Évaluation empirique des modèles d'apprentissage profond pour le problème de tournées de véhicules avec contrainte de capacité

Ali Yaddaden[1], Sébastien Harispe[1], Michel Vasquez[1]

[1] EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France
ali.yaddaden@mines-ales.fr

Mots-clés : *Apprentissage profond, Apprentissage par renforcement, Problèmes de tournées de véhicules avec contrainte de capacité, Réseaux de neurones, Active Search*

1 Introduction

Ce travail propose une évaluation empirique du cadre d'optimisation combinatoire neuronale (*Neural Combinatorial Optimization* [1]) pour résoudre le problème de tournées de véhicules avec contrainte de capacité (CVRP) [4].

Récemment, de nouvelles techniques émergentes basées sur des réseaux neuronaux profonds ont été proposées pour résoudre les problèmes de tournées [3, 2]. Celles-ci visent à apprendre, de bout en bout, une politique de résolution pour une classe de problèmes de tournées via des modèles de réseaux de neurones profonds entraînés par apprentissage (supervisé ou par renforcement). Cependant, ces modèles tendent à être inadaptés aux instances de caractéristiques différentes de celles utilisées pendant l'entraînement. En réponse à cela *Active Search* a été proposé, pour le problème du voyageur de commerce, afin de mettre en œuvre le paradigme de l'apprentissage machine en se restreignant à la seule instance à résoudre. Cet algorithme permet aux réseaux de neurones profonds de parcourir l'espace des solutions d'une seule instance afin de trouver celles de bonnes qualités.

Dans ce travail, nous partons du constat que les évaluations de ces récentes approches ne s'appuient pas sur les instances issues de benchmarks de la littérature. De telles évaluations permettraient de situer objectivement ces approches. Nous proposons donc une évaluation empirique des modèles de réseaux de neurones profonds via l'algorithme *Active Search* sur le CVRP en fournissant des résultats sur des instances bien connues de CVRPLib. [1]

2 Active Search pour les problèmes de tournées

L'algorithme *Active Search* (AS) (Algorithme 1) a été introduit par Bello et al. [1] pour entraîner un modèle de réseau de neurones profond p_θ à résoudre une instance à la fois. L'idée principale est d'utiliser un algorithme d'apprentissage par renforcement de type *policy gradient* avec un échantillonnage de type Monte Carlo pour tirer B solutions candidates suivant la distribution donnée par le réseau de neurones profond ($Y_{[\![1,B]\!]} \sim p_\theta(\cdot|X)$). La fonction de récompense utilisée par l'algorithme est la distance totale parcourue par les véhicules pour une solution Y d'une instance X donnée : $L(Y|X)$. Ce processus est itéré en T étapes pour un total de K solutions candidates. L'algorithme utilise une *baseline* de type *exponential moving average* contrôlée par un paramètre β pour réduire la variance (lignes 16-20 de l'algorithme 1). À chaque itération, l'algorithme garde la trace de la meilleure solution échantillonnée à partir du réseau de neurones. Empiriquement, il s'est avéré particulièrement efficace pour résoudre les instances du TSP de taille 20, 50 et 100 villes.

1. Instances disponibles sur http ://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Algorithm 1: Active Search

Inputs : Instance X, Model p_θ, Number of candidates K, Batch size B, Parameter β
$Y^* \leftarrow$ RandomSolution()
$C^* \leftarrow L(Y^*|X)$
$T \leftarrow \lceil \frac{K}{B} \rceil$
$Y[B]$ `// Empty array of size B`
for $t \leftarrow 1$ *to* T **do**
  for $i \leftarrow 1$ *to* B **do**
    $Y[i] \leftarrow \text{SampleSolution}(p_\theta(.|X))$
  end
  $j \leftarrow \text{ARGMIN}_{k \in [\![1,B]\!]} L(Y[k]|X)$
  $C \leftarrow L(Y[j]|X)$
  if $C < C^*$ **then**
    $Y^* \leftarrow Y[j]$
    $C^* \leftarrow C$
  end
  if $t = 1$ **then**
    $b \leftarrow \frac{1}{B} \sum_{i=1}^{B} L(Y[i]|X)$
  end
  else
    $b \leftarrow \beta \cdot b + (1-\beta) \cdot \frac{1}{B} \sum_{i=1}^{B} L(Y[i]|X)$
  end
  $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L(Y[i]|X) - b) \nabla_\theta \log p_\theta(Y[i]|X)$
  $\theta \leftarrow \text{Adam}(\theta, g_\theta)$ `// Exploitation du gradient par l'optimiseur lors de la rétropropagation`
end

Dans nos tests, nous fixons $K = 576$k solutions candidates évaluées en $T = 4500$ itérations, i.e. 128 évaluations permettent de mettre à jour les paramètres θ à chaque itération (initialisés aléatoirement). Nos résultats préliminaires ont été obtenus sur des instances de 30 à 80 clients (*sets* A et B de la CVRPLib). [2] Nous observons en moyenne un gap à l'optimal de 6.18% ($\sigma = 6\%$) pour le *set* A, et de 4.36% ($\sigma = 2\%$) pour le *set* B.

3 Conclusion et perspectives

Nos résultats sont conformes aux performances annoncées pour les modèles de réseaux de neurones profonds dans des contextes académiques artificiels. [3] Ces approches proposent aujourd'hui des performances inférieures à celles de l'état de l'art. De fortes variabilités sont aussi observées en fonction des instances CVRPLib traitées (e.g. $\sigma = 6\%$ pour le *set* A). Celles-ci mériteraient d'être expliquées en vue d'amener une meilleure compréhension et une amélioration de ces modèles. Cette étude devra être approfondie sur des instances de plus grandes tailles (e.g. CVRPLib *set* X), en considérant par ailleurs d'autres alternatives d'utilisation des modèles de réseaux de neurones (e.g., pré-entraînement, *curriculum learning*).

Références

[1] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv :1611.09940*, 2016.

[2] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems ! *arXiv :1803.08475*, 2018.

[3] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence V Snyder, and Martin Takáč. Reinforcement learning for solving the vehicle routing problem. *arXiv :1802.04240*, 2018.

[4] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

2. Celles-ci constituent une première base de test idéale puisque les tailles correspondent à celles des instances artificielles pour lesquelles ces modèles ont été initialement testés.

3. e.g., 5% de gap à l'optimal pour des instances de taille 50 générées aléatoirement [2].