.

$$s_{t+1} = P(s_t, \pi(s_t)) = P(s_t, a_t) \tag{4.24}$$

4.4.3.2 Critères de performance

La définition de critères de performance ambitionne de caractériser la politique suivie par l'agent apprenant. Différents critères de performances peuvent être employés pour évaluer une politique en mesurant le cumul des signaux de renforcement espérés le long d'une trajectoire.

Un critère fini C_f est une somme des récompenses immédiates et futures qui assigne le même poids à toutes les étapes de la trajectoire tandis qu'un critère actualisé C_a introduit la notion de facteur de dépréciation pour régler l'importance des récompenses futures par rapport aux récompenses immédiates.

L'expression de l'espérance du critère fini est rappelée par l'équation (4.25) où m est la longueur de la trajectoire à optimiser. L'expression du critère actualisé est donnée par l'équation 4.26.

$$C_f = E[\sum_{n=0}^{m} r_n | \pi, s_0] \tag{4.25}$$

$$C_a = E[\sum_{n=0}^{\inf} \gamma^n r_n | \pi, s_0] \tag{4.26}$$

Le facteur γ simule la *confiance* que l'on peut avoir dans l'estimation d'une récompense future probable ; plus la valeur de γ est élevée, plus l'agent aura confiance dans la vision à long terme des récompenses reçues en suivant la trajectoire définie par la politique π.

Le critère actualisé est particulièrement utilisé pour traiter les PDM où il existe une forte incertitude sur les décisions prises sur un horizon lointain en suivant la politique courante. Dans notre étude, l'utilisation de ce critère permet de probabiliser la vision à long terme et est donc privilégiée par rapport au critère fini.

D'autres critères comme le critère total (cumul des récompenses obtenues sur un temps infiniment long) et le critère moyen (moyenne des récompenses sur l'horizon de prédiction) peuvent également être employés mais ne sont pas développés ici, car ils ne présentent pas d'intérêt pour l'étude.

4.4.3.3 Fonction valeur

La *fonction valeur* en un état donné correspond à l'espérance des récompenses immédiates et futures en suivant la politique π. Pour un critère actualisé, la fonction valeur est définie comme la somme pondérée des récompenses futures en suivant la politique π à partir d'un état s_0 (4.27), à savoir l'espérance du critère actualisé.

$$V^{\pi}(s_0) = \sum_{n=0}^{\inf} \gamma^n R(s_n, a_n) \tag{4.27}$$

Le problème de contrôle optimal consiste alors à déterminer la politique optimale notée π^* qui maximise les récompenses perçues par l'agent. Pour cela, il s'agit de sélectionner les actions qui maximisent les récompenses reçues. La fonction valeur vérifie alors l'équation de Bellman (4.28), dont l'unique solution est la fonction valeur optimale que l'on note $V^{\pi *}$.

Le principe d'optimalité de Bellman appliqué à la fonction valeur permet de relier la valeur d'un état à la valeur de tous les états consécutifs qui peuvent être visités.

$$V^{\pi^*}(s) = \max_{a \in A}(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi^*}(s') \quad \forall s \in S \tag{4.28}$$

De (4.28), il est possible de déduire la définition de la politique optimale π^* (4.29).

$$\pi^*(s) = \arg\max_{a \in A}(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi^*}(s') \quad \forall s \in S \tag{4.29}$$

4.4.3.4 Fonction de valeur état-action

Un agent peut également caractériser ses interactions avec l'environnement via la *fonction de valeur état-action* Q, que l'on peut retrouver sous l'appellation *fonction Qualité*. Cette fonction mesure la qualité d'une paire état-action pour une politique π fixée.

La fonction valeur se définit ainsi comme la moyenne des $Q^\pi(s,a)$ pondérées par la probabilité de chaque action (4.30a).

La fonction Q est particulièrement utile pour déterminer le comportement optimal le plus probable d'un agent situé dans un état courant. L'expression de la fonction état-action optimale est reprise par (4.30b).

$$\begin{cases} V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a) & \text{(4.30a)} \\ Q^{\pi^*}(s,a) = (R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi^*}(s') & \text{(4.30b)} \end{cases}$$

4.4.4 Paramétrage de l'apprentissage par renforcement

4.4.4.1 Model-free vs Model-based / exploration vs exploitation

Il convient de différencier les méthodes d'apprentissage par renforcement utilisant un modèle de l'environnement (*model-based*) et celle sans modèle (*model-free*). L'apprentissage model-based a pour objectif d'apprendre les fonctions de transitions P et de récompenses R pour déterminer la stratégie optimale π^*. A l'inverse, l'apprentissage model-free tente d'estimer directement la fonction de valeur état-action Q^* sans étudier les fonctions P et R.

Ces deux modes d'apprentissage par renforcement s'opposent par quelques propriétés caractéristiques. L'apprentissage model-based nécessite une longue phase d'exploration pour déterminer précisément pour chaque point de fonctionnement du système les fonctions P et R afin d'en déduire ensuite une politique décisionnelle optimale. Dans le cas de problèmes possédant un vaste espace état-action, ce type d'apprentissage peut donc s'avérer extrêmement gourmand en temps de calcul voire, dans le pire des cas, impossible à mettre en œuvre pour explorer l'ensemble des couples état-action.

L'apprentissage model-free, est une approche gloutonne dans le sens où lors de l'apprentissage, l'agent va tenter de maximiser les récompenses reçues en exploitant systématiquement les actions menant aux valeurs de $Q(s,a)$ les plus grandes. Cette méthode est donc à dominante exploitante et est en théorie plus rapide qu'une méthode model-based, puisqu'il n'est pas nécessaire d'explorer tous les couples état-action possibles pour obtenir une politique décisionnelle optimale [8] .

En pratique, il n'est pas souhaitable d'utiliser une méthode purement exploratrice ou exploitante. En effet, si on se ramène au problème d'optimisation de la consommation énergétique d'une ligne de métro, il peut s'avérer nécessaire de dégrader une synchronisation de trains à un instant pour générer une meilleure synchronisation dans le futur.

En outre, en considérant un exemple totalement décorrelé de la présente étude : l'apprentissage d'un jeu d'échec, il est parfois nécessaire de sacrifier une pièce pour se trouver par la suite dans une position plus favorable.

8. Plus généralement, les politiques obtenues par cette approche sont dites sub-optimales car pour des problèmes complexes, il est possible de juger si une politique mène à l'objectif fixé, mais il n'est souvent pas possible de connaitre la politique optimale du problème.

Cette notion est connue sous le nom de compromis exploration-exploitation. L'étude de ce compromis a fait l'objet de nombreuses publications comme [184].

Lorsque l'atteinte de l'objectif nécessite de passer par de nombreux états présentant des récompenses négatives ou de faibles signaux de renforcement, une politique à dominante exploratrice est plus appropriée. A l'inverse, dans le cas de jeu de plateaux (échecs, othello, backgammon,...), une stratégie basée majoritairement sur l'exploitation peut s'avérer plus bénéfique pour trouver la politique optimale.

Dans un problème industriel présentant des fortes non linéarités, une hybride entre exploration et exploitation semble être l'approche la plus appropriée qui, bien que nécessitant un temps de calcul plus important qu'une stratégie gloutonne, permet d'obtenir une meilleure connaissance du système étudié.

Dans la suite du manuscrit, une fonction de transition basée sur une distribution de Boltzmann-Gibbs est utilisée (4.31) [185], [186].

$$P(s,a) = \frac{e^{Q(s,a)/\tau}}{\sum\limits_{b \in A(s)} e^{Q(s,b)/\tau}} \tag{4.31}$$

Dans l'équation (4.31), le terme τ est une variable dont le but est de moduler le caractère stochastique de la fonction de transition P. Une valeur élevée de τ entraîne une sélection aléatoire tandis qu'une valeur faible tend à rendre la sélection des actions gloutonne. De plus, de par sa définition, cette règle de sélection présente l'avantage de donner plus de place à l'exploration des actions possibles quand les valeurs de $Q(s,a)$ sont dans le même ordre de grandeur, et de privilégier l'exploitation quand une action semble être beaucoup plus bénéfique que les autres. De plus, l'utilisation d'une règle de décroissance progressive du facteur τ permet d'accélérer la convergence de certaines méthodes, introduites dans la suite du développement, vers la politique optimale en favorisant aux premières itérations l'exploration puis l'exploitation des résultats dans les dernières itérations [187].

4.4.4.2 Caractéristiques de l'environnement

Suivant le problème traité, l'interaction entre l'agent et son environnement peut être caractérisée selon différents aspects :

Le niveau de perception de l'agent. Dans certains cas de figures, il peut être difficile pour l'agent de percevoir l'intégralité de l'état de l'environnement.

Le déterminisme de l'espace état-action. L'environnement est dit déterministe lorsque l'état suivant du système est complètement défini par le couple état-action courant. Cependant, même dans cette éventualité, la perception qu'a l'agent de son environnement peut être incomplète s'il n'a pas accès à l'intégralité des informations concernant l'état courant de l'environnement.

La dynamique du système étudié. Un environnement est dit dynamique lorsque l'état du système peut varier au cours de la prise de décision et il est dit statique dans le cas contraire.

La nature de l'interaction. L'interaction agent-environnement est dite épisodique lorsque celle-ci peut être divisée en événements appelés épisodes. Un épisode consiste alors en un cycle perception-action-renforcement : l'agent perçoit l'état

du système puis choisit l'action appropriée à effectuer et reçoit enfin une récompense de son environnement.

La nature de l'espace état-action. Lorsque le nombre d'état et d'action possibles est fini, cet espace est dit discret, et il est continu dans le cas inverse.

L'analyse de ces aspects permet de sélectionner la méthode ou l'algorithme le plus adapté pour solutionner le problème étudié.

4.4.5 Programmation dynamique

La programmation dynamique est une méthode algorithmique dont la philosophie est de considérer qu'un problème d'optimisation est composé d'un ensemble de sous-problèmes. La solution optimale du problème global est alors obtenu à partir des solutions optimales des sous-problèmes.

Sous l'hypothèse d'un PDM avec des ensembles S et A finis, les équations issues du principe d'optimalité de Bellman permettent d'estimer les fonctions V^{π^*} et Q^{π^*} afin d'en déduire la politique optimale π^*

Dans le cadre de la résolution d'un PDM par programmation dynamique, il existe deux approches : l'itération sur les valeurs et l'itération sur les politiques.

L'itération sur les valeurs est l'approche la plus classique et se base sur l'utilisation de la formulation du principe d'optimalité de Bellman appliqué à la fonction valeur (4.32). Une légère modification de la notation de l'équation (4.28) permet de mettre en évidence une suite récurrente qui converge vers V^{π^*} [188].

$$V^{\pi^*}{}_{n+1}(s) = \max_{a \in A}(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi^*}{}_{n}(s') \tag{4.32}$$

Le principe de l'itération sur les politiques est d'utiliser une politique π_n, puis d'évaluer la fonction valeur V^{π_n} issue de cette politique et d'améliorer cette politique par une règle gloutonne pour déterminer la politique suivante π_{n+1} (4.33). L'itération se termine lorsqu'aucune évolution n'est observée entre deux politiques successives.

$$\pi_{n+1}(s) = \arg\max_{a \in A}(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_n}(s') \tag{4.33}$$

Le cadre de la programmation dynamique correspond à la résolution la plus simple et intuitive d'un PDM. D'autres méthodes dérivées de la programmation dynamique ont également été introduites par la suite pour solutionner des cas plus complexes où la dynamique du système n'est pas connue ou difficilement identifiable [188].

4.4.6 Méthodes de Monte-Carlo

Une méthode de Monte-Carlo (MC) est une technique algorithmique destinée à fournir une valeur approchée d'une quantité déterministe via un procédé probabiliste.

Un paradigme largement répandu pour illustrer cette méthode est l'estimation des probabilités de chaque issue du jeu *Pile ou Face*. En lançant un grand nombre de fois une pièce équilibrée, cela devrait mener à une équiprobabilité de tomber sur le côté pile ou le côté face de la pièce. De fait, en réalisant suffisamment d'essais d'une expérience,

il serait possible d'obtenir une estimation du comportement moyen du système lors de cette expérience.

La méthode MC consiste alors à réaliser un grand nombre d'expériences et à enregistrer pour chacune d'entre elles, les transitions et actions effectuées. Ainsi, au fur et à mesure des expériences, la fonction valeur associée à chaque état se rapproche de la valeur exacte de l'état pour la politique suivie. L'estimation de la fonction V s'apparente ainsi à un moyennage des récompenses reçues lors des différentes expériences.

En utilisant un critère actualisé, la fonction récompense R pour chaque épisode est définie par (4.34a) et la fonction valeur V est mise jour par la règle (4.34b), où m est la longueur de la trajectoire entre l'état courant et l'état terminal, tandis que M est le nombre d'expériences effectuées.

$$\begin{cases} R^n = \sum_{k=0}^{m} \gamma^k r_k & \text{(4.34a)} \\ V^\pi(s) = \frac{1}{n} \sum_{n=1}^{M} R^n & \text{(4.34b)} \end{cases}$$

Comme le rappelle [189], une méthode de Monte-Carlo est généralement utilisée pour calculer les propriétés statiques d'un système et non ses propriétés dynamiques puisqu'il est nécessaire de réaliser un certain nombre d'expériences pour en déduire une politique efficace. Cependant, sous l'hypothèse d'un nombre d'expériences suffisant, une méthode MC peut s'avérer efficace pour étudier la dynamique d'un système.

4.4.7 Méthodes de différences temporelles

Les méthodes de différences temporelles (TD pour *Temporal Difference*) sont une hybridation des méthodes de programmation dynamique et de Monte-Carlo, présentées initialement dans les travaux de [190]. Une méthode TD définit ainsi une politique décisionnelle en se basant sur une série d'expériences et sur les observations du système pour différents points de fonctionnement.

L'apprentissage par TD se subdivise en deux catégories : la prédiction et le contrôle. Le problème de prédiction consiste à déterminer la récompense actualisée en suivant la politique π à partir d'un état s, ce qui revient à prédire la fonction valeur V. Le problème de contrôle, quant à lui, consiste à apprendre la récompense actualisée obtenue en suivant la politique π, en effectuant l'action a dans un état s, ce qui revient à déterminer la valeur de la fonction Q.

La variante TD(0) est la forme la plus simple de la méthode TD. L'argument 0 de la méthode désigne le fait que l'agent a une vision minimaliste : en suivant la politique π dans un état s_k, l'agent n'a accès qu'aux informations r_k et s_{k+1}. Ces informations permettent ensuite de mettre à jour l'approximation de la fonction V, en évaluant l'évolution de V en passant de s_k à s_{k+1} via le système (4.35), où α est le taux d'apprentissage.

$$\begin{cases} {\epsilon^V}_{prediction} = r_{t+1} + \gamma {V^\pi}_n(s_{k+1}) - {V^\pi}_n(s_k) & \text{(4.35a)} \\ {V^\pi}_{n+1}(s_k) = {V^\pi}_n(s_k) + \alpha\, {\epsilon^V}_{prediction} & \text{(4.35b)} \end{cases}$$

Par rapport à la méthode de Monte-Carlo qui attend la fin d'un épisode pour mettre à jour la fonction V, cette méthode de mise à jour permet de recalculer une nouvelle politique au fil des observations de l'agent et ainsi orienter la recherche de l'agent vers les états jugés les plus prometteurs.

Néanmoins, dans un environnement dont l'évolution est incertaine, il est nécessaire de connaitre la probabilité de transition. Ainsi, il s'avère plus pertinent d'effectuer un apprentissage de la fonction Q afin d'effectuer un contrôle plus efficace du problème pour que l'agent atteigne son objectif.

4.4.7.1 Mise à jour de la stratégie

L'utilisation de la fonction valeur état-action est privilégiée car elle permet un meilleur contrôle du problème en spécifiant clairement l'intérêt d'une action dans un état donné. La mise en équation de la fonction Q est très similaire à celle de la fonction V (4.36).

$$\begin{cases} \epsilon^{Q}{}_{prediction} = r_{t+1} + \gamma Q^{\pi}{}_{n}(s_{k+1}, a_{k+1}) - Q^{\pi}{}_{n}(s_k, a_k) & \text{(4.36a)} \\ Q^{\pi}{}_{n+1}(s_k, a_k) = Q^{\pi}{}_{n}(s_k, a_k) + \alpha\, \epsilon^{Q}{}_{prediction} & \text{(4.36b)} \end{cases}$$

L'évaluation et l'amélioration des stratégies déterminées par une méthode TD peuvent se faire de deux manières.

Une méthode *on-policy* La politique suivie est mise à jour au cours des expériences réalisées. L'estimation des performances de la politique se fait en prenant des décisions basées sur cette même politique, autrement dit, le coût de l'exploration est pris en compte lors de la mise à jour de la fonction Q.

Une méthode *off-policy* La stratégie optimale π^* est estimée en suivant une politique π gloutonne effectuant à chaque fois l'action qui maximise le signal de renforcement reçu.

Le choix de l'une ou l'autre de ces méthodes pour réaliser l'apprentissage d'une tâche dépend principalement de la nature du problème à résoudre. Dans un exemple présenté dans [190] mettant en compétition une méthode on-policy et une méthode off-policy sur un même problème avec des complexités différentes, les auteurs mettent en évidence la capacité d'une méthode on-policy à apprendre à éviter les mauvaises actions. Le problème considéré consiste à déterminer la trajectoire la plus rapide allant d'un point initial à un point final dans un environnement présentant des positions néfastes pour l'agent.

Un résumé des observations de [190] est proposé dans le tableau 4.3.

En outre, dans [191], l'auteur effectue une comparaison similaire sur un jeu d'othello et montre qu'une méthode on-policy est plus rapide qu'une méthode off-policy à converger vers une bonne politique bien que la courbe d'apprentissage de la première méthode soit assez instable.

La méthode off-policy souffre ainsi de l'exploitation d'une politique non optimale lors des essais effectués tandis qu'une méthode on-policy intègre les mauvaises décisions prises dans la mise à jour de la politique.

Le choix de l'une ou l'autre de ces méthodes reste donc assez délicat à effectuer puisque ce choix dépend du problème traité. Cependant, une méthode on-policy semble plus propice à traiter des problèmes de grande dimension.

Dimension du problème	On-policy	Off-policy
n = 10	• trajectoire optimale non déterminée • évite d'effectuer les actions menant à des états néfastes	• trajectoire optimale
n = 10000	• apprentissage d'une bonne trajectoire assez rapidement • évite d'effectuer les actions menant à des états néfastes	• trajectoire optimale très lente à déterminer

TABLEAU 4.3 – Comparaison des caractéristiques des méthodes on et off-policy pour deux tailles de problèmes

4.4.7.2 Une méthode off-policy : Q-learning

La méthode Q-learning est assez similaire à la méthode TD(0), à la différence que le Q-learning met à jour la fonction Q au lieu de la fonction V.

$$\begin{cases} \delta_{off} = r_k + \gamma \max\limits_{a_k \in A(s_k)} Q^{\pi}(s_{k+1}, a_{k+1}) - Q^{\pi}(s_k, a_k) \\ Q^{\pi}(s_k, a_k) = Q^{\pi}(s_k, a_k) + \alpha\, \delta_{off} \end{cases} \tag{4.37}$$

La règle de mise à jour de Q est donnée par (4.37). Il s'agit de la version la plus simple de Q-learning avec un seul pas de vue vers l'avant. Les actions de l'agent sont déterminées selon une politique π mise à jour par une règle gloutonne.

Cette méthode présente l'avantage de ne pas prendre en compte le coût d'exploration dans l'évaluation de la fonction Q, ce qui permet de constamment utiliser la meilleure politique déterminée par la méthode.

En outre, ici, une stratégie gloutonne de sélection des actions est utilisée, cependant comme il est expliqué en section 4.4.4.1, il est possible et même préférable d'utiliser une fonction de transition différente, notamment pour intégrer la notion d'exploration (4.31).

4.4.7.3 Une méthode on-policy : SARSA

À l'inverse de la méthode Q-learning, la méthode SARSA permet d'améliorer la politique décisionnelle au fil de l'eau. L'agent sélectionne l'action à entreprendre en suivant la politique courante π puis met à jour la valeur de la fonction Q. Cette méthode tire son nom du fait qu'elle nécessite de connaitre le quintuplet $\{s_k, a_k, r_k, s_{k+1}, a_{k+1}\}$ à chacune des étapes k de l'épisode.

$$\begin{cases} \delta_{on} = r_k + \gamma Q^{\pi}(s_{k+1}, a_{k+1}) - Q^{\pi}(s_k, a_k) \\ Q^{\pi}(s_k, a_k) = Q^{\pi}(s_k, a_k) + \alpha\, \delta_{on} \end{cases} \tag{4.38}$$

La règle de mise à jour de la méthode SARSA est définie par (4.38). Cette règle se démarque de celle du Q-learning par plusieurs caractéristiques. Premièrement, l'agent connait l'action suivante qui sera effectuée et prend en compte cette information pour la mise à jour de la fonction Q. Deuxièmement, la politique suivie π bénéficie d'une amélioration continue ce qui a pour effet d'accélérer l'apprentissage et la convergence vers la politique optimale du problème [185].

Une fonction de transition similaire à (4.31) peut également être utilisée pour inciter l'agent à effectuer une exploration, puis une exploitation dans les dernières étapes du processus itératif.

4.4.7.4 Algorithme type de TD-learning

L'algorithme 9 présente un algorithme généraliste pour la mise en oeuvre des deux variantes de TD-learning : la méthode SARSA et la méthode Q-learning. Dans cet algorithme, les états et actions de l'itération suivante sont respectivement notés s' et a'.

Algorithme 9 Algorithme générique de TD-learning à un pas

```
Initialiser la politique π suivie par l'agent
Pour tout a ∈ A , s ∈ S Faire
    Initialiser la fonction Q(s, a)
Fin du Pour
Pour tout épisodes Faire
    Initialiser l'état de départ s.
    Si méthode on-policy Alors
        a ← π(s)
    Fin du Si
    Tant que L'état final n'est pas atteint Faire
        Si méthode on-policy Alors
            Recevoir la récompense r et observer l'état suivant s'
            a' ← π(s')
            Mettre à jour la fonction Q avec la règle (4.37)
            s ← s'
            a ← a'
        Sinon
            a ← π(s)
            Recevoir la récompense r et observer l'état suivant s'
            Mettre à jour la fonction Q avec la règle (4.38)
            s ← s'
        Fin du Si
    Fin du Tant que
Fin du Pour
```

Il est à noter qu'en théorie, la convergence vers une politique optimale est assurée lorsque chaque couple état-action a été visité un nombre infini de fois afin de connaitre

précisément la meilleure séquence d'actions à effectuer à partir d'un état initial pour atteindre l'objectif fixé le plus rapidement possible [192].

Dans (4.37) et (4.38), les termes δ_{off} et δ_{on} sont à l'origine de l'appellation *différence temporelle*. En effet, ces termes agissent comme un signal d'erreur qui guide l'apprentissage : si ce signal est positif, cela signifie que la récompense obtenue est supérieure à celle attendue et inversement si le signal est négatif.

4.4.7.5 Dimensionnement du signal de renforcement

La définition du signal de renforcement est une propriété fondamentale intrinsèque de toute méthode d'apprentissage par renforcement. Comme le nom de la méthode l'indique, l'agent construit des déductions logiques à partir des récompenses ou des malus reçus au cours de ses interactions avec l'environnement. Un signal de renforcement mal dimensionné entraîne un apprentissage lent, aléatoire et imprécis.

Dans notre cas d'étude, la fonction récompense R est définie comme le taux de réutilisation de l'énergie issue du freinage électrique (4.39), où $E_{freinage}$ est la quantité d'énergie électrique théoriquement récupérable et $E_{dissipee}$ est la quantité d'énergie électrique dissipée lors du freinage.

$$R = \frac{E_{freinage} - E_{dissipee}}{E_{freinage}} \tag{4.39}$$

Le principe retenu est que chaque fois qu'un train arrive en station, il utilise une partie de son temps de stationnement pour choisir l'action à effectuer (la modification de temps de stationnement dans la station). Cette modification implique une certaine synchronisation des phases de freinage/accélération entre le train concerné par l'action et les autres trains en ligne.

Les récompenses sont dimensionnées pour évaluer le taux d'énergie électrique issu du freinage récupératif par rapport à ce qui aurait théoriquement pu être renvoyé. Ainsi, à chaque phase de freinage des trains est, assignée une estimation de R dont la probabilité d'occurence décroît en fonction de son éloignement temporel par rapport à la prise de décision. La probabilité d'occurrence d'un événement correspond ici à la proportion de chance de voir se réaliser les événements prévus par rapport à toutes les éventualités possibles.

Cependant, cette probabilité d'occurrence dépend de nombreux facteurs humains et techniques, ce qui la rend difficilement identifiable. En pratique, c'est le paramètre de diffusion λ qui assure l'évolution de cette probabilité.

Le dimensionnement de la récompense doit tenir compte d'au moins quatre aspects :

- un aspect local : la prise de décision a un impact sur le taux de récupération du train concerné par l'action
- un aspect global : la prise de décision a un impact sur le taux de récupération des autres trains présents sur la ligne
- un aspect court terme concernant le taux de récupération du freinage sur la prochaine interstation
- un aspect long terme concernant le taux de récupération du freinage sur les interstations suivantes. L'horizon de prédiction peut alors être étendu indéfiniment

dans le temps en appliquant des coefficients de diffusion pour simuler l'incertitude sur l'occurence des événements prévus.

Ces différents aspects sont ensuite reliés par une combinaison linéaire afin de calculer la récompense reçue par chaque train suite à la réalisation d'une action.

La figure 4.18 représente la puissance théorique consommée par un train sur 3 interstations. A l'état initial, le train arrive en station, la récompense locale à court terme représente le taux de récupération du freinage électrique lors de la première interstation. La récompense locale à moyen terme concerne le taux de récupération durant la deuxième interstation tandis que la récompense locale à long terme concerne celui de la dernière interstation.

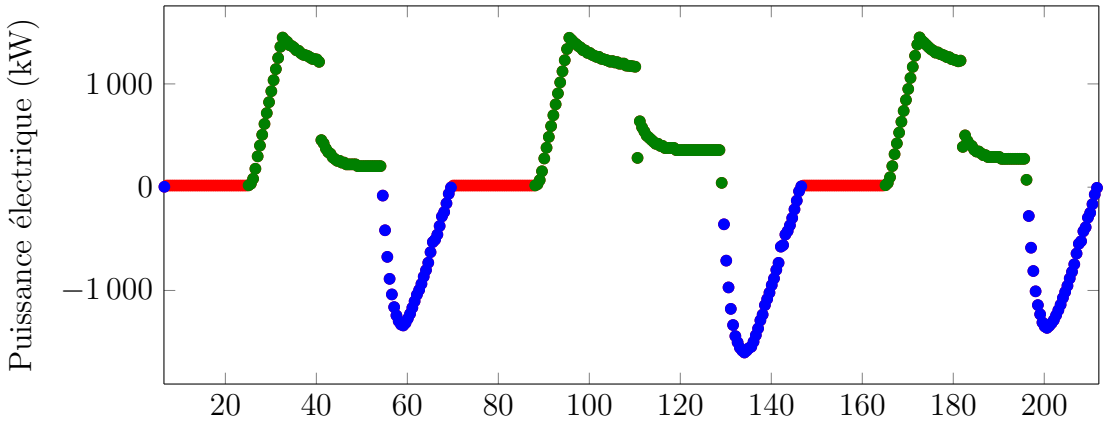


FIGURE 4.18 – Illustration du principe de récompense locale avec un horizon temporel de trois événements.

Les récompenses globales sont calculées sur les mêmes horizons temporels, mais cette fois en étudiant les taux de récupération de l'ensemble des trains du carrousel.

4.4.7.6 Exemple pratique

Le fonctionnement et la distribution des signaux de renforcement sont illustrés en considérant un cas d'étude. Un carrousel de 16 trains évoluant selon un intervalle donné est d'abord simulé, puis l'algorithme d'optimisation hybride est employé pour générer une distribution de combinaisons de temps de stationnement potentielles.

La solution optimale trouvée par l'algorithme est ensuite comparée à la solution nominale. La solution optimale permet un gain énergétique de 11% sur un tour de boucle, tandis que la combinaison nominale est la solution de référence utilisant la combinaison de temps d'arrêts nominaux (gain énergétique nul).

La figure 4.19 décrit l'évolution de la récompense locale à court terme $R^1{}_{t,local}$ vu par l'agent pour les deux combinaisons de temps d'arrêt en station sur 1000s de simulation.

L'analyse de l'évolution de la récompense locale court-terme indique que cette seule information n'est pas suffisante pour concevoir une politique efficace, en effet, l'impact énergétique de la solution optimale par rapport à la solution de référence n'est pas clairement visible.

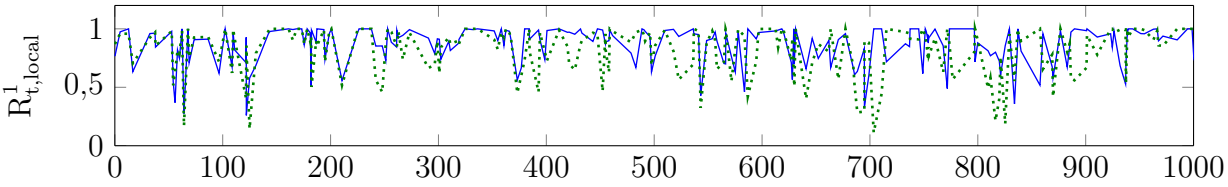
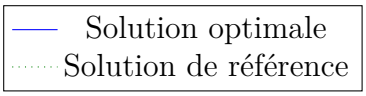


FIGURE 4.19 – Évolution des récompenses locales sur un horizon de 1000s d'exploitation.



La figure 4.20 représente l'évolution des récompenses globales pour trois horizons temporels : court ($R^1{}_{t,global}$), moyen ($R^2{}_{t,global}$) et long terme ($R^3{}_{t,global}$).

Cette figure permet de visualiser la supériorité de la solution optimale par rapport à la solution de référence pour réduire la dissipation de l'énergie du freinage.

Ces figures illustrent l'intérêt du partage des informations entre les trains. En effet, en considérant uniquement la composante locale du signal de renforcement (composante égoïste), la discrimination entre deux solutions extrêmes est difficilement réalisable.

En revanche, la composante globale (composante altruiste) fournit une information plus exploitable pour évaluer les impacts énergétiques de chacune des solutions.

En outre, comme le montre la figure 4.20, la solution optimale n'est pas dominante sur l'ensemble de l'horizon temporel.

Cela implique donc la nécessité de disposer d'un grand nombre de solutions potentielles pour en déduire la solution optimale du problème qui dominera toutes les autres quelles que soient les conditions d'exploitation.

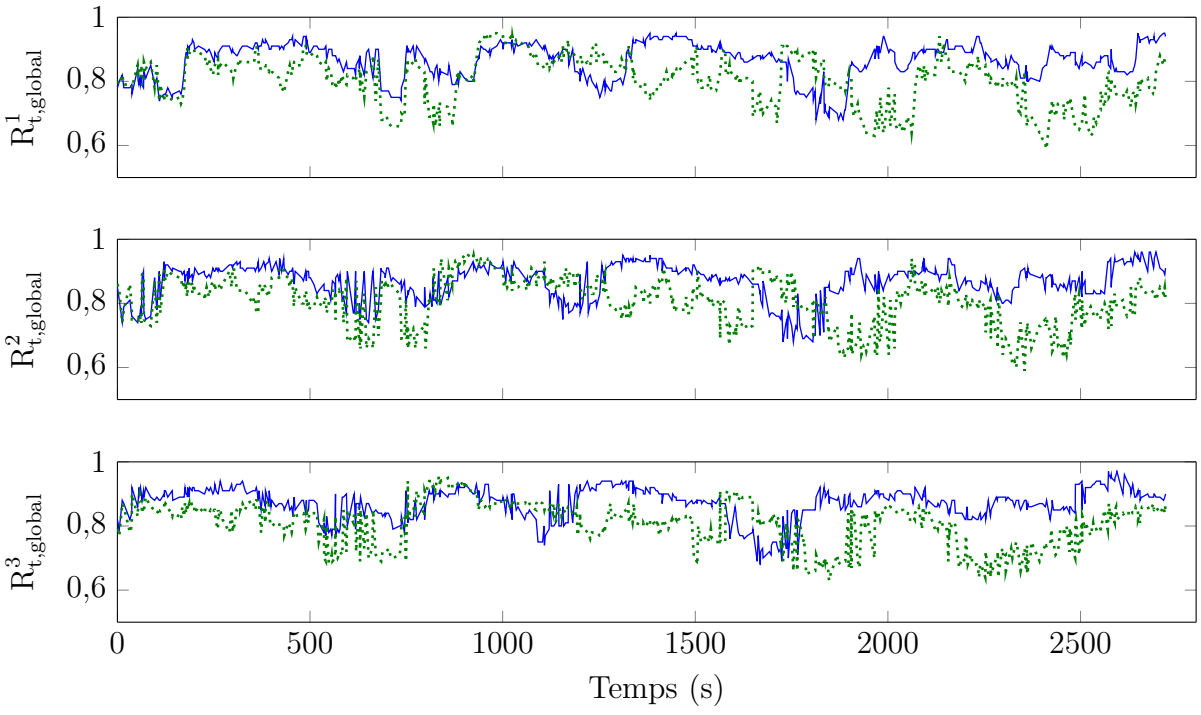


FIGURE 4.20 – Évolution des récompenses globales sur un tour de boucle complet.

4.4.8 Traces d'éligibilité

Dans les méthodes de TD-learning présentées précédemment, il a été supposé que le signal de renforcement reçu par l'agent ne concerne que la dernière transition d'état effectuée. Cependant, dans notre cas d'étude comme dans de nombreuses applications, l'état courant du système résulte d'un cheminement de l'agent parmi les états précédents et donc des actions effectuées lors des transitions précédentes. Il peut donc s'avérer utile de garder en mémoire les transitions effectuées pour caractériser la fonction Q à partir de la trajectoire globale empruntée par l'agent lors de l'apprentissage.

Les *traces d'éligibilité* exploitent ce concept en permettant de modifier la fonction Q pour l'ensemble des couples état-action visités par la trajectoire courante. De cette manière, la politique issue de l'apprentissage développe une aptitude d'anticipation et de prévision, tout en accélérant le processus d'apprentissage [192]. D'un point de vue théorique, les traces d'éligibilité font le lien entre les méthodes MC et TD.

L'un des points faibles des méthodes précédentes qui a été mis en lumière est la nécessité de réaliser un très grand nombre d'épisodes pour s'assurer d'avoir déterminé la politique optimale. Dans la méthode de Monte-Carlo, la mise à jour de la fonction V de la trajectoire suivie est effectuée une fois que l'état terminal a été atteint, ce qui permet lors de l'expérience suivante de bénéficier des connaissances du système acquises précédemment. De fait, en adaptant cette propriété aux méthodes TD, il devrait être intuitivement possible d'améliorer leurs propriétés de convergence.

4.4.8.1 Méthode TD(λ)

Un moyen de développer une politique qui possède une mémoire des états précédemment visités consiste à stocker les couples état-action visités et à propager la récompense courante aux autres transitions en les pondérant par un facteur de diffusion λ : l'erreur d'apprentissage est reportée en arrière dans le temps proportionnellement à l'ancienneté par rapport au couple état-action courant avec un amortissement régi par la valeur de λ.

La trace d'éligibilité $e_k(s,a)$ d'un couple état-action (s,a) indique à quel point la récompense courante influence la valeur de $Q(s,a)$ à la k-ième transition. Les états récemment visités doivent donc être plus fortement impactés par la récompense courante, de fait, la trace d'éligibilité est dégradée d'un facteur $\lambda\gamma$ à chaque transition.

Deux types de traces d'éligibilité sont couramment employés : les traces accumulatives et les traces avec réinitialisation.

4.4.8.2 Trace d'éligibilité accumulative

L'évolution de la trace d'éligibilité accumulative au cours des transitions est définie par le système (4.40). La trace d'éligibilité est incrémentée de 1 pour le couple état-action courant et une loi de décroissance est appliquée pour les autres couples précédemment visités.

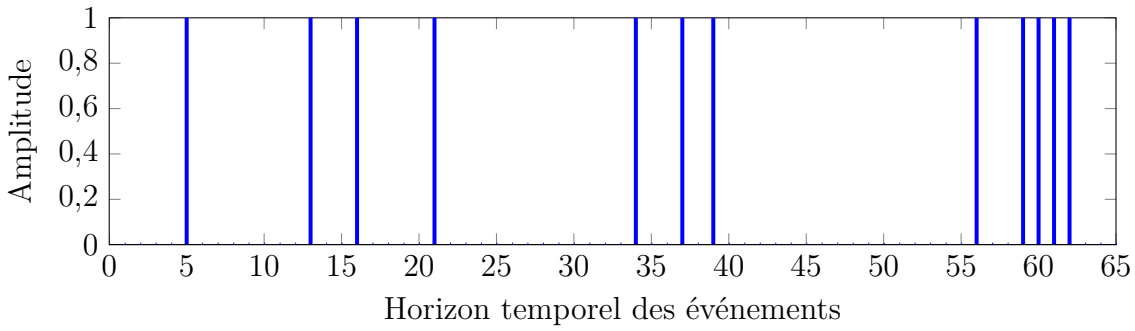
$$e_k(s,a) = \begin{cases} \lambda\gamma e_{k-1}(s,a) + 1 & \text{si } (s,a) = (s_k, a_k) \\ \lambda\gamma e_{k-1}(s,a) & \text{si } (s,a) \neq (s_k, a_k) \end{cases} \quad \forall s \in S, \forall a \in A(s) \tag{4.40}$$

Cependant, lorsqu'un couple état-action est visité plus d'une fois lors d'un épisode, la valeur correspondante de $e_t(s, a)$ devient largement plus grande que 1, ce qui a pour effet de laisser croire que l'action a effectuée dans l'état s est plus profitable qu'elle ne l'est en réalité en produisant une haute valeur de $Q(s, a)$ pour le couple considéré.

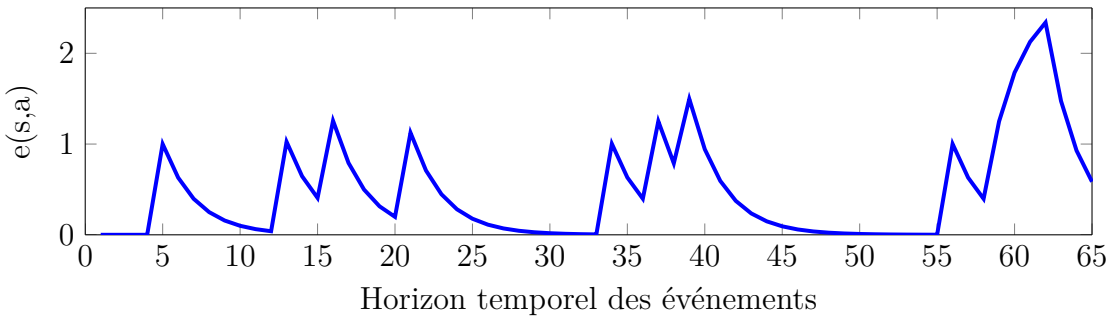
[193] a ainsi proposé une version améliorée de trace d'éligibilité : la trace d'éligibilité avec réinitialisation.

4.4.8.3 Trace d'éligibilité avec réinitialisation

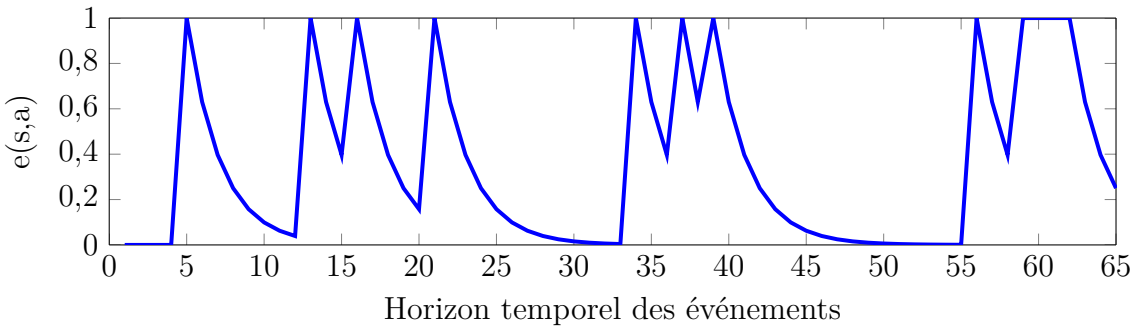
La trace d'éligibilité avec réinitialisation consiste à mettre à 1 la trace du couple courant au lieu de l'incrémenter pour éviter l'explosion de la valeur de $e(s, a)$ et de forcer à 0 la trace des actions non effectuées dans l'état s (4.41). La principale justification de cette variante fournie par [193] est que dans un environnement markovien, seule l'action effectuée dans l'état considéré est responsable des signaux de renforcement reçus par l'agent.



(a) Occurence des visites du couple (s,a) par l'agent



(b) Profil de la trace d'éligibilité accumulative



(c) Profil de la trace d'éligibilité avec réinitialisation

FIGURE 4.21 – Évolution des traces d'éligibilité en fonction de l'occurrence des visites du couple (s, a) par l'agent.

La figure 4.21 illustre l'évolution des deux variantes de traces d'éligibilité en fonc-

tion de la visite par l'agent du couple (s, a).

$$e_k(s,a) = \begin{cases} 1 & \text{si } (s,a) = (s_k, a_k) \\ 0 & \text{si } s = s_k \text{ et } a \neq a_k \\ \lambda\gamma e_{k-1}(s,a) & \text{si } s \neq s_k \end{cases} \quad \forall s \in S, \forall a \in A(s) \tag{4.41}$$

La règle de mise à jour de la fonction valeur état-action devient alors 4.42. À chaque étape k de l'épisode courant, cette règle doit être exécutée pour tous les couples état-action visités précédemment.

$${Q^\pi}_{k+1}(s,a) = {Q^\pi}_k(s,a) + \alpha\, \delta_{TD}\, e_k(s,a) \tag{4.42}$$

Le terme δ_{TD} fait référence indifféremment aux termes δ_{off} et δ_{on} définis par les systèmes (4.37) et (4.38).

Il existe dans la littérature quantité d'autres variantes des méthodes de TD-learning présentées précédemment intégrant ou non des traces d'éligibilité. Ces variantes ne sont pas explicitées car elles n'ont pas clairement démontré une efficacité[9] supérieure par rapport aux méthodes de bases énoncées dans ce manuscrit.

4.4.8.4 Récapitulatif des méthodes d'apprentissage par renforcement

La figure 4.22 présente un récapitulatif des méthodes d'apprentissage qui peuvent être envisagées pour résoudre des problèmes de type PDM, et de celles que nous avons privilégié. Il ne s'agit que d'un aperçu macroscopique et simplifié puisque dans la littérature il existe une grande quantité d'algorithmes de type TD-learning qui ont été formulés.

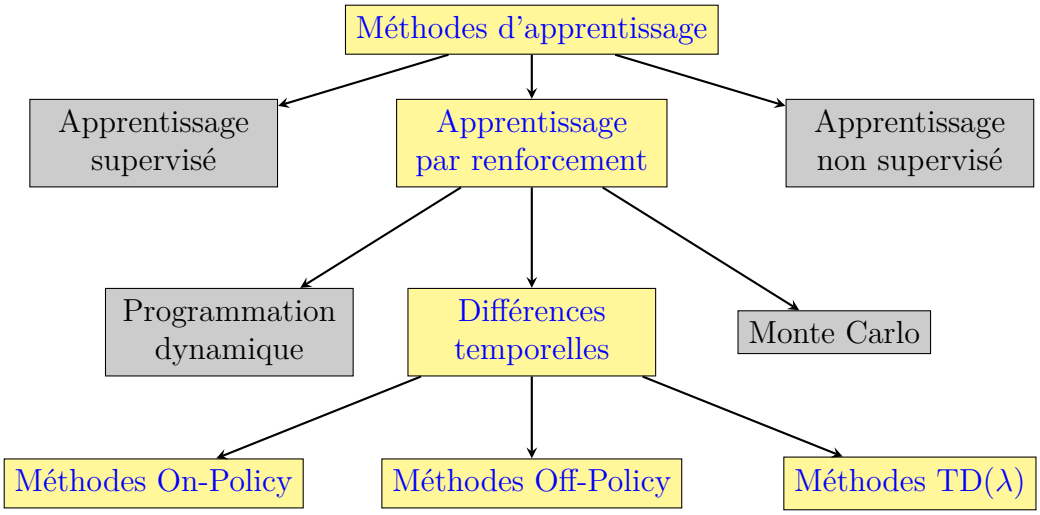


FIGURE 4.22 – Apercu des méthodes d'apprentissage par renforcement.

Par la suite cette arborescence pourra être poursuivie pour intégrer les méthodes utilisant des fonctions d'approximation pour stocker les informations inhérentes au problème traité.

9. en temps de calcul et en précision