Analyzing the Convergence of Multi-Agent Reinforcement Learning for Budget-Constrained Prize-Collecting Travelling Salesman Problem using ILP

By: Christopher Gonzalez, Andrew Asendorf, Mentor: Dr. Bin Tang
Department of Computer Science, California State University, Dominguez Hills

Abstract

- The budget-constrained prize-collecting travelling salesman problem (BC-TSP) is an extension of the classic traveling salesman problem. An example of BC-TSP is an Uber driver trying to maximize the amount of money (prize) gained from driving, with respect to the driver's time and/or fuel (budget).
- This work seeks to analyze our prize-driven multi-agent reinforcement learning algorithm's (P-MARL) convergence by comparing it to the optimal solution using Integer linear programming (ILP), since previous work doesn't compare to an optimal solution. Although ILP outputs an optimal solution, as the number of points that can possibly be traveled to increases, the number of possible routes increases factorially, increasing execution time.
- The motivation behind determining P-MARL's convergence for BC-TSP is the fact that as the number of possible points that can be travelled to increases, P-MARL runs faster because of its smaller time complexity. This work seeks to get optimal solutions from P-MARL by first comparing it to optimal ILP solutions, then tuning hyper parameters such as learning rate, discount factor, and the number of episodes, to consistently get an optimal solution. Experiments show that for small cases, P-MARL can get optimal solutions. For larger instances, P-MARL can outperform greedy algorithms.

BC-PCTSP

- Given a weighted complete graph with a set of nodes and edges, each edge has a weight, indicating a travel distance or a cost and each node has a weight, indicating the prize available at that node.
- Given any two nodes, the goal of the BC-TSP to find a route between the two nodes such that the prize collected is maximized while the total distance below or equal to the Budget.

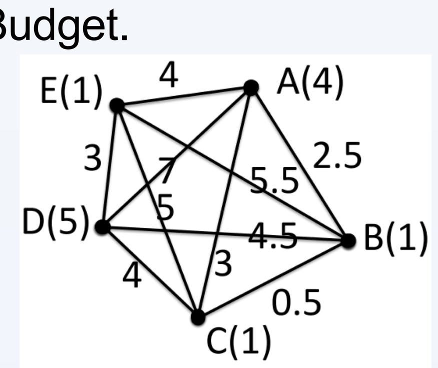


Fig. 1: An example.

- Fig. 1 shows a BC-TSP with a budget of 8. The numbers on the edges are their weights and the numbers in the parentheses are the prizes available at the nodes. Assuming the source and destination nodes are E and C respectively, the optimal walk is E,D,B,C with a total prize of 8 and a total cost of 8.
- BC-TSP is an NP-hard problem
- Previous work used heuristics and a reinforcement learning algorithm to try to solve the BC-TSP [1] but did not show how they compare to optimal solutions or if the reinforcement algorithm converges to an optimal solution.
- This work compares RL algorithms to the optimal solution by optimally solving BC-TSP by formulating it as an integer linear program (**ILP**)

Methodology/ Algorithms

$$\max \sum_{i \in V-\{t\}} \sum_{j \in V-\{s\}} p_i \cdot x_{i,j} \tag{1}$$

s.t.

$$x_{i,j} \in \{0,1\} \quad \forall i,j \in V \tag{2}$$

$$\sum_{j \in V-\{s\}} x_{s,j} = \sum_{i \in V-\{t\}} x_{i,t} = 1, \tag{3}$$

$$\sum_{i \in V-\{t\}} x_{i,k} = \sum_{j \in V-\{s\}} x_{k,j} \le 1, \quad \forall k \in V-\{s,t\} \tag{4}$$

$$\sum_{i \in V-\{t\}} \sum_{j \in V-\{s\}} w_{i,j} \cdot x_{i,j} \le \mathcal{B}, \tag{5}$$

$$2 \le u_i \le |V|, \quad \forall i \in V-\{s\} \tag{6}$$

$$u_i - u_j + 1 \le (|V|-1) \cdot (1 - x_{i,j}), \tag{7}$$

- BC-TSP is solved optimally with **ILP** using the above formulation. The formulation's objective is to maximize the prize collected with respect to several constraints that enforce a single tour from the source to the destination, with a cost lower than the budget. The optimal solution is compared with two reinforcement algorithms and determine their convergence

Input: A graph $G(V,E)$, s, t, and a budget $\mathcal{B}$.
Output: A route R from s to t, C_R, and P_R.
Notations: i: index for episodes; j: index for agents;
U_j: set of nodes agent j not yet visits, initially $V-\{s,t\}$;
R_j: the route taken by agent j, initially empty;
B_j: the currently available budget of agent j, initially $\mathcal{B}$;
l_j: the cost (i.e., the sum of edge weights) of R_j, initially 0;
P_j: the prizes collected on R_j, initially 0;
r_j: the node where agent j is located currently;
s_j: the node where agent j moves to next;
$isDone_j$: agent j has finished in this episode, initially false;
R^m: the global-best route (i.e., with maximum prize) so far;
P^m: the prize of the global-best route;
R: the final route found the MARL, initially empty;
$Q(u,v)$: Q-value of edge (u,v), initially 0;
$r(u,v)$: reward of traversing edge (u,v), initially 0;
α: learning rate, $\alpha = 0.1$;
γ: discount factor, $\gamma = 0.3$;
q_0: trade-off between exploration and exploitation, $q_0 = 0.5$;
δ, β: parameters in action rule; $\delta = 1$ and $\beta = 2$;
W: a constant value of 100;
epi: number of episodes in the MARL, $epi = 30000$;
$R^m = \phi$ (empty set), $P^m = 0$;

for $(1 \le i \le epi)$ // Learning stage
  All the m agents are at node s, $r_j = s, 1 \le j \le m$;
  for $(j = 1; j \le m; j{+}{+})$ // Agent j
    $P_j = 0$, $B_j = \mathcal{B}_j$, $isDone_j = false$;
  end for;
  // At least one agent has not finished in this episode
  while $(\exists\, j, 1 \le j \le m, isDone_j == false)$
    for $(j = 1; j \le m; j{+}{+})$ // Agent j
      if $(isDone_j == false)$ // Agent j has not finished
        if $(U_j \cap \mathcal{F}(r_j, B_j) == \phi)$ // Agent j terminates
          $isDone_j = true$;
          $R_j = R_j \cup \{t\}$; // Agent j goes to t
          $l_j = l_j + w(r_j, t)$, $B_j = B_j - w(r_j, t)$;
          $r_j = t$;
        end if;
        else
          Finds the next node s_j following action rule;
          $R_j = R_j \cup \{s_j\}$;
          $l_j = l_j + w(r_j, s_j)$, $B_j = B_j - w(r_j, s_j)$;
          $P_j = P_j + p_{s_j}$; // Collect prize
          $r_j = s_j$; // Move to node s_j;
          $U_j = U_j - \{s_j\}$;
        end else;
      end if;
    end for;
  end while;
  $j^* = \text{argmax}_{1 \le j \le m} P_j$;
  if $(P_{j^*} > P^m)$ // Found a global-best route
    $P^m = P_{j^*}$, $R^m = R_{j^*}$;
    for (each edge $(u,v) \in R_{j^*}$)
      $r(u,v) = r(u,v) + \frac{i \cdot W}{P_{j^*}}$; // Update reward value
      $Q(u,v) \leftarrow (1-\alpha) \cdot Q(u,v) + \alpha \cdot [r(u,v) + \gamma \cdot \max_b Q(v,b)]$; // Update Q-value
    end for;
  end if;
end for; // End of each episode in learning stage
// Execution stage
$r = s$, $R = \{s\}$, $C_R = 0$, $P_R = 0$, $B = \mathcal{B}$;
while $(r != t)$
  $u = \text{argmax}_b Q(r,b)$;
  $R = R \cup \{u\}, C_R = C_R + w(r,u), P_R = P_R + p_u$, $B = B - w(r,u)$;
  $r = u$;
end while;
RETURN R, C_R, P_R, B.

- The above multi-agent reinforcement learning algorithm (**P-MARL**) augments Q-learning by using multiple agents that are independent and cooperative learners. Fig 2. shows the workflow of **P-MARL**

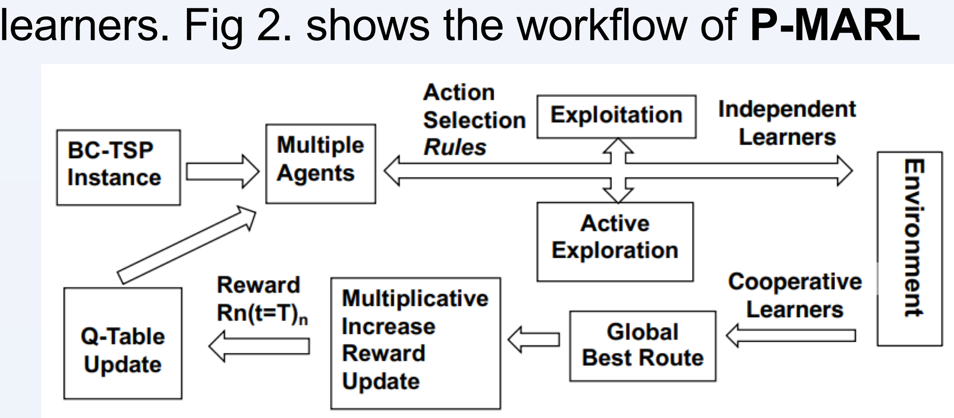
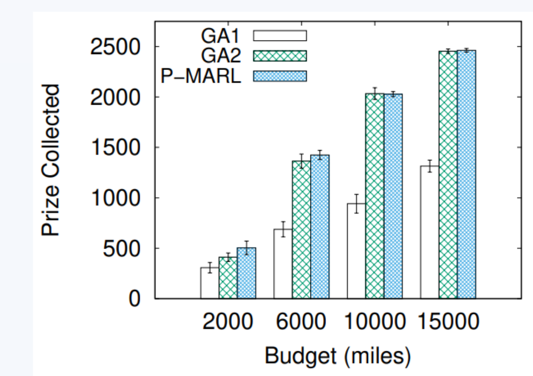


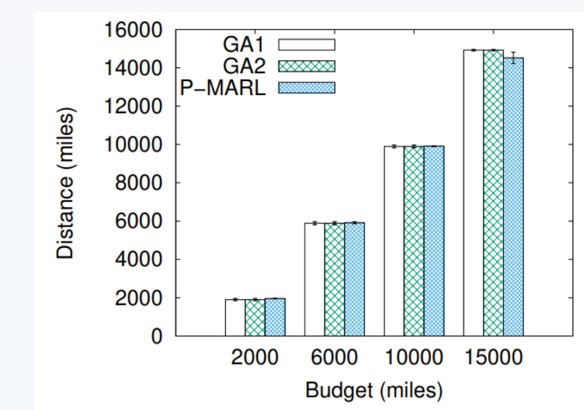
Fig. 2: Workflow of the P-MARL.

- Lastly, **P-MARL** is compared to two greedy algorithms **GA1** and **GA2**. For **GA1**, starting from the first city, the city with the highest prize is visited until there are no unvisited cities or there are no more budget feasible cities to travel to. **GA2** is like **GA1**, except the city with the highest prize to distance ratio is visited instead.

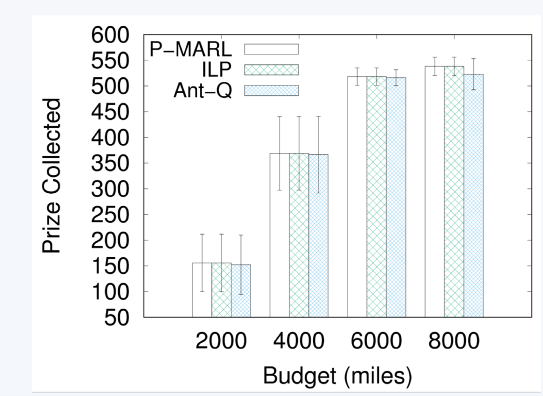
Experiment and Results

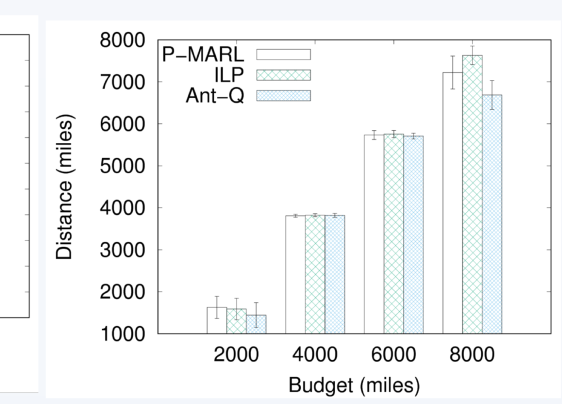
- For the **P-MARL** and **Ant-Q** algorithms, the learning rate used was 0.1, the discount factor was 0.35, and the number of learning episodes was 4000. To evaluate the performance of **P-MARL**, first it is compared to **GA1** and **GA2** on a traveling salesman tours of 48 state capital cities with varying budgets.



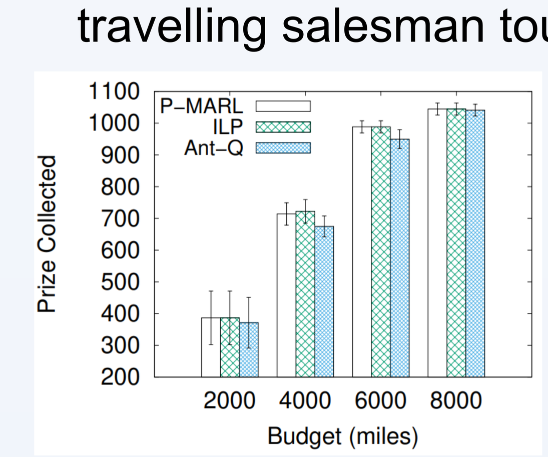


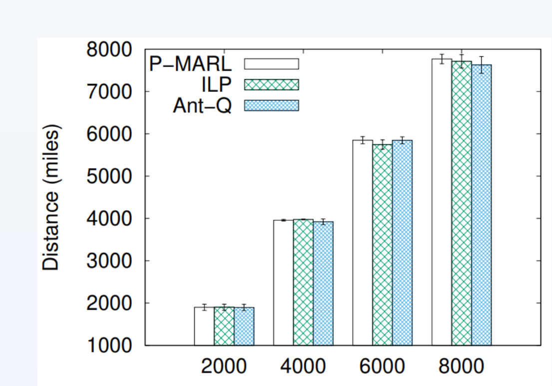
- Then, **P-MARL** is compared to the optimal ILP solution (**ILP**), and Ant-Q MARL [2] (**Ant-Q**) on travelling salesman tours of 10 cities with varying budgets.



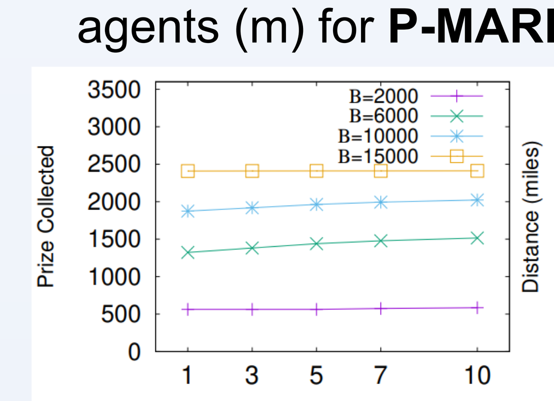


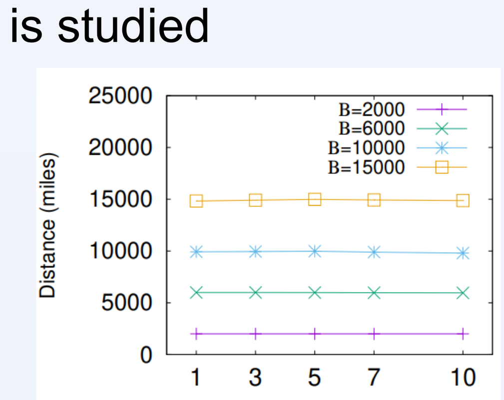
- These Algorithms are also compared on travelling salesman tours of 20 cities.

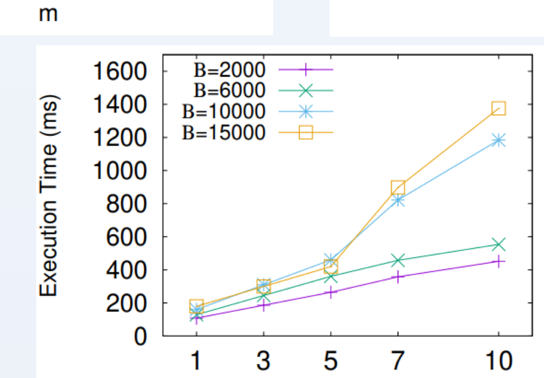




- Finally, the impact of changing the number of agents (m) for **P-MARL** is studied







Conclusions and Future Work

- For a tour of 48 possible cities, It is observed that **P-MARL** performs best on smaller budget instances. For higher budgets, **GA2** performs closer to **P-MARL** even beating **P-MARL** with a budget of 10,000.
- **P-MAR**L performs better with less possible cities to travel to. Performing optimally with 10 cities, slightly worse with 20 cities, and performing close to **GA1** and **GA2** with 48 cities
- Changing the number of agents does not affect the traveled distance of the salesman, as the focus of **P-MARL** is to maximize the prize collected.
- Execution time has a big jump from 5 agents to 7 agents.
- Future work will investigate whether 5 agents is the right number of agents, and if there is a theoretical optimal number of agents for **P-MARL** with respect to execution time of the learning process.

References

[1] J. Ruiz, C. Gonzalez, Y. Chen, and B. Tang. Prize-collecting traveling salesman problem: a reinforcement learning approach. In Proc. of IEEE ICC, 2023

[2]L. Gambardella and M. Dorigo. Ant-q: A reinforcement learning approach to the traveling salesman problem. In ICML, 1995.

Acknowledgment

This work was supported in part by NSF Grant: CISE-MSI: 2240517