INFO0054-1 : Programmation fonctionnelle

Projet : Puzzles réguliers

Maxime Meurisse (s161278)
François Rozet (s161024)

19 avril 2019

1 Solver

Notre solver se base sur l'algorithme BFS. Ce dernier ne permet pas d'obtenir toutes les solutions d'un problème régulier, seulement les plus courtes, c.-à-d. celles qui ont été les « premières » à visiter leurs états.

Dans la méthode originale, à chaque étape, les chemins dont le dernier état a déjà été visité précédemment sont supprimés. Dans notre implémentation, nous avons décidé de les garder en mémoire. De cette façon, lorsque une solution est trouvée par l'algorithme, nous pouvons en déduire un ensemble de solutions en associant la queue, ou plutôt les queues, de cette solution avec les chemins intermédiaires stockés en mémoire.

Par exemple, si une solution est décrite par la suite d'états $(q_0, q_1, \dots, q_k, q_{k+1}, \dots, q_n)$ où q_n est un état accepteur, et qu'un chemin intermédiaire décrit par $(q_0, q'_1, \dots, q'_j, q_k)$ est stocké en mémoire, alors nous construisons la solution $(q_0, q'_1, \dots, q'_j, q_k, q_{k+1}, \dots, q_n)$.

En réalité, toutes les solutions *déduites* d'une solution trouvée par l'algorithme sont au mieux aussi courtes que cette dernière. Dès lors, tant qu'il existe d'autres solutions plus courtes, il n'est pas nécessaire de les construire.

C'est pour cette raison que, dans notre implémentation, chaque solution est enregistrée dans une paire dont le premier élément est son mot associé et le second une procédure retournant la liste des solutions qui en sont déductibles.

Ces nouvelles solutions sont-elles même des paires *mot-procédure* car, à leur tour, elles peuvent servir de bases pour en déduire d'autres. C'est en quelques sortes un « arbre paresseux ».

Il faut néanmoins prendre garde à ne pas créer de redondances.

3 Heuristique

Le solver « avec heuristique » implémente une *priority queue* dont la priorité est la somme de l'heuristique au dernier état du chemin et de la longueur de ce dernier. L'addition de la longueur permet de privilégier les chemins courts.

Nous avons implémenté les deux heuristiques de taquin H_1 et H_2. En mesurant leur temps d'exécution pour trouver un grand (100) nombre de solutions, il est apparu clairement que la seconde était plus rapide (15 s contre 2000 s) . C'est donc cette dernière que nous avons désignée comme heuristique.