2.1.3 Apprentissage des coûts

Supposons que nous ne sommes pas donnés les valeurs de Cost(s,a). Nous souhaitons estimer ces quantités à partir d'un ensemble d'apprentissage de chemins à coût minimaux d'actions $(a_1, a_2, ..., a_k)$.

❒ **Perceptron structuré** – L'algorithme du perceptron structuré vise à apprendre de manière itérative les coûts des paires état-action. À chaque étape, il :

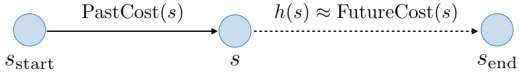
— fait décroître le coût estimé de chaque état-action du vrai chemin minimisant y donné par la base d'apprentissage,

— fait croître le coût estimé de chaque état-action du chemin y' prédit comme étant minimisant par les paramètres appris par l'algorithme.

Remarque : plusieurs versions de cette algorithme existent, l'une d'elles réduisant ce problème à l'apprentissage du coût de chaque action a et l'autre paramétrisant chaque Cost(s,a) à un vecteur de paramètres pouvant être appris.

2.1.4 Algorithme A*

❒ **Fonction heuristique** – Une heuristique est une fonction h opérant sur les états s, où chaque $h(s)$ vise à estimer FutureCost(s), le coût du chemin optimal allant de s à s_{end}.



❒ **Algorithme** – A^* est un algorithme de recherche visant à trouver le chemin le plus court entre un état s et un état final s_{end}. Il le fait en explorant les états s triés par ordre croissant de PastCost$(s) + h(s)$. Cela revient à utiliser l'algorithme UCS où chaque arête est associée au coût Cost$'(s,a)$ donné par :

$$\boxed{\text{Cost}'(s,a) = \text{Cost}(s,a) + h(\text{Succ}(s,a)) - h(s)}$$

Remarque : cet algorithme peut être vu comme une version biaisée de UCS explorant les états estimés comme étant plus proches de l'état final.

❒ **Consistance** – Une heuristique h est dite consistante si elle satisfait les deux propriétés suivantes :

— Pour tous états s et actions a,

$$\boxed{h(s) \leqslant \text{Cost}(s,a) + h(\text{Succ}(s,a))}$$