Multi-Armed Bandit for Solving Quadratic Assignment Problem

Reinforcement Learning

Milad Khademi Nori

November 18, 2018

Contents

Introduction to Reinforcement Learning

- Origin
  - "A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 20 miles per hour." Sutton and Barto

Introduction to Reinforcement Learning

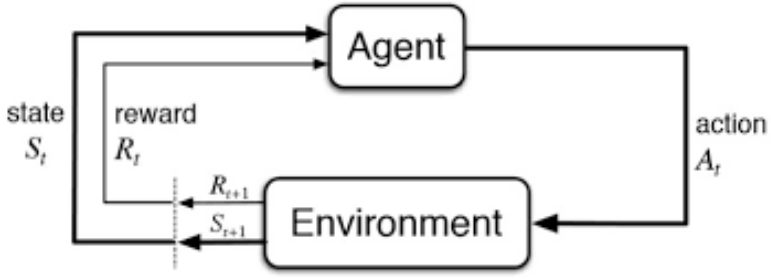
- Goals
  - Agents interact dynamically with its environment, moves from one state to another.
  - Based on the actions taken by the agent, rewards are given.
  - Guidelines for which action to take in each state is called a policy.
  - Try to efficiently find an optimal policy in which rewards are maximized.
- Achievement
  - Google's AlphaGo used deep reinforcement learning in order to defeat world champion Lee Sedol at Go. In Go number of possible games is larger than the number of atoms in the universe and it is much more challenging than chess.

Reinforcement Learning vs Supervised Learning

- Supervised Learning
  - Learning from examples (Dataset) provided by knowledgeable external supervisor.
  - For any state that the agent may be in, the supervisor can supply enough relevant examples of the outcomes which result from similar states so that we may make an accurate prediction.
- Reinforcement Learning
  - No supervisor exists.
  - Agent must learn from experience as it explore the range of possible states.

Introduction to Reinforcement Learning

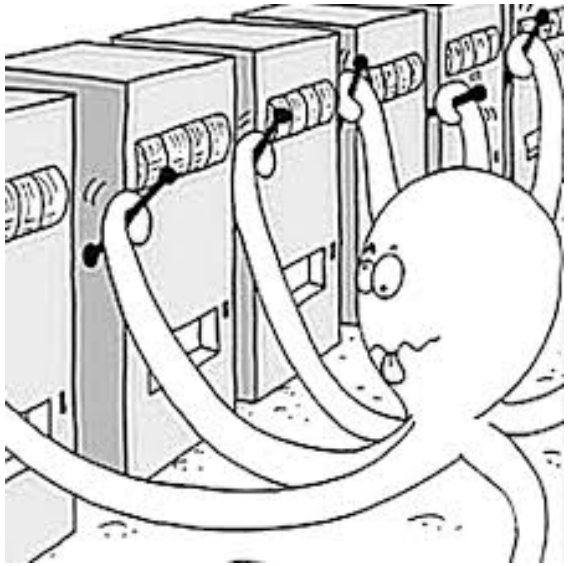
- Reinforcement Learning



- Examples

Agent	Environment	Actions	Rewards	Policy
Chess player	Set of all game configs.	Legal	Winning the game	Optimal strategy
Mouse	Maze	Running & turning	Cheese	Most direct path to cheese

- Exploration & Exploitation
  - In the absence of a supervisor, the agent must explore the environment in order to gain information about rewards, while exploiting its current information to maximize its rewards.
  - Balancing this tradeoff is a common theme
- A Single State Example: Multi-Armed Bandit Problem



The Paper Which I'm Going to Present

Bandit-Inspired Memetic Algorithms for Solving Quadratic Assignment Problems

Francesco Puglierin
Information and Computing Sciences
Utrecht University, The Netherlands
Email: francesco@puglier.in

Mădălina Drugan
Artificial Intelligence Lab
Vrije Universiteit Brussel, Belgium
Email: mdrugan@vub.ac.be

Marco Wiering (*IEEE Member*)
Department of Artificial Intelligence
University of Groningen, The Netherlands
Email: m.a.wiering@rug.nl

***Abstract*—In this paper we propose a novel algorithm called the Bandit-Inspired Memetic Algorithm (BIMA) and we have applied it to solve different large instances of the Quadratic Assignment Problem (QAP). Like other memetic algorithms, BIMA makes use of local search and a population of solutions. The novelty lies in the use of multi-armed bandit algorithms and assignment matrices for generating novel solutions, which will then be brought to a local minimum by local search. We have compared BIMA to multi-start local search (MLS) and iterated local search (ILS) on five QAP instances, and the results show that BIMA significantly outperforms these competitors.**

***Index Terms*—Meta-heuristics, Memetic Algorithms, Combinatorial Optimization, Quadratic Assignment Problem, Multi-armed Bandit Algorithms**

I. Introduction

Many real-world problems in logistics, transport, and manufacturing can be modeled as combinatorial optimization prob-

Several optimization objectives of the problem have been proposed in literature; the following one, proposed by Çela in [9], is probably the most commonly used:

$$cost(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)}, \qquad (1)$$

where n is the size of the problem instance, f is the *flow matrix*, f_{ij} is the directed flow between facility i and facility j, d is the *distance matrix*, and d_{ij} is the directed distance between location i and location j. Finally, π represents a possible permutation over (1,2,...,n) and $\pi(i)$ corresponds to the index of the location to which facility i is assigned. The aim is to minimize the cost function defined by formula (1). The QAP has been proven NP-hard in [10].

Main contributions. This paper describes the novel bandit-inspired memetic algorithm (BIMA), which combines memetic

Multi-Armed Bandit (MAB) Problem

- Multi-Armed Bandit Problem
  - Given N different arms to choose from, each with an unknown reward, what strategy should we use to explore and learn the values of each arm, while exploiting our current knowledge to maximize profit?
  - This is a very common approach for optimizing online marketing campaigns.
  - This can be thought of as a single-state reinforcement learning problem.
- MAB Solvers
  - Epsilon Greedy
  - Upper Convergence Bound (UCB)

Multi-Armed Bandit (MAB) Problem

- Upper Convergence Bound (UCB)

$$Score_j^t = \bar{x}_j^t + \sqrt{\frac{c \times ln \sum_k p_k^t}{p_j^t}} \tag{1}$$

- The first term in the formula, $\bar{x}_j^t$, encodes the expected average reward for arm j according to knowledge available in time-step t.
- Always choosing the arm with the highest expected reward would result in a purely exploitative algorithm, so the formula includes a second term to deal with exploration.
- The variable p_j^t represents the number of times arm j has been pulled at time-step t, making the value of the second term in formula inversely proportional to the arm popularity.

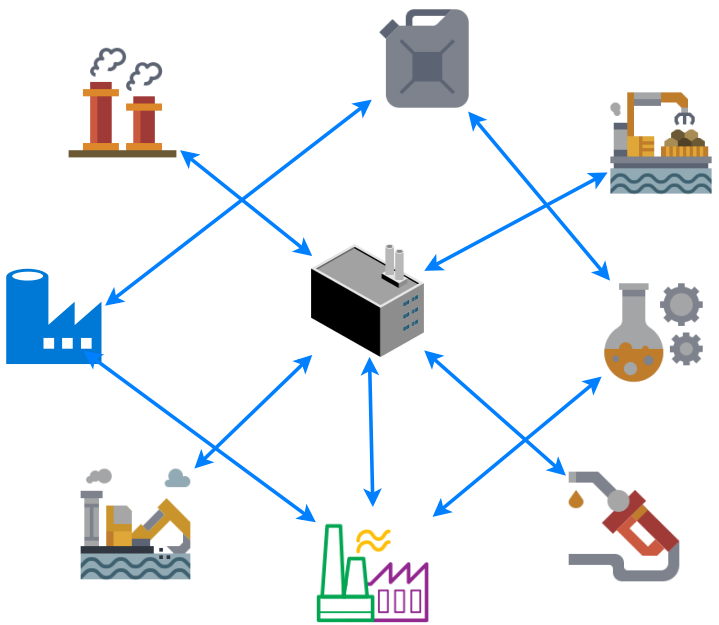
Quadratic Assignment Problem (QAP)

- The quadratic assignment problem (QAP) is a combinatorial optimization problem introduced by Koopmans and Beckmann in 1957 as a formal model for allocating economical facilities.
- There are a number of facilities to assign to the same number of locations in an optimal way.
- A mutual distance is given between locations.
- Also, a mutual flow is given between facilities which quantifies the mutual interaction between facilities.
- The aim is to minimize the cost function.
- The QAP has been proven to be NP-hard ($N!$ solutions).

Quadratic Assignment Problem (QAP)

$$D = \begin{bmatrix} d_{11} & \cdots & d_{1N} \\ d_{21} & \cdots & d_{2N} \\ \vdots & \ddots & \vdots \\ d_{N1} & \cdots & d_{NN} \end{bmatrix}, F = \begin{bmatrix} f_{11} & \cdots & f_{1N} \\ f_{21} & \cdots & f_{2N} \\ \vdots & \ddots & \vdots \\ f_{N1} & \cdots & f_{NN} \end{bmatrix}, \quad d_{11}, f_{11} = d_{22}, f_{22} = \cdots = d_{NN}, f_{NN} = 0 \quad (2)$$

$$\Pi() = [\pi(1) \quad \pi(2) \quad \cdots \quad \pi(N)], \quad 1 \leq \pi(1), \pi(2), \cdots, \pi(N) \leq N, \quad cost(\Pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij} d_{\pi(i)\pi(j)} \quad (3)$$



Development Process

$Population(rnd) \leftarrow \{\Pi_1^{rnd}, \Pi_2^{rnd}, \cdots, \Pi_{ps}^{rnd}\}$
$Populaion(0) \leftarrow LOCAL\_SEARCH(Population(rnd))$
Repeat
  $A^t \leftarrow SELECT\_SUBSET(Population(t))$
  $\Pi_i^{tmp} \leftarrow BUILD\_SOLUTION(A^t, \Pi_i^t)$
  $\Pi_i^{new} \leftarrow LOCAL\_SEARCH(\Pi_i^{tmp})$
  If cost $(\Pi_i^{new}) \leq$ **cost** (Π_i^t) **then**
  $Population(t+1) \leftarrow Population(t) \backslash \{\Pi_i^t\} \cup \{\Pi_i^{new}\}$
  Else
  $Population(t+1) \leftarrow Population(t)$
  End if
  $t \leftarrow t+1$
Until stop_condition

How to BUILD_SOLUTION Based on Solution's Memory

- Local Assignment Cost Matrices & Local Assignment Pull-count Matrices

$$\bar{C}^t(\ell) = \begin{bmatrix} c_{11} & \cdots & c_{1N} \\ c_{21} & \cdots & c_{2N} \\ \vdots & \ddots & \vdots \\ c_{N1} & \cdots & c_{NN} \end{bmatrix}, \quad P^t(\ell) = \begin{bmatrix} p_{11} & \cdots & p_{1N} \\ p_{21} & \cdots & p_{2N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NN} \end{bmatrix} \tag{4}$$

$$\bar{C}^t(\ell) = w\bar{C}^t(\ell) + (1-w)\hat{C}^t(\ell) \tag{5}$$

$$\bar{C}^2(1) = 0.5 \begin{bmatrix} 10 & \times & \times \\ \times & \times & 10 \\ \times & 10 & \times \end{bmatrix} + 0.5 \begin{bmatrix} \times & \times & 5 \\ 5 & \times & \times \\ \times & 5 & \times \end{bmatrix} = \begin{bmatrix} 5 & \times & 2.5 \\ 2.5 & \times & 10 \\ \times & 7.5 & \times \end{bmatrix} \tag{6}$$

$$Cost_{ij}^t = \frac{\bar{c}_{ij}^t(\ell) - min_{(s,t)\in A^t}(\bar{c}_{st}^t(\ell))}{max_{(s,t)\in A^t}(\bar{c}_{st}^t(\ell)) - min_{(s,t)\in A^t}(\bar{c}_{st}^t(\ell))} + \sqrt{\frac{p_{ij}^t(\ell)}{c \times ln \sum_{(s,t)\in A^t} p_{st}^t(\ell)}} \tag{7}$$

Conclusion

1. Introduction to Reinforcement Learning
    - Origin & Goals
    - Reinforcement Learning vs Supervised Learning
    - Exploration vs Exploitation
    - A Single State Example
2. Multi-Armed Bandit (MAB)
    - Problem Statement
        - Epsilon Greedy
        - Upper Convergence Bound
3. Quadratic Assignment Problem (QAP)
    - Problem Statement
        - MAB for Solving QAP

Any Question?