Using Neural Networks to Speed Up Multi Agent Assignment Problems

Saketh Karumuri

April 19, 2025

1 Problem Formulation

Two robots want to cross a terrain represented by a string. We have a set of p terrain types $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_p\}$ Landscape q of length N is a string of terrain types that define the terrain to be crossed.

$$q \in \mathcal{T}^N = \underbrace{\mathcal{T} \times \mathcal{T} \times \cdots \times \mathcal{T}}_{N \text{ times}} \tag{1}$$

We have the Terrain Resistance for robot i: $c_i(t)$ where $t \in \mathcal{T}$. The robots can collaborate in a mode defined in the set of modes $\mathcal{M}$.

$$\mathcal{M} = \begin{Bmatrix} 0 & \text{If robots are not collaborating} \\ 1 & \text{If Robot 1 is carrying Robot 2} \\ 2 & \text{If Robot 2 is carrying Robot 1} \end{Bmatrix} \tag{2}$$

Our decision variable is σ the mode indicator.

$$\sigma : \mathcal{T}^N \to \mathcal{M}^N \tag{3}$$

Where the kth element of σ is written as σ_k. The robot i has mass μ_i. Our robot effective mass function $M_i(m)$ with collaborative mode $m \in \mathcal{M}$ indicates the effective mass of robot i for a given mode of collaboration. Our effective mass functions are defined as:

$$M_1(m) = \begin{cases} \mu_1 & \text{if } m = 0 \\ \mu_1 + \mu_2 & \text{if } m = 1 \\ 0 & \text{if } m = 2 \end{cases}, \quad M_2(m) = \begin{cases} \mu_2 & \text{if } m = 0 \\ 0 & \text{if } m = 1 \\ \mu_1 + \mu_2 & \text{if } m = 2 \end{cases} \tag{4}$$

The energy consumed by each robot for a given assignment string is

$$E_i(\sigma) = \sum_{k=1}^{N} [c_i(q_k)M_i(\sigma_k) + c_{si}\delta(\sigma_{k-1}, \sigma_k)] \tag{5}$$

$$\delta(m_i, m_j) = \begin{cases} 1 & \text{if } m_i \neq m_j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Where $\sigma_0 = 0$, q_k is the kth element of the landscape q and c_{si} is the switching cost for the robots to change collaboration modes. Finally our objective function to minimize energy consumption is

$$\min_{\sigma} \left\| \begin{bmatrix} E_1(\sigma) \\ E_2(\sigma) \end{bmatrix} \right\|_p \tag{7}$$

Where $\|\cdot\|_p$ is the pth norm and $p \in \{1, \infty\}$

2 Background

Our objective is to use a trained network that can get within 95% of the optimal solution described by the linear program. This network should also have a significant perfomance benefit to justify its use. We can assume that the robots location is shared as whenever they collaborate, their location is shared and if they are not collaborating, they will either share their location when they next collaborate or when they complete traversing the landscape. We also assume that we can discritize the landscape into $\hat{q}$

3 Method

We train on a set of terrains and optimal assignments. We use fixed terrain efficiency and switching cost, and use a mixed integer linear program to identify the optimal assignment $\sigma^\star$. We feed these as training data to a multi-layer neural network composed of 1 convolutional layer and 2 fully connected layers with dropout to remove unnecessary weights in the network. [insert figure of neural network]

3.1 Model Architecture Details

Our multilayer neural network first embeds the input into We then use a convolutional layer since for any given point x the optimal assignment at that point is heavily influenced by it and its immediate surroundings. We use a kernel size of 9. We finally use a small fully connected layer with three outputs of which the maximum value signifies the predicted optimal strategy.

3.2 Current Model Implementation

- `embed(input_dictionary = 2, embedded_size=32)`
- `ConvolutionalLayer(input=216,output=128,kenel_size=9)`
- `DenseLayer(input=128,output=32)`
- `ReLUActivation()`
- `DenseLayer(input=32,output=32)`
- `ReLUActivation()`
- `DenseLayer(input=32,output=3)`