UNIVERSITY OF GUELPH

The Analysis of Task Scheduling with Unsupervised Self Organizing Map

by

Zimo Zhou (xxxxxxx)
Uday Singh (xxxxxxx)
Yaowen Mei (1177855)

A thesis submitted in partial fulfillment for the
Summer 2021 ENGG6570 Course

in the
School of Engineering

June 2021

UNIVERSITY OF GUELPH

Abstract

School of Engineering

by
Zimo Zhou (xxxxxxx)
Uday Singh (xxxxxxx)
Yaowen Mei (1177855)

Dynamic task scheduling is a very famous open question in computer science, it is consisting in generating a task arrangement for scheduling K robots to accomplish M tasks with the minimum possible total traveling length. The same as the famous traveling sales person problem, this is also a NP-Complete problem. This implies that the time requirement for solving this problem increased rapidly as the number of tasks and number of robots increase. Due to this fact, soft computing, especially self organizing map (SOM), could possibly shining a light on finding the optimal or near optimal solution of this problem. In this paper, different task scheduling algorithms were analysed in terms of accuracy and time efficiency.

Key Words: SOM, Algorithms, NP-Complete, Soft Computing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Assume there are K homogeneous robots and M targets randomly distributed in a bounded 2D space. Each target requires **at least one** robot, and all the robots need to be assigned to one target, and can only be assigned to one target. This is to say, the task assignment, f , is a map on $\mathbf{R^2}$ from set **Robots** to set **Tasks**.

$$f : \textbf{Robots} \rightarrow \textbf{Tasks} \tag{1.1}$$

The cost function, C, is defined as the summation of robots' travelling path from their origin to their assigned tasks:

$$C = \sum_{\text{robots}} \text{Path} \quad \text{Length} \tag{1.2}$$

The challenge is to develop a good algorithm to find a traveling path for all the robots with the minimum or near minimal cost.

A visual illustration is given as shown in the picture 1.1, all the M green squares are tasks (in this case, $M = 5$, and tasks are labeled as T_0 to T_4) need to be accomplished, and all the K red dots are robots (in this case, $K = 8$, and robots are labeled as R_0 to R_8) that need to be assigned for a task. All the initial position for tasks and robots are randomly generated in the range of $[0, 5]$, and the arrangement with the lowest cost is la-bled with red lines which are generate from a Python brute-force program .

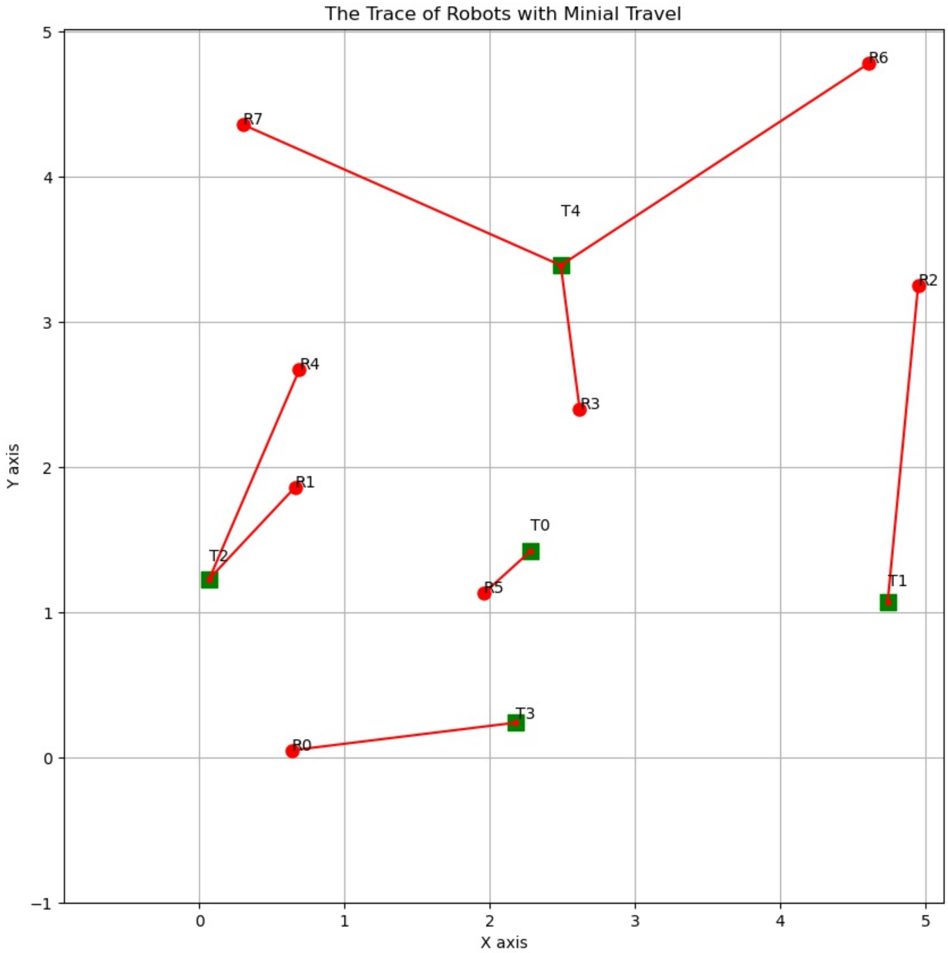

FIGURE 1.1: The illustration of Task Scheduling Problem with 5 tasks and 8 robots.

1.2 The Brute-force Method

Theoretically, at small scale, we can solve this problem and find the best solution with the Brute-force method. The time complexity will be bound by $O(M^K)$. Same as the famous traveling sales person (TSP) problem, this is a NP-complete problem as well.

M	K	Time	χ	$H(\chi)$
2	2	0.0005	2	
2	4	0.0005	14	
2	8	0.001488	254	
2	16	0.65571	65534	
2	20	13.02	1048574	

TABLE 1.1: The time complexity analysis of this problem with 2 tasks, and $K = 2^n$ robots

1.2.1 Entropy of this problem

According to Shannon's information theory[3], for a random variable X in a finite set χ with probability distribution $p(x)$, the Shannon's entropy can be written as:

$$H(X) = -\sum_{x\in\chi} p(x)\log_2(p(x)) \tag{1.3}$$

For each of the robots, it will have at most M choice to choose from as its the target. The total number of possible of arrangement is at most M^K. With considering the boundary condition that each of these M target need to have at least one robot, we can reduce the total number of possible of arrangement, χ, as:

$$\begin{aligned} \chi &= \underbrace{M(M-1)(M-2)\dots 1}_{M!} \times C^K_{K-M} M^{K-M} \\ &= M! \frac{K!}{(K-K+M)!(K-M)!} M^{K-M} \\ &= \frac{K!}{(K-M)!} M^{K-M} \end{aligned} \tag{1.4}$$

1.2.2 Time Analysis and Entropy Analysis

With a home used i7-9700K CPU (8 Cores, 3.6GHz), the maximum scale size of this problem that can be solved without memory leaking is 2 tasks and 20 robots (See Figure 1.2). Table 1.1 is a summaries the time complexity of this problem with the Brute-force method:

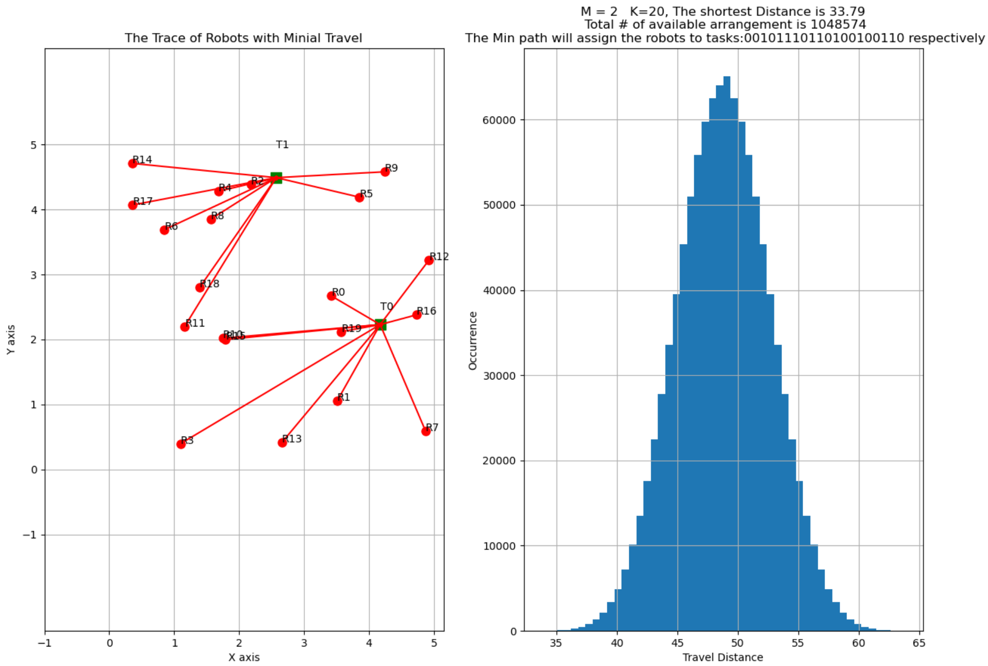

FIGURE 1.2: The upper limit of a PC is 2 tasks and 20 robots

Chapter 2

Overview of the Previous Work

2.1 The history of Self Organizing Map (SOM)

Self-Organizing Map (SOM), also know as Kohonen map, is a topological preserving map that can map a higher dimensional space to a lower dimensional space. Along this process, information will be compressed; while, the key parameters in terms of "topological and metric relationships"[4] will be retained.

There are two steps involved in forming a self-organizing map from a raw input data-set[5], respectively to be 1) **competition** and 2) **cooperation**. When a set of data is feed into the system sequentially with random shuffle, for each input data point, **competition** will take place first and, based on a pre-defined cost function, one of the neurons on the output layer with the minimal cost will be selected as a winner; Following the competition, the **cooperation** will then take place. Based on a neighborhood function, the winner together with it's neighbor neurons will proceed the learning; while, the neurons outside of the winner's neighbor zone will gain no learning. The purpose of the cooperation step is to increase the like-hood that if a similar input pattern present again, the same group of neurons will become the winner with a higher possibility. Iterate with this strategy on the input data-set over a suitable period, without supervising (providing error to the system), the output layer will simultaneously form a map that contains the similar topological structure as the input data.

2.2 Simon and his two methods

In 2006, Simon and his Phd student published two very important papers for this task scheduling problem.

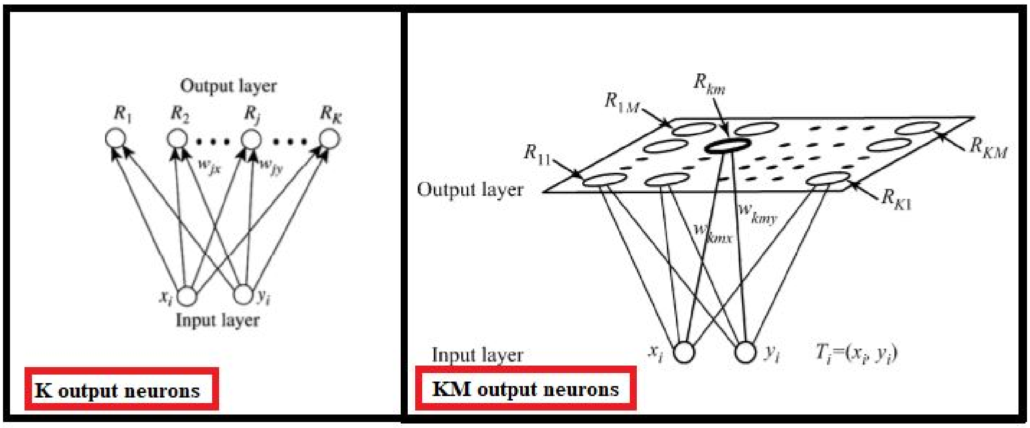

FIGURE 2.1: The illustration of two SOM configuration[1, 2].
The figure on the left has K neurons on the output layer[2]; while, the figure on the right has KM neurons on the output layer[1]

2.2.1 A simple SOM with K output neurons

2.2.2 The impletation, SOM with KM output neurons

Chapter 3

The Proposed Approach to the Problem

In this section, you should briefly outline your work for this problem: what do you plan to do in your project? What results may you expect to have? You don't need to give any details of your approach in your proposal, just list the ideas you plan to use to solve the problems. Of course, if you know some details, you can put them here, e.g., your preliminary results. When you write your interim report, you should provide the details of your proposed approach.

Chapter 4

Discussion

In this section, you may present your preliminary ideas on how your work can be compared with previous works or related works: what's new for your work? What are the differences of your work from existing works? What the advantages and limitations of your approach? How the parameter variation in your model may result the model performance (parameter sensitivity of your model)? How would you improve you work in the future (beyond this course project)? ...

Chapter 5

Summary

In this section, conclude what have in this proposal, and summarize the important points naturally drawn from the proposal.

Bibliography

[1] Anmin Zhu and Simon X Yang. A neural network approach to dynamic task assignment of multirobots. *IEEE transactions on neural networks*, 17(5):1278–1287, 2006.

[2] Anmin Zhu and Simon X Yang. A som-based multi-agent architecture for multirobot systems. *International Journal of Robotics & Automation*, 21(2):91, 2006.

[3] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[4] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325.

[5] Colin Fyfe. *Hebbian learning and negative feedback networks.* Springer Science & Business Media, 2007.