ReinforceNS: Reinforcement Learning-based Multi-start Neighborhood Search for Solving the Traveling Thief Problem

Tao Wu[1], **Huachao Cui**[2], **Tao Guan**[1], **Yuesong Wang**[1] and **Yan Jin**[1*]
[1] Huazhong University of Science and Technology
[2] Nanjing Research Institute of Electronic Engineering
m202273706@hust.edu.cn, jinyan@mail.hust.edu.cn

Abstract

The Traveling Thief Problem (TTP) is a challenging combinatorial optimization problem with broad practical applications. TTP combines two NP-hard problems: the Traveling Salesman Problem (TSP) and Knapsack Problem (KP). While a number of machine learning and deep learning based algorithms have been developed for TSP and KP, there is limited research dedicated to TTP. In this paper, we present the first reinforcement learning based multi-start neighborhood search algorithm, denoted by ReinforceNS, for solving TTP. To accelerate the search, we employ a pre-processing procedure for neighborhood reduction. A TSP routing and an iterated greedy packing are independently utilized to construct a high-quality initial solution, further improved by a reinforcement learning based neighborhood search. Additionally, a post-optimization procedure is devised for continued solution improvement. We conduct extensive experiments on 60 commonly used benchmark instances with 76 to 33810 cities in the literature. The experimental results demonstrate that our proposed ReinforceNS algorithm outperforms three state-of-the-art algorithms in terms of solution quality with the same time limit. In particular, ReinforceNS achieves 12 new results for 18 instances publicly reported in a recent TTP competition. We also perform an additional experiment to validate the effectiveness of the reinforcement learning strategy.

1 Introduction

The Travelling Thief Problem (TTP) is a challenging combinatorial optimization problem as it combines two NP-hard problems: the Traveling Salesman Problem (TSP) [Flood, 1956] and Knapsack Problem (KP) [Pisinger and Toth, 1998]. And it holds practical importance with varies applications in real-life scenarios, such as water tank transportation [Bonyadi *et al.*, 2013], printed circuit board design [Reinelt, 1991], airport ground movement [Weiszer *et al.*, 2014], wavelength assignment for mesh networks [Hu and Leida, 2004] and the routing problem with loading constraints [Iori and Martello, 2010].

Given its theoretical challenge and practical significance, TTP has attracted increasing research effort in the last decade [Wagner *et al.*, 2018; El Yafrani and Ahiod, 2018; Namazi *et al.*, 2023]. Existing research on TTP includes exact algorithms, heuristics and hyper-heuristics. Due to its intrinsic intractability, the research lies in the development of effective heuristics to find near-optimal solutions within a reasonable time limit. Compared to the close relative problems TSP and KP that have been intensively studied [Lienland and Zeng, 2015; Osaba *et al.*, 2020; Cacchiani *et al.*, 2022], TTP has received limited attention. Moreover, the TSP and KP algorithms integrating machine learning techniques have demonstrated remarkable performances [Afshar *et al.*, 2020; Zheng *et al.*, 2021; Pierotti *et al.*, 2021; Xin *et al.*, 2021; Tu *et al.*, 2023; Pan *et al.*, 2023], while few efforts have been dedicated to developing machine learning-based TTP algorithms [Namazi *et al.*, 2023].

To the best of our knowledge, no study has been reported in the literature investigating the interest of the reinforcement learning technique for solving TTP. In this work, we fill the gap by introducing the first reinforcement learning-based multi-start neighborhood search algorithm ReinforceNS for TTP. ReinforceNS integrates pre-processing, initialization, reinforcement learning based neighborhood search and post-optimization. To ensure effective exploration of the search space, ReinforceNS employs a multi-start mechanism to escape local optima traps and the dedicated initialization to regenerate new high-quality initial solutions. The main contributions of the paper are summarized as follows.

- The proposed ReinforceNS algorithm integrates a pre-processing procedure and a reinforcement learning-based neighborhood search procedure to ensure an effective and efficient examination of promising neighboring solutions.
- ReinforceNS applies a post-optimization procedure to further improve the local optimal solutions. This is ensured jointly by an item flipping phase and a route refining phase.
- ReinforceNS is extensively tested on the commonly used 60 TTP benchmark instances. Experimental results show the proposed algorithm outperforms three state-of-

*Yan Jin is the corresponding author, jinyan@mail.hust.edu.cn.

the-art algorithms within the same time limit. In particular, it finds 12 improved results for 18 instances publicly reported in a recent TTP competition.

2 Related Work

The TTP problem was first introduced in 2013 [Bonyadi *et al.*, 2013] and has received increasing attention in recent years. The existing approaches for solving TTP can be mainly classified into three families: exact algorithms, heuristics and hyper-heuristics. In this section, we briefly review exact algorithms and hyper-heuristics, and then focus on heuristics.

Exact algorithms utilizing dynamic programming or mixed-integer programming [Polyakovskiy and Neumann, 2017; Wu *et al.*, 2017] can find optimal solutions with theoretical guarantees, but face challenges when dealing with large instances within reasonable time due to the inherent exponential complexity.

In contrast, hyper-heuristic and heuristic algorithms can find near-optimal solutions for practical applications, but they lack optimality guarantees. The hyper-heuristic algorithms often utilize genetic programming to generate or select low-level heuristics for exploring routes or packing plans, such as [Mei *et al.*, 2015], [Martins *et al.*, 2017; El Yafrani and Ahiod, 2018], [Martins *et al.*, 2017], [El Yafrani and Ahiod, 2018], [Ali and Mohamedkhair, 2021].

The heuristics are widely studied for solving TTP, which can be classified into the following three families [Namazi *et al.*, 2023].

Constructive paradigm. The Chained Lin-Kernighan Heuristic (CLKH) [Applegate *et al.*, 2003] is used to generate a route, and greedy packing strategies are designed while keeping this route unchanged, such as [Polyakovskiy *et al.*, 2014; Bonyadi *et al.*, 2014; Faulkner *et al.*, 2015]. S5 applies them within its restart-based framework [Faulkner *et al.*, 2015]. The constructive paradigm often serves as an initialization procedure for other TTP algorithms.

Cooperative paradigm. After the generation of an initial TTP solution through the constructive paradigm, two separate search procedures for TSP and KP are iteratively applied to improve the route and packing plans. Some notable algorithms in this family include the Cooperative Solver (CoSolver) [Bonyadi *et al.*, 2014] and its improved versions: CoSolver with 2-opt and Simulated Annealing (CS2SA) [El Yafrani and Ahiod, 2016], CS2SA with offline instance-based parameter tuning (CS2SA*) [El Yafrani and Ahiod, 2018], CoSolver with Reverse-Weighted Selection (RWS) [Zhang *et al.*, 2021], as well as a state-of-the-art heuristic Cooperation Coordination (CoCo) proposed most recently [Namazi *et al.*, 2023].

Full-encoding paradigm. Full-encoding paradigm addresses TTP without segregating it into TSP and KP. Many algorithms adopt a hybrid evolutionary framework [El Yafrani and Ahiod, 2016; El Yafrani and Ahiod, 2017; Mei *et al.*, 2014; Wuijts and Thierens, 2019], such as Memetic Algorithm with Two-Stage Local Search (MATLS) [Mei *et al.*, 2014], Memetic algorithm with Edge Assembly and 2-Point crossover operator (MEA2P) [Wuijts and Thierens, 2019]. Additionally, a swarm intelligence algorithm MMAS [Wagner, 2016] based on the Max-Min Ant System [Stützle and Hoos, 2000] has been proposed for solving TTP.

Note that three best-performing algorithms, i.e., S5, CoCo and MATLS, from each family of heuristics are used for comparisons in the experiments.

3 Problem Description

The TTP problem is defined as follows. Let $G = (N, E)$ be a complete graph, where $N = \{1, \dots, n\}$ is the vertex set with 1 being the starting-ending city (depot) x_1 and other vertices representing $n-1$ other cities and E is the set of edges. Let $D = (d_{i,j})$ be a non-negative distance matrix associated with E. A set of items $M = \{1, \dots, m\}$ is distributed among these cities except city x_1. Suppose that each city x_i ($i \in \{2, \dots, n\}$) contains m_i distinct items, denoted as $M_i = \{i_1, \dots, i_{m_i}\}$, $M = \cup_{i=1}^{n} M_i$ and $M_i \cap M_j = \emptyset, i \neq j$. Each item i_k ($i_k \in \{1, \dots, m\}$) positioned in the city x_i has a profit value p_{i_k} and weight w_{i_k}. The thief starts from city x_1, visits each of the cities exactly once, and returns back to the starting city x_1. During the tour, the thief can pick up any item into his knapsack in any city until the total weight of items does not exceed the maximum weight W of the knapsack. A renting rate R is to be paid per each time unit being on the tour.

The v_{min} and v_{max} are the minimum and maximum speeds that the thief can move respectively. Suppose that v_{x_i} represents the speed of the thief in city x_i. Eq. (1) gives the linear relationship between the thief's speed and the knapsack's weight.

$$v_{x_i} = v_{\text{max}} - C \times w_{x_i} \tag{1}$$

where $C = \frac{v_{\text{max}} - v_{\text{min}}}{W}$ is a constant value, w_{x_i} indicates the knapsack weight of the thief in city x_i, as shown in Eq. (2).

$$w_{x_i} = \sum_{j=1}^{i} \sum_{k=1}^{m_j} w_{j_k} \times z_{j_k} \quad \text{s.t.} \quad z_{j_k} \in \{0, 1\} \tag{2}$$

Let $\Pi = (x_1, \dots, x_n)$ represent a route plan and $P = (z_1, \dots, z_m)$ be a binary vector representing a packing plan, where z_j equals to 1 when item j is picked up, and 0 otherwise. The objective of TTP is to find a route plan Π and a packing plan P of the maximal profit value as Eq. (3).

$$\begin{aligned} f(\Pi, P) &= g(P) - R \cdot \psi(\Pi, P) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{i_k} z_{i_k} - R \cdot \Big(\frac{d_{x_n, x_1}}{v_{\text{max}} - C \cdot w_{x_n}} + \\ &+ \sum_{i=1}^{n-1} \frac{d_{x_i, x_{i+1}}}{v_{\text{max}} - C \cdot w_{x_i}} \Big) \end{aligned} \tag{3}$$

where $g(P)$ represents the total profit values of all items packed in the knapsack, $R \cdot \psi(\Pi, P)$ represents the amount that the thief pays for the knapsack's rent.

One easily notices that TTP can be reduced to TSP by assigning zero to the profit values of all items. Similarly, TTP can be considered as KP by setting the renting value to zero. These observations also confirm the challenge of solving the TTP problem.

Algorithm 1 The proposed ReinforceNS algorithm

Input: A TTP instance
Output: The best solution (Π^*, P^*)

Initialize $\Pi^*, P^* \leftarrow \emptyset, f^* \leftarrow -\infty$;
$t \leftarrow 0$;
Candidate edge sets $\leftarrow$ A pre-processing procedure;
while $t < T_{max}$ **do**
  An initial route $\Pi \leftarrow$ TSP solver CLKH;
  An initial packing $P \leftarrow$ An iterated greedy packing procedure IGP(Π);
  $(\Pi', P') \leftarrow$ RLNS(Π, P); /* The reinforcement learning based neighborhood search procedure */
  $P'' \leftarrow$ An Item Flipping Phase IFP(Π', P'); /* Further improve P' */
  $\Pi'' \leftarrow$ A Routing Refining Phase RRP(Π', P''); /* Further improve Π' */
  if $f(\Pi'', P'') > f^*$ **then**
    $f^* \leftarrow f(\Pi'', P'')$;
    $\Pi^* \leftarrow \Pi''$;
    $P^* \leftarrow P''$;
  end if
end while
return (Π^*, P^*)

4 The Proposed ReinforceNS Algorithm

The proposed ReinforceNS algorithm is a multi-start iterated neighborhood search heuristic reinforced by a reinforcement learning mechanism. After pre-processing, it explores the search space by repeating a dedicated solution construction, local optimization and post-optimization. The local optimization and post-optimization procedures apply neighborhood search by transitioning from the current solution to one of its neighboring solutions. Each iteration involves replacing the current solution with the best neighboring solution that offers an immediate improvement over the current solution.

Algorithm 1 provides the general scheme of ReinforceNS that relies on a multi-start iterated local search framework. Specifically, after the initialization of the optimal solution (Π^*, P^*) and its corresponding objective value f^*, ReinforceNS applies a pre-processing procedure to reduce the candidate edge sets, and then performs a series of iterations until reaching the maximum time limit T_{max}. At each iteration, a solution Π is generated by the TSP solver CLKH [Applegate *et al.*, 2003]. Subsequently, an iterated greedy packing procedure (IGP) is applied to Π to produce a packing plan P, thus initializing a high-quality solution (Π, P) for TTP. This solution is improved through the reinforcement learning based neighborhood search procedure. To help the search escape from the local optima, a post-optimization procedure including an item flipping phase and a route refining phase is employed for further improvement. The best-found solution (Π^*, P^*) is updated and finally returned as the output at the end of the ReinforceNS algorithm.

4.1 Pre-Processing

Before starting the search, we pre-process the input TTP instance G using a neighborhood reduction procedure. This is achieved by a subgradient optimization approach on the 1-tree and α-nearness adopted from the well-known TSP algorithm Lin-Kernighan-Helsgaun (LKH) [Helsgaun, 2000].

Given a complete graph G, the neighborhood $\mathcal{N}(x_i)$ of city x_i is defined as the set of adjacent cities x_j connected to x_i by an edge (x_i, x_j). Initially, each vertex has $n-1$ neighbors.

A 1-tree of G is a spanning tree for the graph of vertex set $V \backslash x_i$ combined with two edges in E connected to vertex x_i, an arbitrary special vertex in V. A minimum 1-tree T is the 1-tree with minimum length. The α-nearness of an edge $(x_i, x_j) \in E$ is defined as $\alpha(i,j) = L(T^+(x_i, x_j)) - L(T)$, where $L(T)$ is the length of T and $L(T^+(x_i, x_j))$ is the length of minimum 1-tree that includes the edge (x_i, x_j).

To reduce the neighborhoods of each vertex, we use a subgradient optimization procedure that gradually adjusts the vertex degree towards 2 for a minimum 1-tree by introducing a penalty to each vertex. In our experiment, the size of $\mathcal{N}(x_i)$ is set to 15. The ReinforceNS algorithm utilizes these reduced neighborhoods for the subsequent search.

4.2 Initialization

To ensure efficient search, the quality of initial solutions is crucial for guiding the search towards promising regions. We apply TSP and KP approaches separately to generate an initial TTP solution (Π, P) with high-quality.

We employ Chained Lin-Kernighan Heuristic [Applegate *et al.*, 2003] (CLKH) to generate a high-quality route Π. It is noteworthy that CLKH is one of the most widely used algorithms for solving TSP.

Based on the route Π, we design a dedicated Iterated Greedy Packing (IGP) procedure to attain a good packing plan P for constructing an initial solution (Π, P) for TTP. We adopt the scoring function as Eq. (4) [Faulkner *et al.*, 2015] to evaluate item k picked up from city x_i.

$$s_{i_k} = \frac{p_{i_k}}{w_{i_k} \times \xi_i} \tag{4}$$

where $\xi_i = d_{x_n, x_1} + \sum_{j=i}^{n-1} d_{x_j, x_{j+1}}$ represents the distance from city x_i to the ending city (depot) of the route. Notice that there is a trade-off between weight and profit value, we also introduce exponents to Eq. (5) and the function becomes as the following Eq. (5).

$$s_{i_k} = \frac{p_{i_k}^{\kappa}}{w_{i_k}^{\beta} \times \xi_i^{\gamma}} \tag{5}$$

Different from [Faulkner *et al.*, 2015], we use three parameters κ, β and γ to adapt the impact of profit value, weight and distance respectively.

The IGP procedure is composed of four steps. First, we randomly sample the values of κ, β and γ from a uniform distribution between 0 and 1. Second, iteratively choose an item with the highest score, calculated by Eq. (5), and pack it into the knapsack. When the total weight of items does not exceed the maximum weight W of the knapsack multiplied by a randomly selected factor δ between 0.5 and 1, a feasible packing P_0 is obtained. Third, repeat the above two steps for $\frac{10000}{\log(n \cdot m)}$ times to determine a best-found packing P_0'. Finally, starting with the parameters κ, β, γ and δ that attains P_0', we further explore the search for optimizing the packing by adjusting κ and β, such as $(\kappa - \theta, \beta - \theta)$, $(\kappa + \theta, \beta + \theta)$, $(\kappa - \theta, \beta + \theta)$, and $(\kappa + \theta, \beta - \theta)$, where θ is a parameter set to 5 and will progressively decreases by 0.75 times during the search. This process repeats 20 times, ultimately yielding a high-quality initial packing solution P. Consequently, an initial solution (Π, P) is effectively constructed.

4.3 Reinforcement Learning-Based Neighborhood Search

Neighborhood search is a powerful local optimization approach, we apply a reinforcement learning mechanism to speed up neighborhood examination, and propose a Reinforcement Learning-based Neighborhood Search (RLNS) procedure to discover better solutions.

Recall that $L(T)$ is the length of the minimum 1-tree, and the α-nearness $\alpha(i,j)$ of an edge (x_i, x_j) is the extra length of the $L(T)$ to include this edge, we define the Q-value for city x_i and city x_j as shown in Eq. (6), where the parameter τ is set to 0.5 and b is set to $1e^{-6}$.

$$Q(i,j) = \frac{L(T)}{\tau \cdot \alpha(i,j) + (1-\tau) \cdot d_{i,j} + b} \tag{6}$$

The pseudocode of our RLNS is shown in Algorithm 2. After initializing the Q-value for each edge using Eq. (6), RLNS performs a series of iterations, with the maximum number of iterations $Iter_{max}$ set to 1000. At each iteration, RLNS employs the $\epsilon - greedy$ strategy for each city x_i (excluding the depot) to select a neighboring city x_j for 2-opt exchange. More precisely, it selects the best neighboring city x_j with the highest Q-value from the neighborhoods $\mathcal{N}(x_i)$ with a probability of $1 - \epsilon$, and randomly selects x_j with a probability of $\epsilon = 0.2$. Note that, the items of the sub-route from x_i to x_j should be removed from the knapsack of P, since the sub-route between x_i and x_j is changed. For the cities located along the new sub-route from x_j to x_i, RLNS adds an item with the highest score (Eq. (5)) among the items from these cities into the knapsack and repeats until the total weights exceeds the maximum weight of knapsack W multiplied by δ. The best local optimal solution is updated once a better objective value is found. Afterwards, the Q-value of (x_i, x_j) is updated using Eq. (7). Here, the parameter λ and ϕ are set to 0.1 and 0.9 respectively, and the state s_t, action a_t, and reward $r(s_t, a_t)$ are defined as follows.

$$\begin{aligned} Q(s_t, a_t) = &(1-\lambda) \cdot Q(s_t, a_t) + \\ &+ \lambda \cdot \left[r(s_t, a_t) + \phi \max_{a'} Q(s_{t+1}, a') \right] \end{aligned} \tag{7}$$

- State s_t: The state at the current iteration t is a city x_i that is going to select a neighboring city for applying the 2-opt exchange operator.
- Action a_t: Select a neighboring city x_j of city x_i at the current iteration t.
- Transition: The next state after performing the action is the next city that needs to select a neighboring city for the 2-opt exchange operator.

Algorithm 2 The pseudocode of the RLNS procedure

Input: The initial solution (Π, P), the best objective value f'
Output: The local optimal solution (Π', P')

Initialize Q-value of each edge by Eq. (6);
for $iter \leftarrow 0$ To $Iter_{max}$ **do**
  Initialize the local objective value $f_{local} = -\infty$;
  for $x_i \leftarrow 2$ To x_n **do**
    $x_j = \epsilon - greedy(\mathcal{N}(x_i))$ /* Select x_j of the highest Q-value with a probability $1 - \epsilon$ and select x_j at random with a probability ϵ */
    Remove items of P from the sub-route that begins at city x_i and ends at city x_j from the knapsack;
    $\overline{\Pi} \leftarrow$ Apply 2-opt exchange on Π at city x_i and x_j;
    $\overline{P} \leftarrow$ Iteratively add the items with the highest score as Eq. (5) of the sub-route from city x_j to city x_i into the knapsack without exceeding $W \times \delta$;
    if $f(\overline{\Pi}, \overline{P}) \geq f_{local}$ **then**
      $f_{local} \leftarrow f(\overline{\Pi}, \overline{P})$, $\Pi_{local} \leftarrow \overline{\Pi}$, $P_{local} \leftarrow \overline{P}$
    end if
    Update Q-value of (x_i, x_j) according to Eq. (7)
  end for
  if $f_{local} > f'$ **then**
    $\Pi' \leftarrow \Pi_{local}$, $P' \leftarrow P_{local}$, $f' \leftarrow f_{local}$
  end if
  $\Pi \leftarrow \Pi_{local}$, $P \leftarrow P_{local}$
end for
return local optimal solution (Π', P')

- Reward $r(s_t, a_t)$: Suppose that the current solution (Π_0, P_0), after performing the 2-opt exchange operator at city x_i and x_j, the neighboring solution is (Π_1, P_1). The difference between the objective values of the neighboring solution and the current solution is denoted as $r(s_t, a_t) = f(\Pi_1, P_1) - f(\Pi_0, P_0)$.

During $Iter_{max}$ consecutive iterations, the best-found solution (Π', P') is updated if an improved solution has been found, and the best-found solution (Π', P') is returned as the final output of RLNS.

4.4 The Post-Optimization Procedure

In addition to the RLNS procedure described above, the proposed ReinforceNS algorithm develops a post-optimization procedure to further improve the quality of the best-found solution (Π', P') obtained by RLNS. This post-optimization is ensured jointly by an item flipping phase and a route refining phase.

The item flipping phase is devised to optimize the packing solution P' while keeping the route solution Π' unchanged. The flipping operator for an item involves removing it from the knapsack if it is already included or including it if not. During each iteration, the item flipping phase pseudo-performs the flipping operator for each item i. If this operation leads to a new objective value, the item i is flipped using the item flipping operator. If no improvement is achieved, the item i remains unchanged. This process repeats until no new objective value is discovered during an iteration.

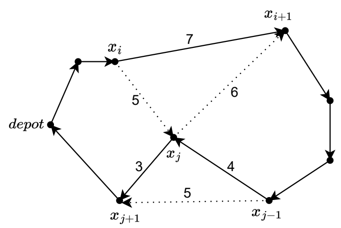

Figure 1: Illustration of adjusting the position of a city in the route

From the problem definition of TTP, one easily notices that as the thief approaches the ending city (depot), the weight of the knapsack increases, leading to a decrease in speed. We introduce a route refining phase to try to move the cities with heavy yet valuable items included in the knapsack to the last positions of the route, for further optimization of the TTP solution.

Figure 1 provides an illustrated example of efficiently calculating the time cost $\psi(\Pi, P)$ in Eq. (3) when adjusting the position of a city in the route. Suppose that the route solution is Π and the packing solution is P, the lengths of edges (x_i, x_{i+1}), (x_{j-1}, x_j), and (x_j, x_{j+1}) are 7, 4, and 3 respectively, and the speed of the thief in city x_i is 5 and in city x_{j-1} is 2. City x_j is moved to a position between cities x_i and x_{i+1} of the route, resulting in a new path Π'. The lengths of the new edges (x_i, x_j), (x_j, x_{i+1}) and (x_j, x_{j+1}) are 5, 6, and 5 respectively. As city x_j has no picked-up items, the speed of the thief remains unchanged. Using Eq. (8), the time increment $\triangle\psi$ is calculated as $\frac{5+6}{5} + \frac{5}{2} - (\frac{7}{5} + \frac{3+4}{2}) = -0.2$, which can be efficiently computed in $O(1)$ time complexity. One observes that adjusting the position of a city in the route makes the new path Π' longer (16 vs. 14), but reduces the time cost (-0.2), resulting in a better route solution.

$$\begin{aligned}\Delta\psi &= \psi(\Pi', P) - \psi(\Pi, P) \\ &= \frac{d_{x_i,x_j} + d_{x_j,x_{i+1}}}{v_{x_i}} + \frac{d_{x_{j-1},x_{j+1}}}{v_{x_{j-1}}} \\ &\quad - \left(\frac{d_{x_i,x_{i+1}}}{v_{x_i}} + \frac{d_{x_{j-1},x_j} + d_{x_j,x_{j+1}}}{v_{x_{j-1}}}\right) \qquad (8)\end{aligned}$$

To clarify the route refining phase, a path Π_x is defined as a route that starts from the depot and ends at city x. For each city x_i without items included in the knapsack, we pseudo-insert it into each edge of Π_{x_i} and efficiently calculate the time cost using Eq. (8). If several improved objective values are discovered, city x_i will be inserted into the edge with the best objective value. If no improved objective values are found for any of these cities, the route refining phase stops.

5 Experiments

To evaluate the effectiveness of the proposed ReinforceNS algorithm, we present computational results in comparison with the best publicly reported results from the TTP competition at the Genetic and Evolutionary Computation Conference (GECCO) 2023 on three sets of 18 instances. Additionally, we compare the performance of ReinforceNS with that of the best-performing algorithms in the literature using three sets of 60 widely used benchmark instances.

ReinforceNS was implemented in C++ and compiled using GNU g++ 9.4.0 with '-O3' optimization flag. The experiments were conducted on the AutoDL server with an Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50 GHz and a memory limit of 40 GB. The maximum runtime limit T_{max} was set to 600 seconds, which is a common stopping condition for evaluating TTP algorithms. Given the stochastic nature of ReinforceNS, each instance was executed 10 times. Note that the source code of the reference algorithm CoCo [Namazi *et al.*, 2023] is publicly available, we re-ran CoCo on our platform to ensure a fair comparison.

Benchmark instances. The TTP benchmark library[1], proposed by Polyakovskiy et al. [2014], comprises a total of 9720 instances with city counts ranging from 76 to 85900. In our experiments, we used a subset of 60 benchmark instances selected from this library, ranging from 76 to 33810 cities. These instances are widely used in previous studies and are categorized into three sets: CatA, CatB and CatC, each containing 10 specific instance files. We use suffixes to denote the specific selected instance files.

- **CatA:** The maximum weight of knapsack is relative small. Each city has only one item. The weights and profits of the items are bounded and strongly correlated.
- **CatB:** The maximum weight of knapsack is moderate. Each city has five items. The weights and profits of the items are uncorrelated, and the weights of all items are similar.
- **CatC:** The maximum weight of knapsack is high. Each city has ten items. The weights and profits of the items are uncorrelated.

Comparison with the best results from TTP competition. We compare our ReinforceNS algorithm with the results of the most recent TTP competition organized by Markus Wagner et al. at the GECCO conference in 2023. Table 1 provides the comparative results for a total of 18 instances with the same time limit. These 18 instances are also classified into three sets: CatA, CatB and CatC. They are randomly selected from 10 instance files of each category, indicated by the suffix "-r". The column "Competition*" presents the best average results among 5 top-performing algorithms publicly reported on the website of GECCO 2023. The next column "ReinforceNS" reports our best average results over 10 runs. Note that we truncate the decimal of all results for brevity. The better results between Competition* and our ReinforceNS are indicated in bold. One easily observes that our proposed algorithm outperforms the best results publicly reported on the website by 12 out of 18 instances. Upon closer examination of ReinforceNS' performance, it is observed that it achieves more new results as the maximum weight of the knapsack and the items in each city increase.

Comparison with three best-performing TTP algorithms. Then we compare ReinforceNS with three top-performing

[1]The TTP instances are available at https://cs.adelaide.edu.au/~optlog/CEC2014COMP_InstancesNew/

Instance	CatA		CatB		CatC	
	Competition*	ReinforceNS	Competition*	ReinforceNS	Competition*	ReinforceNS
lin105-r	9676	**9682**	**24192**	24065	26578	**26595**
a280-r	**19585**	18764	116814	**117616**	429258	**435919**
pcb3038-r	410490	**422658**	1104937	**1139976**	5980458	**6271190**
fnl4461-r	**270073**	268202	1647439	**1696880**	6582360	**6716312**
d18512-r	2383189	**2488247**	5064611	**5230325**	24482316	**25110240**
pla33810-r	**1952100**	1845009	**16114318**	15395460	**58945556**	57201510

Table 1: Comparison with the best results from TTP competition at GECCO2023 within the same time limit

Instance	Cat A				CatB				CatC			
	MATLS	S5	CoCo	ReinforceNS	MATLS	S5	CoCo	ReinforceNS	MATLS	S5	CoCo	ReinforceNS
eil76-1	3798	3741	4108	**4492**	11446	10615	11756	**12815**	34627	35947	36531	**41627**
kroA100-1	4680	4278	**4747**	4723	13546	12054	**14137**	13425	40773	40578	41120	**41284**
ch130-1	8759	9250	9563	**9586**	23024	21746	23200	**23590**	72214	71351	**75871**	75739
u159-1	8358	8637	8763	**9017**	25005	24612	25817	**25830**	77503	76609	**82358**	79855
a280-1	17677	18402	18579	**19034**	38189	38981	38991	**39540**	136745	140724	141642	**143500**
u574-1	25188	26909	27290	**27592**	99943	103471	103840	**104159**	338644	336282	341736	**342746**
u724-1	49321	50346	51030	**51531**	113943	116460	118111	**118434**	372407	386856	**396398**	391756
dsj1000-1	144118	142726	144219	**144415**	147982	148794	153050	**157509**	485557	503440	517894	**526704**
rl1304-1	76674	79848	81104	**81276**	218940	218950	**221487**	220789	759038	759791	**784959**	772945
fl1577-1	86615	90300	92192	**106730**	244808	251442	252795	**261846**	847734	884372	895948	**924266**
d2103-1	114140	120674	121769	**124269**	304993	311941	324502	**325730**	1124960	1136320	1173330	**1178710**
pcb3038-1	146943	160196	161672	**164526**	502804	518939	519845	**525384**	1576420	1619324	1637390	**1651740**
fnl4461-1	247830	263040	264278	**268554**	695519	706454	708201	**720026**	2308650	2335839	2357480	**2393520**
pla7397-1	363395	392979	**396697**	394495	1373580	1427093	**1429701**	1423741	4350370	5074983	**5228070**	5000960
rl11849-1	653507	706972	**717283**	707586	1880390	1929624	**1959430**	1932340	6439530	6607124	**6719360**	6581260
usa13509-1	749344	804847	**817734**	810759	2379690	2569175	**2639170**	2596250	9156090	9306518	**9578140**	9445360
brd14051-1	760496	874655	885054	**891925**	2313280	2504101	2550740	**2566320**	8020670	8581641	8869890	**8873420**
d15112-1	858458	938614	**973416**	938768	2397170	2514992	**2689720**	2539980	8662630	9191849	**10335900**	9296610
d18512-1	961759	1071013	973416	**1076830**	2820810	2989434	3024180	**3056510**	14898000	16419198	16662700	**16792324**
pla33810-1	1688720	1863667	**1931620**	1835050	5337440	5936408	**6140350**	6007239	19460600	20406280	**20611150**	19770750
Best number	0	0	6	**14**	0	0	7	**13**	0	0	9	**11**

Table 2: Comparison with three best-performing TTP algorithms in the literature within the same time limit

TTP algorithms from the literature: MATLS [Mei *et al.*, 2014], S5 [Faulkner *et al.*, 2015] and CoCo [Namazi *et al.*, 2023]. Table 2 lists the best average results of these algorithms, consistent with previous studies. These 60 instances are selected from the first instance file of each category, typically used for comparing results of the TTP algorithms, indicated by the suffix "-1". One easily observes that our ReinforceNS consistently dominates MATLS and S5 in terms of the best average result for all 60 instances. Additionally, ReinforceNS outperforms CoCo for 38 out of the 60 instances, especially for the two benchmark sets CatA and CatB with small and medium sizes.

The 95% confidence intervals based on the best average results in Table 2 are further introduced for a statistical evaluation of these algorithms. They are calculated using Eq. (9).

$$95\%\ \text{Confidence Interval} = \mu \pm 1.96 \times \frac{\sigma}{\sqrt{z}} \qquad (9)$$

Here, μ represents the mean of the Relative Deviation Index RDI $= \frac{\text{res}-\text{min}}{\text{max}-\text{min}} \times 100$ [Kim and Kim, 1996], where res is the result of the reference algorithm, min and max are the minimum and maximum results among all the algorithms, respectively. The σ denotes the standard deviation of results, and z represents the number of instances. Figure 2 illustrates the 95% confidence intervals for these algorithms, highlighting that ReinforceNS outperforms the three comparison algorithms and demonstrates stable performance with the lowest variance.

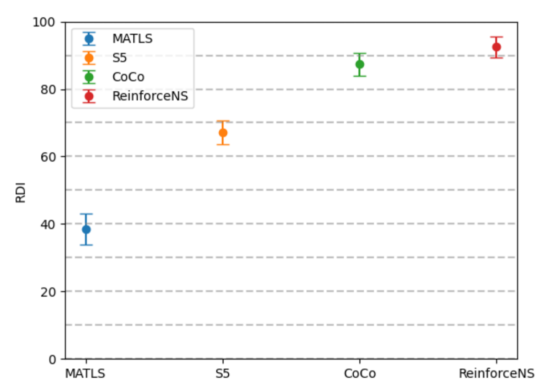

Figure 2: The 95% confidence intervals for ReinforceNS and three top-performing TTP algorithms (MATLS, S5, CoCo)

Ablation study. To demonstrate the effectiveness of incorporating Reinforcement Learning (RL), we conduct an ablation study comparing ReinforceNS with RL against ReinforceNS without RL, denoted as ReinforceNS w/o RL. The

Instance	CatA		CatB		CatC	
	ReinforceNS w/o RL	ReinforceNS	ReinforceNS w/o RL	ReinforceNS	ReinforceNS w/o RL	ReinforceNS
eil76-1	4425	**4492**	**13014**	12815	41079	**41627**
kroA100-1	4565	**4723**	13057	**13425**	41284	**41284**
ch130-1	9432	**9586**	23337	**23590**	75319	**75739**
u159-1	**9052**	9017	25806	**25830**	**81058**	79855
a280-1	18899	**19034**	39479	**39540**	143452	**143500**
u574-1	**27892**	27592	**104573**	104159	342521	**342746**
u724-1	51253	**51531**	**119115**	118434	**393002**	391756
dsj1000-1	144283	**144415**	151821	**157509**	515114	**526704**
rl1304-1	**81667**	81276	**221059**	220789	762346	**772945**
fl1577-1	101337	**106730**	261280	**261846**	920634	**924266**
d2103-1	123927	**124269**	323658	**325730**	1174880	**1178710**
pcb3038-1	**165357**	164526	**527866**	525384	1644460	**1651740**
fnl4461-1	**268992**	268554	719383	**720026**	**2388919**	2393520
pla7397-1	392257	**394495**	1419201	**1423741**	**5046910**	5000960
rl11849-1	**713446**	707586	**1938050**	1932340	6520150	**6581260**
usa13509-1	807241	**810759**	2523510	**2596250**	9326630	**9445360**
brd14051-1	890845	**891925**	**2525700**	2566320	**8710360**	8873420
d15112-1	**965523**	938768	**2590480**	2539980	9234710	**9296610**
d18512-1	**1096370**	1076830	3041130	**3056510**	16422100	**16792324**
pla33810-1	1793450	**1835050**	5935330	**6007239**	19699160	**19770750**
Best number	8	**12**	8	**12**	5	**15**

Table 3: Comparisons between ReinforceNS with and without reinforcement learning

comparative results are summarized in Table 3. From the table, we can observe that ReinforceNS achieves superior performances, surpassing ReinforceNS w/o RL by 12, 12 and 15 instances in CatA, CatB and CatC, respectively. This suggests that the RL mechanism can indeed select promising neighboring solutions and facilitate an efficient search, leading to better results in most instances. Moreover, ReinforceNS with RL demonstrates competitive performance against ReinforceNS w/o RL, particularly on the large TTP instances. Additionally, Figure 3 shows that both ReinforceNS and ReinforceNS w/o RL demonstrate greater stability than the three top-performing TTP algorithms. Among them, ReinforceNS stands out as the most stable with the lowest variance.

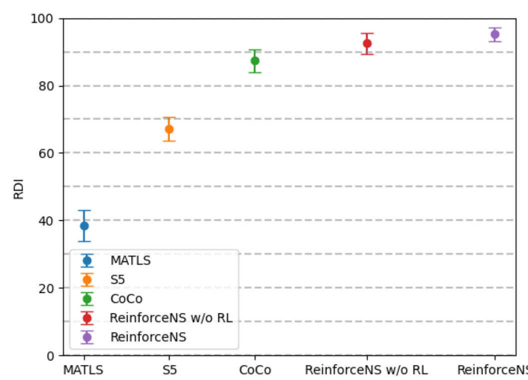

Figure 3: The 95% confidence intervals for ReinforceNS w/o RL, ReinforceNS and three best-performing TTP algorithms (MATLS, S5, CoCo)

6 Conclusion

In this paper, we propose the first learning-based algorithm utilizing the great power of reinforcement learning mechanism to combine with an effective neighborhood search heuristic for TTP. The proposed ReinforceNS algorithm incorporates a dedicated initialization procedure to identify promising regions, a neighborhood search procedure reinforced by a Q-learning strategy to find high-quality local optima, a post-optimization procedure for further improvement, and a multi-start mechanism to overcome deep local optima traps. We perform extensive experiments of the ReinforceNS on three sets of 60 benchmark instances. The comparisons with three state-of-the-art algorithms demonstrate the high competitiveness of our ReinforceNS in terms of solution quality and stability. In particular, ReinforceNS achieves improved results for 12 out of the 18 instances that were publicly reported in the recent TTP competition at GECCO 2023. This work provides supporting evidence for using reinforcement learning to improve the efficiency of heuristics. As such it would be interesting to investigate other machine learning or deep learning techniques for solving TTP for future work.

Acknowledgements

This work was supported in part by CCF-Huawei Populus Grove Fund and the National Natural Science Foundation of China (No. 62302174).

References

[Afshar *et al.*, 2020] Reza Refaei Afshar, Yingqian Zhang, Murat Firat, and Uzay Kaymak. A state aggregation approach for solving knapsack problem with deep reinforcement learning. In *Asian Conference on Machine Learning*, pages 81–96. PMLR, 2020.

[Ali and Mohamedkhair, 2021] Fathelrahman Ali and Mohamedelfatih Mohamedkhair. Hyper-heuristic approaches for the travelling thief problem. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pages 1–6. IEEE, 2021.

[Applegate *et al.*, 2003] David Applegate, William Cook, and André Rohe. Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, page 82–92, Feb 2003.

[Bonyadi *et al.*, 2013] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *2013 IEEE Congress on Evolutionary Computation*, Jun 2013.

[Bonyadi *et al.*, 2014] Mohammad Reza Bonyadi, Zbigniew Michalewicz, Michal Roman Przybylek, and Adam Wierzbicki. Socially inspired algorithms for the travelling thief problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, Jul 2014.

[Cacchiani *et al.*, 2022] Valentina Cacchiani, Manuel Iori, Alberto Locatelli, and Silvano Martello. Knapsack problems—an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143:105693, 2022.

[El Yafrani and Ahiod, 2016] Mohamed El Yafrani and Belaïd Ahiod. Population-based vs. single-solution heuristics for the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, Jul 2016.

[El Yafrani and Ahiod, 2017] Mohamed El Yafrani and Belaïd Ahiod. A local search based approach for solving the travelling thief problem: The pros and cons. *Applied Soft Computing*, 52:795–804, 2017.

[El Yafrani and Ahiod, 2018] Mohamed El Yafrani and Belaïd Ahiod. Efficiently solving the traveling thief problem using hill climbing and simulated annealing. *Information Sciences*, 432:231–244, 2018.

[Faulkner *et al.*, 2015] Hayden Faulkner, Sergey Polyakovskiy, Tom Schultz, and Markus Wagner. Approximate approaches to the traveling thief problem. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, Jul 2015.

[Flood, 1956] Merrill M Flood. The traveling-salesman problem. *Operations research*, 4(1):61–75, 1956.

[Helsgaun, 2000] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

[Hu and Leida, 2004] Jian-Qiang Hu and Brett Leida. Traffic grooming, routing, and wavelength assignment in optical wdm mesh networks. In *IEEE INFOCOM 2004*, volume 1. IEEE, 2004.

[Iori and Martello, 2010] Manuel Iori and Silvano Martello. Routing problems with loading constraints. *TOP*, 18(1):4–27, Jul 2010.

[Kim and Kim, 1996] Jung-Ug Kim and Yeong-Dae Kim. Simulated annealing and genetic algorithms for scheduling products with multi-level product structure. *Computers & Operations Research*, 23(9):857–868, 1996.

[Lienland and Zeng, 2015] Bernhard Lienland and Li Zeng. A review and comparison of genetic algorithms for the 0-1 multidimensional knapsack problem. *International Journal of Operations Research and Information Systems (IJORIS)*, 6(2):21–31, 2015.

[Martins *et al.*, 2017] Marcella SR Martins, Mohamed El Yafrani, Myriam RBS Delgado, Markus Wagner, Belaïd Ahiod, and Ricardo Lüders. Hseda: a heuristic selection approach based on estimation of distribution algorithm for the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 361–368, 2017.

[Mei *et al.*, 2014] Yi Mei, Xiaodong Li, and Xin Yao. Improving efficiency of heuristics for the large scale traveling thief problem. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 631–643. Springer, 2014.

[Mei *et al.*, 2015] Yi Mei, Xiaodong Li, Flora Salim, and Xin Yao. Heuristic evolution with genetic programming for traveling thief problem. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015.

[Namazi *et al.*, 2023] Majid Namazi, MA Hakim Newton, Conrad Sanderson, and Abdul Sattar. Solving travelling thief problems using coordination based methods. *Journal of Heuristics*, pages 1–58, 2023.

[Osaba *et al.*, 2020] Eneko Osaba, Xin-She Yang, and Javier Del Ser. Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics. *Nature-inspired computation and swarm intelligence*, pages 135–164, 2020.

[Pan *et al.*, 2023] Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. H-tsp: Hierarchically solving the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9345–9353, 2023.

[Pierotti *et al.*, 2021] Jacopo Pierotti, Maximilian Kronmueller, Javier Alonso-Mora, J Theresia van Essen, and Wendelin Böhmer. Reinforcement learning for the knapsack problem. In *Optimization and Data Science: Trends and Applications: 5th AIROYoung Workshop and AIRO PhD School 2021 Joint Event*, pages 3–13. Springer, 2021.

[Pisinger and Toth, 1998] David Pisinger and Paolo Toth. *Knapsack Problems*, page 299–428. Jan 1998.

[Polyakovskiy and Neumann, 2017] Sergey Polyakovskiy and Frank Neumann. The packing while traveling problem. *European Journal of Operational Research*, 258(2):424–439, 2017.

[Polyakovskiy *et al.*, 2014] Sergey Polyakovskiy, Mohammad Reza Bonyadi, Markus Wagner, Zbigniew Michalewicz, and Frank Neumann. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, Jul 2014.

[Reinelt, 1991] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA Journal on Computing*, page 376–384, Nov 1991.

[Stützle and Hoos, 2000] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[Tu *et al.*, 2023] Chaofan Tu, Ruibin Bai, Uwe Aickelin, Yuchang Zhang, and Heshan Du. A deep reinforcement learning hyper-heuristic with feature fusion for online packing problems. *Expert Systems with Applications*, page 120568, 2023.

[Wagner *et al.*, 2018] Markus Wagner, Marius Lindauer, Mustafa Mısır, Samadhi Nallaperuma, and Frank Hutter. A case study of algorithm selection for the traveling thief problem. *Journal of Heuristics*, 24:295–320, 2018.

[Wagner, 2016] Markus Wagner. *Stealing Items More Efficiently with Ants: A Swarm Intelligence Approach to the Travelling Thief Problem*, page 273–281. Jan 2016.

[Weiszer *et al.*, 2014] Michal Weiszer, Jun Chen, Stefan Ravizza, Jason Atkin, and Paul Stewart. A heuristic approach to greener airport ground movement. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, Jul 2014.

[Wu *et al.*, 2017] Junhua Wu, Markus Wagner, Sergey Polyakovskiy, and Frank Neumann. Exact approaches for the travelling thief problem. In *Simulated Evolution and Learning: 11th International Conference, SEAL 2017, Shenzhen, China, November 10–13, 2017, Proceedings 11*, pages 110–121. Springer, 2017.

[Wuijts and Thierens, 2019] Rogier Hans Wuijts and Dirk Thierens. Investigation of the traveling thief problem. In *Proceedings of the genetic and evolutionary computation conference*, pages 329–337, 2019.

[Xin *et al.*, 2021] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34:7472–7483, 2021.

[Zhang *et al.*, 2021] Zitong Zhang, Lei Yang, Peipei Kang, Xiaotian Jia, and Wensheng Zhang. Solving the traveling thief problem based on item selection weight and reverse-order allocation. *IEEE Access*, 9:54056–54066, 2021.

[Zheng *et al.*, 2021] Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li. Combining reinforcement learning with lin-kernighan-helsgaun algorithm for the traveling salesman problem. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 12445–12452, 2021.