PARETO FRONTIER APPROXIMATION NETWORK (PA-NET) APPLIED TO MULTI-OBJECTIVE TSP

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective optimization is used in various areas of robotics like control, planning etc. Their solutions are dependent on multiple objective functions, which can be conflicting in nature. In such cases, the optimality is defined in terms of Pareto optimality. A set of these Pareto Optimal solutions in the objective space form a Pareto front (or frontier). Each solution has its own trade off. For instance, the travelling salesman problem (TSP) is used in robotics for task/resource allocation. Often this allocation is influenced by multiple objective functions and is solved using Multi-objective travelling salesman problem (MOTSP). In this work, we present PA-Net, a network that generates good approximations of the Pareto front for the multi-objective optimization problems. Our training framework is applicable to other multi-objective optimization problems; however, in this work, we focus on solving MOTSP. Firstly, MOTSP is converted into a constrained optimization problem. We then train our network to solve this constrained problem using the Lagrangian relaxation and policy gradient. With PA-Net we are able to generate good quality Pareto fronts with fast inference times. Finally, we present the application of PA-Net to find optimal visiting order in coverage planning.

1 INTRODUCTION

Travelling Salesman Problem (TSP) is a popular sequencing problem. TSP aims at finding a sequence or a tour of visiting each node in a given graph and finally returning to the starting node such that the overall cost is minimized. TSP and its variants are widely used in robotics for applications like path planning for UAVs (Xu & Che, 2019), multi-robot path planning (Yu et al., 2002), task allocation for robotic manipulators (Zacharia & Aspragathos, 2005), coverage planning (Bormann et al., 2018) etc.

We intend to use multi-objective travelling salesman problem (MOTSP) for application of coverage planning. Algorithms for coverage planning generates trajectories for robots that can cover a given area (Bormann et al., 2018). Area coverage is used in robotic applications like cleaning robots, surveillance etc. Grid based TSP planners (Bormann et al., 2018), segment a given map into multiple cells, and generate a coverage pattern for each cell.The optimal visiting order for these cells is generated by solving TSP that minimizes the length of the tour. Now imagine a scenario where a robot has to visit these cells and the order is dependent on multiple objectives such as tour length, priority order of the cells etc. So MOTSP would be a more appropriate choice in such a scenario. There is a wide variety of algorithms ranging from exact methods to evolution based methods to solve MOTSP (Lust & Teghem, 2010). Evolutionary algorithms like non-dominated sorting genetic algorithm-II (NSGA-II) (Beirigo & dos Santos, 2016) and multi-objective evolutionary algorithm (MOEA/D) (Peng et al., 2009) are a popular choice of methods to tackle MOTSP and other multi objective optimization problems. Many algorithms also use evolutionary algorithms coupled with local search heuristics (Jaszkiewicz, 2002; Ke et al., 2014; Cai et al., 2014). In practice, these evolutionary based methods suffer in performance and computation time with an increase in the scale of the problem (Zhang et al., 2016). We intend to address these issues using Deep Reinforcement Learning (DRL).

Contribution: In this work, we present Pareto frontier approximation network (PA-Net) that generates an approximation of a set of Pareto optimal tours for MOTSP. We use PA-Net to generate MOTSP tours for coverage planning. Our main contributions are: (1) The drawback of existing ap-

proaches (Li et al., 2020) is costly training of separate networks for different preferences in objective space. Our method avoids this costly training through the use of a single network, while providing generalizability over different preferences in objective space. (2) PA-Net can generate solutions for large number of preferences, which contributes to a dense approximation of Pareto front. Our network has competitive results on various metrics evaluating the quality of Pareto front and faster inference times. (3) Our design can be easily extended to generate a set of Pareto optimal solutions for other multi-objective reinforcement learning and Multi-Objective Optimization (MOO) tasks.

Related Work: Sequencing problems are a subset of Combinatorial Optimization (CO) where the decision variables are discrete. Most CO problems are NP-Hard, as a result state-of-the-art algorithms rely on handcrafted heuristics for making decisions that are otherwise too expensive to compute or mathematically not well-defined. Recently, researchers are addressing these issues using deep learning and machine learning (ML) (Bengio et al., 2020; Vesselinova et al., 2020; Mazyavkina et al., 2020). Many recent works have been dedicated to solving TSP and other sequencing problems using deep networks. Google Brain's Pointer Network (Ptr-Net) (Vinyals et al., 2015) learns the conditional probability of an output sequence of elements that are discrete tokens corresponding to positions in an input sequence. They used Ptr-Net to solve Euclidean TSP (and other CO problems) in an end to end fashion, where the solutions from classical methods are used as baselines for training. Similarly, TSP was solved using 2D graph conv-nets followed by beam search (Joshi et al., 2019). RL was used to solve various combinatorial optimization problems in (Bello et al., 2016). Their network uses an RNN based encoder and a Ptr-Net. They trained their network using policy gradient. Similar methodology was adopted in (Deudon et al., 2018). They used transformers as the encoder. Kool et al. (2018) proposed an attention based network which used a baseline based on greedy rollouts. Their network outperformed other deep-learning based solvers. While these methods generated competitive results, however, they still lag in performance in comparison to TSP solvers like OR-Tools and Concorde.

Finding Pareto optimal solutions has been studied in deep learning literature for various multi-objective tasks. In supervised learning tasks, many works focussed on multi-objective classification tasks (Sener & Koltun, 2018; Lin et al., 2019; Mahapatra & Rajan, 2020; Ruchte & Grabocka, 2021; Navon et al., 2021). Similarly, many multi-objective RL methods have been to solve multi-objective MDPs (Roijers et al., 2013; Parisi et al., 2014). Some works in RL train many single policy networks to approximate Pareto front (Vamplew et al., 2017; Li et al., 2020). While others have trained a single network to generate a set of Pareto optimal solutions (Yang et al., 2019; Parisi et al., 2016).

There are various methodologies to tackle MOO problems. In ϵ-constrained methods optimize one of the objectives at a time while using the other objectives as constraints (Mavrotas, 2009; Chinchuluun & Pardalos, 2007). One of the most common ways is to use preference vectors (or weights). Some methods use the weighting vectors to scalarize the objective function and Pareto front can be obtained by solving the optimization for multiple preference vectors (Coello et al., 2009; Boyd et al., 2004). On the other hand, some methods (including ours) use these weighting vectors in constraints (Das & Dennis, 1998).

Li et al. (2020) solved MOTSP by training multiple single policy networks. They converted the MOO problem into a single objective using linear scalarization with the help of preference vectors. They train K different networks, each with different preference vector, to approximate the Pareto front. Their network, called DRL-MOA, generated competitive results in comparison to classical methods. However, the downside of their method is that it is redundant to train multiple networks and requires a lot of resources. Furthermore, solutions on concave regions of Pareto fronts cannot be uncovered using the linear scalarization technique, as proved in (Boyd et al., 2004). In our work, we train a single network that can predict solutions for any preference vector. This enables us to produce a much denser Pareto front. Instead of using linear scalarization, our network learns to solve a constrained optimization problem where the constraints are dependent on the preference vectors.

In this work, we present PA-Net, a framework of deep network trained using policy gradient that can approximate Pareto front for MOO problems. Our choice of using RL is motivated by success of Deep CO methods and the fact that it's hard to generate training data for complex problems like MOTSP. We use PA-Net to find a set of Pareto optimal tours for MOTSP. An augmented version of the network presented in (Deudon et al., 2018) is adopted for PA-Net to solve MOTSP. The novelty of our algorithm is that we pose the problem of finding a set of Pareto optimal solutions as a con-

strained optimization problem, rather than using linear scalarization. We use preference vectors as constraints, which indicates the desired location of the solution in objective space. Finally, we train our network using the reward constrained policy optimization (Tessler et al., 2018), a constrained RL method. Our network performs better than other methods in computation time and generates competitive results in terms of quality of the Pareto front.

2 BACKGROUND

Here we review the definition of Pareto optimality, with a brief primer on solving TSP using DRL.

2.1 PROBLEM SETUP

A MOO is defined as:

$$min_x \ \vec{F}(x) = (f_1(x), f_2(x), ...f_m(x)) \tag{1}$$

where $\vec{F}(x)$ is a vector of m-objective functions and $x \in \mathbf{X}$ is the decision variable in $\mathbb{R}^n$. In such problems, often different objectives are conflicting in nature, i.e. no single solution can simultaneously optimize all the objectives. Instead, a set of Pareto optimal solutions provide the best solutions with different trade-offs between various objectives. Pareto optimality is defined as follows:

- **Dominance:** A solution x^a is said to dominate x^b ($x^a \prec x^b$) if and only if $f_i(x^a) \leq f_i(x^b)$, $\forall i \in \{1, ...m\}$ and $f_j(x^a) < f_j(x^b)$ such that $\exists j \in \{1, ...m\}$.
- **Pareto Optimality**: A solution x^* is said to be Pareto optimal if there does not exist any solution $x^{'}$ such that $x^{'} \prec x^*$. A set of all such points form a Pareto frontier denoted by Υ.

The Euclidean TSP is defined over a graph of n cities, where each city has coordinates $a \in \mathbb{R}^2$. A TSP tour π provides a sequence of visiting cities exactly once and then returning to the starting city. MOTSP is a MOO problem that aims to find a set of Pareto optimal TSP tours Π ($\Pi \subset \mathbb{Z}^n$) on a complete graph s, while optimizing for m objectives. Each city in s can have p a set of features. In case of bi-objective TSP, the input graph s is a sequence of n nodes in a four dimensional space $s = \{a_i^1, a_i^2\}_{i=1:n}$, where $a_i^m \in \mathbb{R}^2$ for each $m \in \{1, 2\}$ (Li et al., 2020). The goal is to find a tour $\pi \in \Pi$ that visits each city once and can simultaneously optimize the objectives for $m \in \{1, 2\}$:

$$f_m(\pi|s) = \left\|a^m_{\pi(n)} - a^m_{\pi(1)}\right\|_2 + \sum_{i=1}^{n-1} \left\|a^m_{\pi(i)} - a^m_{\pi(i+1)}\right\|_2 \tag{2}$$

2.2 TSP USING DRL

An attention based network to solve TSP was proposed by Deudon et al. (2018). We will refer to this network as TSP-Net. We use a modified version of TSP-Net for our network. TSP-Net uses an actor-critic network which is trained using REINFORCE (Williams, 1992). The input to the network is a graph s. Each city coordinates is embedded into a higher dimension. These embedded representations are then passed on to a Multi-Headed attention encoder that generates an encoded representation of the complete graph. Both actor and critic networks share the same encoder architecture. The decoder network of the actor uses a Pointer-Network (Ptr-Net) to generate the TSP tours. The tour is sequentially constructed, where at each step an appropriate city is selected based on the current state and the previous actions by the Ptr-Net. The actor network θ, is trained to minimize the total tour length given by equation 2. The network is trained on a batch of TSP problem instances of size B. The training objective for the actor is given by:

$$D(\theta) = \mathbb{E}_{s \sim \mathcal{S}}[\mathbb{E}_{\pi \sim p_\theta(.|S)}[Q(\pi|S)]] \tag{3}$$

Here, $\mathcal{S}$ is the distribution from which training graphs are drawn and $p_\theta(\pi|S)$ is the probability of a tour generated by the Ptr-Network. The decoder of the critic with parameters ϕ, is a feed forward network that predicts the baseline for the objective function.

3 METHODOLOGY

This section provides our mathematical formulation for MOO. We use the formulation to train a network that can generate a good approximation of the Pareto front for MOTSP.

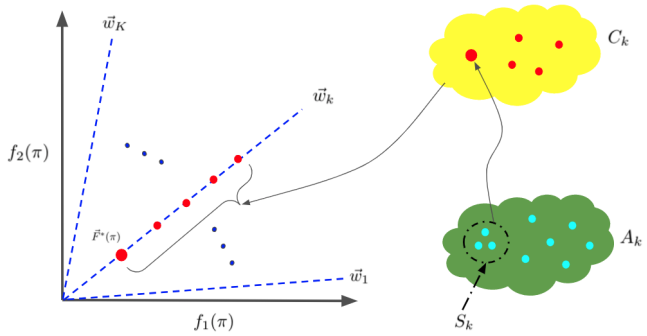


Figure 1: Visualization of surrogate optimization of equation 6 for 2-D cost ($\vec{F}(\pi)$) along the preference vector $\vec{w}_k$. All the members in set C_k map to the line along the preference vector $\vec{w}_k$ in the objective space. The optimum point $\vec{F}^*(\pi)$ dominates all other members in C_k. The dominant point, i.e. $\vec{F}^*$ is unique in C_k, although there can exist different solutions $\pi_k \in S_k$ that map to $\vec{F}^*$.

3.1 Problem Formulation

We intend to generate a good quality approximation to Pareto front denoted by $\widetilde{\Upsilon}$, where $\widetilde{\Upsilon} \subset \Upsilon$. The set $\widetilde{\Upsilon}$ should capture a range of possible dominant solutions in the objective space.

MOTSP is an extension of TSP to the MOO domain. Let the vector cost function for a MOTSP is given by $\vec{F}(\pi)$ where $\vec{F} \in \mathbb{R}^m$ and the tour $\pi \in \Pi$ ($\Pi \subset \mathbb{Z}^n$). The optimization problem can be written as:

$$min_\pi \ \vec{F}(\pi) = (f_1(\pi)...f_m(\pi)) \tag{4}$$

Where $f_i : \pi \rightarrow f_i(\pi)$ for $f_i(\pi) \in \mathbb{R}$. We further assume that all the cost functions are strictly positive:

$$f_i(\pi) > 0 \quad \forall i \in \{1...m\} \tag{5}$$

In order to find the Pareto front, we convert the MOO in equation 4 to a set of constrained optimization problem. This is done by discretizing the objective space using a collection of unit preference vectors $W : \{\vec{w}_1...\vec{w}_K\}$. These preference vectors are a set of rays emanating from origin that uniformly divide the objective space. Each element in $\vec{w}_k$ lies in the interval $\{0, 1\}$ and also $||\vec{w}_k||_2 = 1$. The key idea is to solve a surrogate optimization problem along each preference vector in order to generate a set of dominant solutions for equation 4. This surrogate optimization is expressed as a set of K constrained optimization problems where the k^{th} problem corresponding to $\vec{w}_k \in W$ is given by:

$$\begin{aligned} min_{\vec{F}(\pi_k)} \ & J(\vec{F}(\pi_k)) = ||\vec{F}(\pi_k)||_2 \\ & s.t. \ \ \vec{F}(\pi_k) \in C_k \end{aligned} \tag{6}$$

The constraint set is defined as $C_k = \{\vec{F}(\pi) \in C_k : \ g(\vec{F}(\pi), \vec{w}_k) \leq 0\}$ and the corresponding tour set is defined as $A_k = \{\pi \in A_k : \ g(\vec{F}(\pi), \vec{w}_k) \leq 0\}$. Here, the dot product constraint $g(\vec{F}(\pi_k), \vec{w}_k)$ is given by:

$$g(\vec{F}(\pi_k), \vec{w}_k) = 1 - \frac{\vec{w}_k \cdot \vec{F}(\pi_k)}{J(\pi_k)} \tag{7}$$

We assume A_k is non-empty. As a consequence of this assumption, C_k is also non-empty.

The constraint set C_k represents a set of vector cost $\vec{F}(\pi_k)$ associated with $\pi_k \in A_k$ that lie on the unit preference vector $\vec{w}_k$ in objective space. The objective function in equation 6 minimizes $\mathbb{L}_2$-norm which finds the points closer to origin. Below, we state a theorem that is the motivating factor of our work:

Theorem 1. *$\vec{F}^* \in C_k$ is the optimum solution of equation 6 if and only if it dominates all other points in the set.*

Proof: Let $\vec{F}^{"} \in C_k$ minimize the equation 6 such that $\vec{F}^* \prec \vec{F}^{"}$. This dominance relation implies that $f_i^* \leq f_i^{"}$ $\forall i \in \{1, 2..m\}$ and $\exists j \in \{1, 2..m\}$ such that $f_j^* < f_j^{"}$. This dominance relation leads to the following result:

$$||\vec{F}^*||_2 < ||\vec{F}^{"}||_2 \tag{8}$$

But this result is a contradiction because $\vec{F}^{"}$ minimizes equation 6. Hence, $\vec{F}^* \in C_k$ that dominates all other points in the set is the optimum solution for problem equation 6. ■

The dominant point in the set, i.e. $\vec{F}^* \in C_k$ is also unique in the set. An intuitive proof for this can be visualized using the case for $C_k \subset \mathbb{R}^2$ as shown in Fig. 1. Because of the dot product constraint, all the possible members in the set lie on the unit vector $\vec{w}_k$. It is clear from the image that the point in the set C_k closest to origin dominates all other points and is in fact the optimum solution of equation 6. Further, there can be multiple solutions in A_k that lead to the dominant objective value, i.e. $\vec{F}^*$. Mathematically, the solution set $S_k \subset A_k$, where $S_k = \{\pi^* \in S_k : \vec{F}(\pi^*) = \vec{F}^*\}$, all $\pi_k^* \in S_k$ will generate dominant objective values.

The essence of Theorem 1 is that it demonstrates the viability of approximating the Pareto front for problem in equation 4 through the surrogate optimization problem in equation 6. Solving optimization problem in equation 6 for large values of K can be computationally intractable. We address this issue, through the generalization power of deep neural networks. PA-Net learns to approximately solve equation 6 on a given input preference set and has an ability to generalize to other preferences as well. Unlike linear scalarization methods, our formulation can also find concave Pareto frontier. We demonstrate this with an example in Appendix.

3.2 Training Methodology

The reward constrained policy optimization is an actor-critic algorithm (Tessler et al., 2018). It uses a Lagrangian of the constrained problem as the objective function, where after each gradient update step, the Lagrangian multipliers are updated based on the constraint violation. We use the reward constrained policy optimization to train a network to solve the problem in equation 6 for all $k \in \{1...K\}$.

Algorithm 1: Training of PA-Net

input : $[\theta, \phi, \eta_A, \eta_C]$, $[W, \alpha, \lambda_{min}, \lambda_{max}]$ ← Initialization of network weights, set of preference vectors and corresponding parameters.
output: Trained network parameters of PA-Net θ^*, ϕ^*.

for $i \leftarrow 1, 2...N$ **do**
  $\Omega : \{s^1 .... s^B\}$ ← Sample a Batch of TSP Graphs of size B from distribution $\mathcal{S}$.
  for $k \leftarrow 1, 2...K$ **do**
    for $j \leftarrow 1, 2...B$ **do**
      $\pi_k^j \leftarrow$ Actor network Generates TSP Tour for each s^j and $\vec{w}_k$.
      $b_\phi(\vec{w}_k, s^j) \leftarrow$ Critic Network predicts the baseline
      $L_k(\pi_k^j | s^j) \leftarrow$ Calculate the larangian using equation 11.
  Actor Update: $\theta \leftarrow \theta - \eta_A \cdot \nabla_\theta D_{AC}(\theta)$
  Critic Update: $\phi \leftarrow \phi - \eta_C \cdot \{\frac{2}{B} \sum_{k=1}^{K} \sum_{j=1}^{B} (b_\phi(\vec{w}_k, s^j) - (L_k(\pi_k^j | s^j))\}$
  for $k \leftarrow 1, 2...K$ **do**
    $\lambda_k \leftarrow$ Update the lagrangian multipliers using equation 15

We intend to train a single network that generates a set of dominant tours $T : \{\pi_1, ..., \pi_K\}$. Hence, the problem in equation 6 for each $\pi_k \in T$ can be written in the parametric format as:

$$\begin{aligned} min_\theta \ & J(\pi_k(\theta)) = ||\vec{F}(\pi_k(\theta))||_2 \\ s.t. \ & g_k(\vec{F}(\pi_k(\theta), \vec{w}_k)) \leq 0 \end{aligned} \tag{9}$$

Here, θ is the parameters of the actor network. The Lagrangian dual problem for equation 9 is:

$$L_k(\theta, \lambda_k) \ = \ max_{\lambda_k \geq 0} \ min_\theta \ J(\pi_k(\theta)) + \lambda_k \cdot g_k(\vec{F}(\pi_k(\theta))) \tag{10}$$

Here, λ_k is the k^{th} Lagrangian multiplier corresponding to the preference vector $\vec{w}_k$. PA-Net uses modified TSP-Net in order to generate a dominant set of tours T . We augment TSP-Net network by adding an input of a set of preference vectors W of size K. Each $\vec{w}_k \in W$ is encoded in higher dimensions using a feed-forward network. These additional layers learn features corresponding to different preferences. This encoding is then combined with the encoded representation of the graph and then passed on to the decoder. With this architecture, the network can be trained for various preferences. Complete details regarding the architecture is presented in Appendix. We use the Lagrangian in equation 10 as the reward for the network. The reward for each tour generated by the actor corresponding to each preference for a given graph can be written as:

$$L_k(\pi_k^j, \vec{w}_k|s^j) = J(\pi_k(\theta)|s^j) + \lambda_k \cdot g_k(\vec{F}(\pi_k(\theta))|s^j). \tag{11}$$

Based on this reward, the training objective for the actor can be written in our case as:

$$D_{AC}(\theta) = \mathbb{E}_{s\sim\mathcal{S}}[\mathbb{E}_{\vec{w}_k\sim W}[\mathbb{E}_{\pi\sim p_\theta(.|s)}[L_k(\pi, \vec{w}_k|s)]]]. \tag{12}$$

The critic network provides predictions $b_\phi(\vec{w}_k, s^j)$ on the reward given in equation 11. The critic network is trained on the mean squared error between its predictions and rewards of the actor, which is given by:

$$D_{CR}(\phi) = \frac{1}{B}\sum_{k=1}^{K}\sum_{j=1}^{B}(b_\phi(\vec{w}_k, s^j) - (L_k(\pi_k^j|s^j))^2 \tag{13}$$

The gradient for the training of the actor network is approximated using REINFORCE (Williams, 1992):

$$\nabla_\theta D_{AC}(\theta) \approx \frac{1}{B}\sum_{k=1}^{K}\sum_{j=1}^{B}[(L_k(\pi_k^j|s^j) - b_\phi(\vec{w}_k, s^j)) \cdot \nabla_\theta log(p_\theta(\pi_k^j|s^j))]. \tag{14}$$

The description of the training of PA-Net is given in **Algorithm 1**. We start with initialization of weights and learning rates for the network and the set of preference vectors, along with other hyperparameters that are the ascent rate of the Lagrangian multipliers α and $[\lambda_{min}, \lambda_{max}]$ the limits for the multipliers. The network is trained for N iterations. At each iteration, a set of graphs Ω of size B are generated. For each $s^j \in \Omega$ corresponding to every preference vector, $\vec{w}_k \in W$ a tour π_k^j. Based on generated tours, the objective for actor and critic are calculated. Then parameters of the network are updated using gradient descent. At the end of each iteration, the Lagrangian multiplier corresponding to each preference vector are updated in an ascent step using:

$$\lambda_k^{i+1} = \Gamma_\lambda(\lambda_k^i + \alpha \cdot \sum_{j=1}^{B} g_k(F(\pi_k^j(\theta), \vec{w}_k))) \tag{15}$$

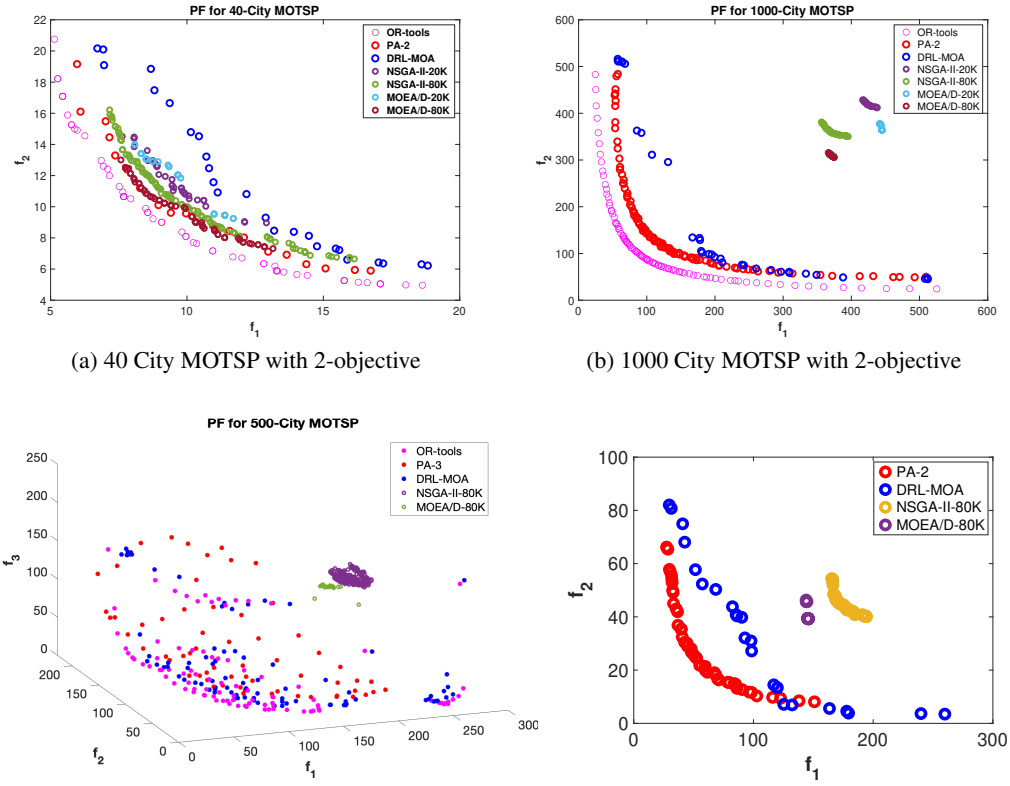
Here $\Gamma_\lambda(\cdot)$ ensures that the multipliers remains within the limits, i.e. $[\lambda_{min}, \lambda_{max}]$ and α is prespecified ascent rate. Although the network is trained on a fixed set of preferences W, it however can generalize to a larger set of preferences. A detailed discussion on preference selection mechanism and study on effect on network performance due to some key hyperparameters is presented in the Appendix.

There are a few caveats in solving equation 6 with our approach. The guarantees provided in **Theorem 1** may not hold because the reward constrained policy optimization guarantees convergence of equation 9 to a saddle point $(\theta^*(\lambda^*), \lambda*)$ which may not lead to the global optimum of equation 6. Further, a strong duality would have to be established between equation 9 and its dual form in equation 10. However, in practice our approach is able to generate competitive results as presented next.

4 Experiments

In this section, we present experiments to evaluate the efficacy of PA-Net and use case of two objective MOTSP for coverage planning. The performance of PA-Net and other algorithms is compared on MOTSP instances with {2, 3 and 5} objectives, respectively. The performance of PA-Net is compared with deep learning based method DRL-MOA (Li et al., 2020), evolution based strategies: NSGA-II and MOEA/D (Peng et al., 2009) and OR-Tools solver (using linear scalarization) that primarily uses local search methods and meta-heuristics.

Experimental Details: All the above-mentioned algorithms are evaluated on a set of 25 randomly generated MOTSP instances. Each MOTSP instance is a graph s of size $n \times t$, where, n the number of



(a) 40 City MOTSP with 2-objective

(b) 1000 City MOTSP with 2-objective

(c) 500 City MOTSP with 3-objective

(d) 500 City MOTSP with 2-objective for CP

Figure 2: Visualization of the dominant solutions for different problem instances. It can be seen that our network (PA-2 and PA-3) generates significantly better objective values

cities and each city is represented by t dimensional features. These features are drawn uniformly at random from $[0, 1]$. Here, $t = 4$ for $\{2, 3\}$-objective TSP and $t = 6$ for 5-objective TSP. Two forms of objective functions are used, one with reward of $\mathbb{L}_1$-norm and the other one with $\mathbb{L}_2$-norm. For 2-objective instances, we use $\mathbb{L}_2$-norm for both objective. In the case of $\{3, 5\}$-objective instances, we use $\mathbb{L}_2$-norm for f_1 and $\mathbb{L}_1$-norm for the remaining. The objective with $\mathbb{L}_2$-norm is generated using equation 2. Similarly, the objective for $\mathbb{L}_1$-norm for $b^* \in \mathbb{R}$ is:

$$f_i(\pi|s) = \left\|b^*_{\pi(n)} - b^*_{\pi(1)}\right\|_1 + \sum_{j=1}^{n-1} \left\|b^*_{\pi(ij)} - b^*_{\pi(j+1)}\right\|_1 \tag{16}$$

The results of PA-Net are compared with other methods on the basis of average run time and the average hypervolume (HV) of the solutions obtained across 25 instances. HV is a hybrid metric used to evaluate Pareto fronts (Bérubé et al., 2009; Audet et al., 2020). It represents the volume covered by the non-dominated set of solutions with respect to the a reference point.A higher HV represents a better quality of the Pareto front, both in terms of optimality and coverage of the objective space. We use Monte-carlo sampling to calculate HV, where, HV is given by the percentage of points dominated by the solution set out of randomly sampled points in a fixed volume in the objective space. A fixed reference point is used to compare all the algorithms. It is calculated as the product of number of cities and unity vector, for example, reference point for 200-city bi-objective TSP is $[200.0, 200.0]^\top$.

For bi-objective TSP 100 solutions are evaluated for each case. In {3,5}-objective MOTSP 91 and 40 solutions are reported. Further, we also report 500 solutions for PA-Net in {3,5}-objective MOTSP to demonstrate its generalization ability. It should be noted that the preferences used in linear scalar-

ization are sampled from [0,1] and satisfy convexity constraints. In PA-Net the preferences used are unit vectors that uniformly segment the objective space.

The trained model of bi-objective TSP for DRL-MOA (Li et al., 2020) is used. For NSGA-II and MOEA/D, we use a MATLAB based software platform, PlatEMO (Tian et al., 2017). The experiments for PA-Net and DRL-MOA are carried out on NVIDIA V100 Volta GPU. Whereas, for NSGA-II, MOEA/D and OR tools experiments are carried out on dual-core Intel i5 processor.

Training Details:For all three instances of MOTSP problems we train separate networks i.e. PA-2, PA-3 and PA-5 for $\{2, 3, 5\}$-objective TSP respectively. All PA-Nets are trained on 120 city MOTSP instances. The preferences for PA-Net are randomly sampled (more details in Appendix).

Table 1: Training Details

	2-Obj		**3-Obj**		**5-Obj**	
	PA-2	DRL-MOA	PA-3	DRL-MOA	PA-5	DRL-MOA
Batch size	60	200	60	200	60	200
Epochs	1	5	1	5	1	5
Input Graph size	120×4	40×4	120×4	40×4	120×6	40×6
Steps (per epoch)	20000	2000	25000	2000	20000	2000
Training time (hrs)	~ 14	$\sim 100 \times 0.70$	~ 23	$\sim 91 \times 0.70$	~ 18	$\sim 40 \times 1$

The details of training of PA-Net and DRL-MOA are given in Table-1. The times are reported based on training from NVIDIA V100 Volta GPU. Note that in case of DRL-MOA, average training time for each preference network is $\sim 1\,hr$ (for $\{2,3,5\}$ objective). So 100 networks are trained with the total training time $\sim 70hrs$.

Discussion: The results of quantitative comparison for 2-objective MOTSP is given in Table-2. Similarly, the results for $\{3, 5\}$-objective is summarized Table-3. Visualization for various MOTSP instances is given in Fig-2 (a)-(c). It is clear that OR tools achieves the best performance in terms of HV. However, it has longer runtimes as compared to other methods. On the other hand, evolutionary methods significantly underperform as the scale of the problem increases. This is likely because these algorithms are unable to explore the solution space well for larger problems. Running these algorithms for more iterations could potentially improve their performance in terms of HV, but this comes with an additional computational cost. Both PA-Net and DRL-MOA achieve competitive results in terms of HV. Our network achieves better performance as compared to DRL-MOA in 2 objective problem. PA-Net generates the complete Pareto front much faster as compared to other methods. Further, we have significantly lowered the training times as compared to DRL-MOA, see Table. 1. Another notable point is that for each problem set, PA-Net can infer solution from a single network, whereas DRL-MOA has to train and rely on multiple networks. For the $\{3,5\}$-objective case, while our network is much faster than DRL-MOA, our network lags behind in HV when using lower number of preferences. This issue can certainly be addressed with prolonged training and better tuning of the network and use of local search heuristics to refine solutions.

Table 2: Quantitative comparison of Pareto front for 2-Objective MOTSP

	40-City		**200-City**		**500-City**		**1000-City**	
Algo.	HV (%)	Time (s)	HV (%)	Time (s)	HV (%)	Time (s)	HV (%)	Time (s)
NSGA-II (20K)	67.3	5.64	45.4	8.04	38.4	14.7	33.8	27.1
NSGA-II (80K)	72.5	21.7	53.9	30.8	46.36	58.7	41.48	107.3
MOEA/D (20K)	66.7	9.357	47.5	12.7	40.4	20.7	35.7	33.5
MOEA/D (80K)	70.7	34.65	55.98	48.2	48.8	79.53	43.89	130.75
DRL-MOA	73.6	5.87	80.63	29.3	84.5	72.9	85.9	145.5
PA-2 (ours)	75.4	**1.59**	83.2	**6.03**	86.92	**15.14**	88.45	**30.38**
OR-tools (LS)	**78.14**	2.16	**86.5**	86.8	**91.07**	732	**93.39**	3730

Table 3: Quantitative comparison of the Pareto front, for 3,5-Objective MOTSP

		200-City		500-City	
Problem	Algo.	HV (%)	Time (s)	HV (%)	Time (s)
3-obj	NSGA-II (80K)	44.24	35.3	36.15	78.6
	DRL-MOA	86.3	26.6	89.8	66.3
	PA-3 ($K = 91$)	84.72	**5.2**	88.05	**12.6**
	PA-3 ($K = 500$)	85.41	29.27	88.51	70.8
	OR Tools (LS)	**89.71**	77.8	**93.4**	672.9
5-obj	NSGA-II (80K)	21.6	55.2	16.6	133.2
	MOEA/D (80K)	30.1	103.54	22.8	180.4
	DRL-MOA	64.6	12.1	69.3	30.1
	PA-5 ($K = 40$)	62.3	**2.6**	67,4	**6.35**
	PA-5 ($K = 500$)	67.3	29.9	71.8	72.6
	OR Tools (LS)	**69.8**	34.5	**76.32**	299.2

Application for Coverage Planning: We test our network for coverage planning. We assume a scenario where the robot has to visit all the cells while ensuring the maximum adherence to a pre-computed priority order. Such a scenario is representative of real-world applications. For instance, a cleaning robot has to clean a large area where different regions have varied priorities based on the number of people visiting those areas. So the goal is to visit all the cells while minimizing the total distance travelled and maximizing the adherence to pre-computed priority order. This task can be cast as a 2-objective MOTSP instance. For this experiment, we use a different graph than the one used in the previous experiment. Each cell in the graph has four features $\{a, b, 0\}$. Here, $a \in \mathbb{R}^2$ is the Euclidean coordinates of the and $b \in \mathbb{R}^+$ is the priority. Note that a lower value of b corresponds to greater priority. So for this task we synthetically generate TSP instances of size 200 and 500 cells respectively. We compute tours from PA-2 and other algorithms. Comparative results for different algorithms for the coverage planning task are listed in Table-4. The plots for the Pareto front for this experiment are shown in Fig-2(d). An intuitive visualization of tours generated in this experiment can be found in Appendix and here.

Table 4: Quantitative comparison of the Pareto front for coverage planning.

	200-City	500-City
Algo.	HV (%)	HV (%)
NSGA-II (80K)	52.9	39.9
MOEA/D (80K)	60.4	43.2
DRL-MOA	88.8	82
PA-2 (ours)	**89.6**	**84.34**

5 Conclusions

We presented PA-Net, a network that approximates the Pareto frontier for the multi-objective TSP. Our results indicate a superior performance in terms of optimality of the solutions. This is achieved by segmenting the objective space using a set of unit vectors which represent trade-offs among various objectives. We then use these preference vectors to convert the unconstrained optimization problem into a set of constrained optimization problems. Then the network is trained using policy gradient to generate solutions for these constrained problems. While PA-Net is trained on a fixed number of preference vectors, it generalizes well to other unseen preferences as well. The effectiveness of our method is highlighted by the competitive results in terms of quality of solutions, faster inference and training times. Although we focus on multi-objective TSP in this work, our training framework can be applied to other MOO problems. We also demonstrated a use case of PA-net for a simple coverage planning application. The future direction is extending the work to multi-robot system that can also account for the uncertainty in the environment.

REFERENCES

Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European journal of operational research*, 2020.

Breno Alves Beirigo and André Gustavo dos Santos. Application of nsga-ii framework to the travel planning problem using real-world travel data. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 746–753. IEEE, 2016.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

Jean-François Bérubé, Michel Gendreau, and Jean-Yves Potvin. An exact -constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European journal of operational research*, 194(1):39–50, 2009.

Richard Bormann, Florian Jordan, Joshua Hampp, and Martin Hägele. Indoor coverage path planning: Survey, implementation, analysis. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1718–1725. IEEE, 2018.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Xinye Cai, Yexing Li, Zhun Fan, and Qingfu Zhang. An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 19(4):508–523, 2014.

Altannar Chinchuluun and Panos M Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154(1):29–50, 2007.

Carlos Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan. *Advances in multi-objective nature inspired computing*, volume 272. Springer, 2009.

Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, 8 (3):631–657, 1998.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pp. 170–181. Springer, 2018.

Andrzej Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, 2002.

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Liangjun Ke, Qingfu Zhang, and Roberto Battiti. Hybridization of decomposition and local search for multiobjective optimization. *IEEE transactions on cybernetics*, 44(10):1808–1820, 2014.

Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Kaiwen Li, Tao Zhang, and Rui Wang. Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, 2020.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 12060–12070, 2019.

Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: a survey and a new approach. In *Advances in Multi-Objective Nature Inspired Computing*, pp. 119–141. Springer, 2010.

Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*, pp. 6597–6607. PMLR, 2020.

George Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, 213(2):455–465, 2009.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *arXiv preprint arXiv:2003.03600*, 2020.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NjF772F4ZZR.

Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 2323–2330. IEEE, 2014.

Simone Parisi, Matteo Pirotta, and Marcello Restelli. Multi-objective reinforcement learning through continuous pareto manifold approximation. *Journal of Artificial Intelligence Research*, 57:187–227, 2016.

Wei Peng, Qingfu Zhang, and Hui Li. Comparison between moea/d and nsga-ii on the multi-objective travelling salesman problem. In *Multi-objective memetic algorithms*, pp. 309–324. Springer, 2009.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Michael Ruchte and Josif Grabocka. Efficient multi-objective optimization for deep learning. *arXiv preprint arXiv:2103.13392*, 2021.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017.

Peter Vamplew, Rustam Issabekov, Richard Dazeley, Cameron Foale, Adam Berry, Tim Moore, and Douglas Creighton. Steering approaches to pareto-optimal multiobjective reinforcement learning. *Neurocomputing*, 263:26–38, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Natalia Vesselinova, Rebecca Steinert, Daniel F Perez-Ramirez, and Magnus Boman. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 8:120388–120416, 2020.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems*, pp. 2692–2700, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Yunpeng Xu and Chang Che. A brief review of the intelligent algorithm for traveling salesman problem in uav route planning. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 1–7. IEEE, 2019.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, pp. 14636–14647, 2019.

Zhong Yu, Liang Jinhai, Gu Guochang, Zhang Rubo, and Yang Haiyan. An implementation of evolutionary computation for path planning of cooperative mobile robots. In *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)*, volume 3, pp. 1798–1802. IEEE, 2002.

P Th Zacharia and NA Aspragathos. Optimal robot task scheduling based on genetic algorithms. *Robotics and Computer-Integrated Manufacturing*, 21(1):67–79, 2005.

Xingyi Zhang, Ye Tian, Ran Cheng, and Yaochu Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):97–112, 2016.

APPENDIX

CONVERGENCE TO CONCAVE PARETO FRONTS

The backbone of PA-Net is as follows: to generate a Pareto front for the Multi Objective Optimization (MOO) problem in equation 4, we solve a set of K constrained surrogate optimization problem given by equation 6. In order to show that our framework can work for MOO problems with concave frontier, we have generated solutions for a MOO problem with concave Pareto front by solving equation 6 for K preferences. The MOO problem we solve is taken from Lin et al. (2019) and is given by:

$$min_x \, \vec{F}(x) = [f_1(x), f_2(x)]^\top, \tag{17}$$

where,

$$\begin{aligned} f_1(x) &= 1 - exp(-\Sigma_{i=1}^{d}(x_i - \frac{1}{\sqrt{d}})^2), \\ f_2(x) &= 1 - exp(-\Sigma_{i=1}^{d}(x_i + \frac{1}{\sqrt{d}})^2). \end{aligned} \tag{18}$$

Here $x = [x_1, x_2]^\top \in \mathbb{R}^{2+}$ and $d = 2$. The surrogate optimization in this case with preference $\vec{w_k}$ is given by:

$$\begin{aligned} min_{\vec{F}(x^k)} \; & J(\vec{F}(x^k)) = ||\vec{F}(x^k)||_2 \\ s.t. \; & 1 - \frac{\vec{w}_k \cdot \vec{F}(x^k)}{J(\vec{F}(x^k))} \leq 0 \end{aligned} \tag{19}$$

We solve the above problem in Matlab for $K = 20$. The preference is generated by $\vec{w_k} = [cos(\phi_k), sin(\phi_k)]^\top$, where $\phi_k \in \{0, 90\}$.

We also solve the MOO problem with a simple linear scalarization of objective. In this case, the preference is given by $\alpha^k = [\alpha_1, \alpha_2]^\top \in \mathbb{R}^{2+}$ such that $\alpha_1 + \alpha_2 = 1$. We use $K = 100$ prefrences in this case. The k^{th}objective function for linear scalarization is:

$$min_{\vec{F}(x^k)} \; R(\vec{F}(x^k)) = \alpha_1 \cdot f_1(x^k) + \alpha_2 \cdot f_2(x^k) \tag{20}$$

The results are shown in the image below. It can be clearly seen that our method is able to produce the concave Pareto front. On the other hand, linear scalarization is unable to find solutions on the concave part of the Pareto front(Boyd et al., 2004) and converges to one solution for all preferences. This example demonstrates that our method can certainly be extended to MOO with concave Pareto fronts.

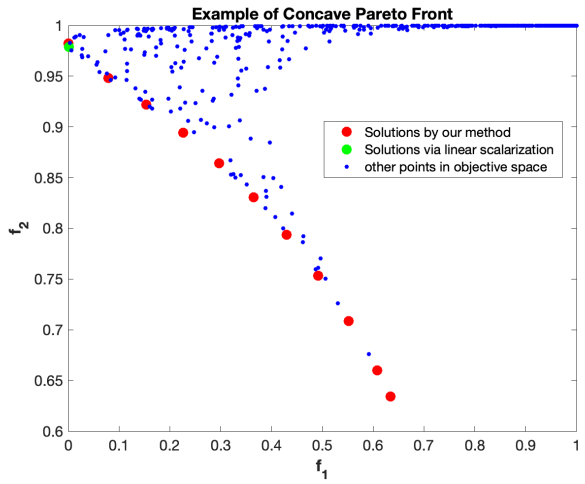


Figure 3: The above plot depicts the convergence of our method to Concave Pareto Front. Here red points are the results generated by our algorithm, green points are the solution from linear scalarization method and blue points represent the set of possible solutions in the objective space.

NETWORK ARCHITECTURE

PA-Net uses a modified architecture of TSP-Net Deudon et al. (2018). TSP-Net aims at finding tours with minimal path length. On the other hand, PA-Net finds a set of Pareto optimal tours that are dependent on multiple criterias. The architecture of the for both PA-Net and TSP-Net is shown in Fig-4.

TSP-Net: The input to the network is a batch of N-city TSP graphs $\Omega : \{s^1....s^B\}$. Each city, $a_i^b \in \mathbb{R}^p$ in the input graph, is encoded to a higher dimension embedding $d_i^b \in \mathbb{R}^d$ using a multi-headed attention encoder (Vaswani et al., 2017; Deudon et al., 2018). So each graph $s^b \in \Omega$ can be represented as $D^b : \{d_1^b...d_N^b\}$. This encoded graph sequence is then converted into a unified graph representation F_{ac}^b:

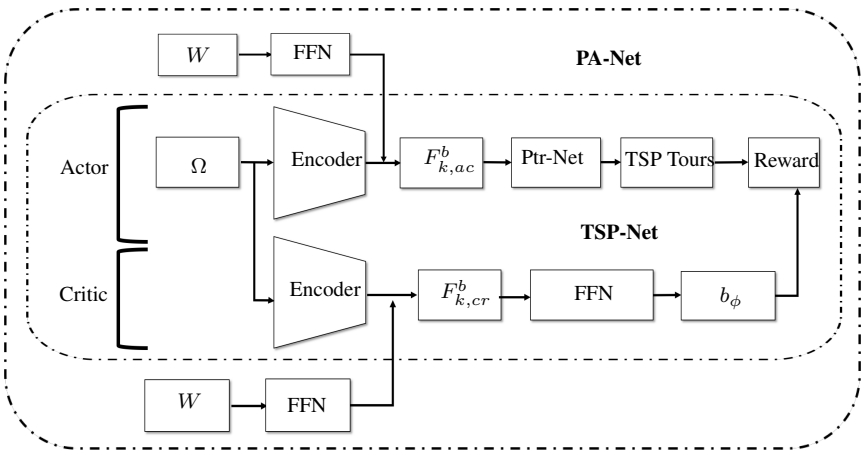
$$F_{ac}^b = W_p(\theta) \cdot D^b \tag{21}$$

Here, θ are the parameters of the actor network and W_p is a matrix that projects the encoded graph sequence to a unified representation. This unified graph representation is then used by pointer network to generate TSP tours for the complete batch Ω. The details of the pointing mechanism to generate a tour can be found in (Vinyals et al., 2015; Deudon et al., 2018). The gradient at training time is computed using REINFORCE. The critic network uses the same attention based encoder to generate a unified graph representation (F_{cr}^b), which is then used by a feedforward network to predict the baseline for training.

PA-Net: For PA-Net, the input is a batch of TSP tours Ω and a set of preference vectors $W : \{\vec{w}_1...\vec{w}_K\}$. Like before, each city is encoded to a higher dimension embedding d_i^b. Each $\vec{w}_k$ is encoded using a feed forward network to obtain a higher dimension embedding $h_k \in \mathbb{R}^d$. Now, the encoding of each city is combined with the k^{th} preference encoding to generate augmented embedding for the cities $d_{i,k}^b$:

$$d_{i,k}^b = d_i^b + h_k \tag{22}$$

From Eq-equation 22 augmented embeddings for all the cities in a graph are $D_k^b : \{d_{1,k}^b...d_{N,k}^b\}$. The unified graph representation $F_{k,ac}^b$, for G^b and k^{th} preference, is obtained using Eq-equation 21. Finally, a set of K tours for each G^b is generated by the pointer network. The gradient at training time is computed using Eq-equation 14. The critic uses the same encoder architecture to generate an augmented unified graph representation ($F_{k,cr}^b$), which is used by the feed-forward network (FFN) to predict the baseline.



Ω : Input Graphs
W : Input Preferences
b_ϕ : Baseline

$F_{k,ac}$, $F_{k,cr}$: Unified graph Representation
FFN : Feed Forward Network
Ptr-Net : Pointer Network

Figure 4: PA-Net uses an augmented version of TSP-Net. The input to the network is a set of TSP graphs Ω and a set of preference vectors W. The output is K TSP tours for each graph in Ω

PREFERENCE SELECTION

One of the key features of PA-Net is its ability to generate solutions for different preferences in the objective space. This is achieved by simultaneously training the network on a set of preferences W. Since each $\vec{w}_k \in W$ is a unit vector, it must satisfy $||\vec{w}_k||_2 = 1$. For training of 2-objective network, these preferences can be generated by sampling unit vectors from the unit circle for angles in $\{0°, 90°\}$. For objectives more than two, selecting preferences is not so straight forward. In this case, the preference $\vec{w}_k \in \mathbb{R}^p$ is generated using:

$$\vec{w}_k = \frac{1}{\sqrt{(a_k^1)^2 + ...(a_k^p)^2}} \cdot (a_k^1.....a_k^p)^\top, \tag{23}$$

where a_k^i is sampled from $\{0, 1\}$ $\forall i \in \{1...p\}$. So using Eq-equation 23 preference vectors for higher dimension optimization are generated. The network is trained on a fixed set of preferences. Towards the end of the training, we resample the set of preference vectors after every fixed number of iterations. This step improves the generalizability of the network.

PA-NET VS TSP-NET + LS

The base network used in PA-Net is TSP-Net. Here, we evaluate the effect of using linear scalrization in TSP-Net and compare it with PA-Net. For this, we train 40 separate networks of TSP-Net for different preferences. Just like PA-Net, each network of TSP-Net+LS is trained on a graph of size 120×4 with batch size 60 and total training iterations of 20000. We then compare Pareto fronts generated by TSP-Net+LS and PA-Net. It should be noted that preferences used for PA-Net are unit vectors. So for inference on PA-Net, normalized preferences of TSP-Net+LS is used. The results are reported in Table 5. It is clear that PA-Net significantly outperforms TSP-Net+LS both in terms of HV and training times. Prolonged training of TSP-Net+LS may improve its performance, but that would lead to greater training times.

Table 5: Hypervolume comparison for PA-Net and TSP-Net+LS

Cities	**Network type**	**No. of Parameters**
	HV (%)	HV (%)
40	74.8	53.6
200	82.7	47.12
500	86.8	47.5
10000	88.2	147.7
Training time (hrs)	~ 14	~ 39

ABLATION STUDIES

In order to understand the contribution of different parts of PA-Net a few ablation studies were performed on $2-$ objective MOTSP instances. Description for various studies performed are as follows:

- Ablation 1: For this study, PA-Net is trained without the feed forward network of preference encoder.
- Ablation 2: In this study, PA-Net is trained with only a single deep layer of the preference encoder. In the baseline network, we use 4 deep layers,
- Ablation 3: For this study, PA-Net is trained without the transformer based encoder. A single 1-D convolution layer is used to encode the input graph.

All of these networks are compared on the basis of HV and the total time taken to generate the Pareto front for $2-$ objective MOTSP where 100 preference vectors are used at inference time. The comparative results for different ablation studies are summarized in Tab. 6. Further, the number of trainable network parameters for each network are given in Table 7. The network trained for

Table 6: Quantitative comparison of Pareto front for 2-Objective MOTSP

	40-City		**200-City**		**500-City**		**1000-City**		Training time (hrs)
Algo.	HV (%)	Time (s)	HV (%)	Time (s)	HV (%)	Time (s)	HV (%)	Time (s)	
PA-2 (baseline)	**75.7**	1.66	**83**	6.66	**86.7**	16.5	**88.3**	34.4	~ 14
Ablation 1	39.1	1.48	25.6	5.5	25.0	14.5	24.6	29.8	~ 13
Ablation 2	75.4	1.53	82.6	5.7	86.3	14.3	87.8	29.6	~ 13
Ablation 3	**75.74**	**1.29**	82.8	**5.5**	86.6	**14.15**	**88.3**	**28.9**	~ 12

Table 7: Number of trainable parameters for each network in Ablation studies

Network type	**No. of Parameters**
Ablation 1	1454337
Ablation 2	1465617
Ablation 3	464129
PA-2 (baseline)	1554459

Ablation 1 has the worst performance. This indicates that encoder to learn representation of preferences plays a critical role in the performance of PA-Net. Interestingly enough, it seems like the choice of encoder for both preferences and the input TSP graph does not have much impact on the performance. This indicates that a relatively faster network can be obtained by using a relatively simpler choice of encoder. Although, the network in Ablation 3 has significantly lower number of trainable parameters, yet its training time is not significantly lower. This is primarily due to the fact that during the training, $K \times B$ TSP tours are generated in all the networks. This serves as the computation bottleneck for the network during the training. Hence, that's why no significant difference in training times are observed.

HYPERPARAMETER TUNING

In order to obtain the best performing network, we trained networks with different values of some key hyperparameters. These hyperparameters are:

- Number of preference vectors (K).
- Maximum value of lagrangian multipliers (λ_{max})
- Ascent rate of the lagrangian multiplier (α)

Table 8: Comparison of HV for different hyperparameter values for 2-objective MOSTP

		40-City	**200-City**	**500-City**	**1000-City**
Parameter	Value	HV (%)	HV (%)	HV (%)	HV (%)
K	20 (baseline)	**75.7**	**83**	**86.7**	**88.3**
	10	73.5	82.2	85.4	85.4
	5	74.8	78.1	81.1	82.3
λ_{max}	20 (baseline)	75.7	**83**	**86.7**	**88.3**
	10	75.6	82.6	86.3	88.0
	5	**75.9**	81.9	85.4	87.3
α	2.5×10^{-5} (baseline)	75.7	**83**	**86.7**	**88.3**
	2.5×10^{-3}	73.5	82	85.8	87.4
	1.25×10^{-5}	**76.1**	**83**	86.6	88.1
	5.0×10^{-5}	75.8	82.5	86.4	87.9
	2.5×10^{-8}	75	81.7	85.5	87.0

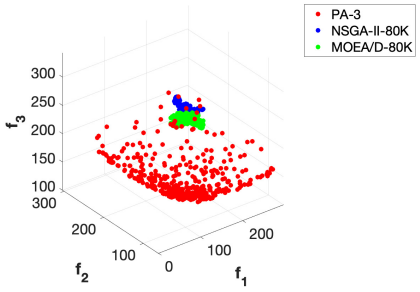
These networks trained on different values of the above-mentioned hyperparameters are compared on the basis of obtained HV for 2-objective MOTSP where 100 preference vectors were used at the inference time. These results are presented in Tab. 8. The following conclusions can be made:

- Higher values of K leads to better results. Intuitively, it makes sense because a higher number of preference vectors during training can help network learn better and generalize better. However, large K leads to longer training time. For example, training time of network with $K = 20$ is $\sim 14\,hrs$ and for $K = 5$ its $\sim 6hrs$.
- While varying λ_{max} mixed results were observed. For the most part, parameters in the baseline network gave a better performance,
- For ascent rate (α) it was observed that for most of the instances, extremely high values or extremly low values lead to sub-par performance.

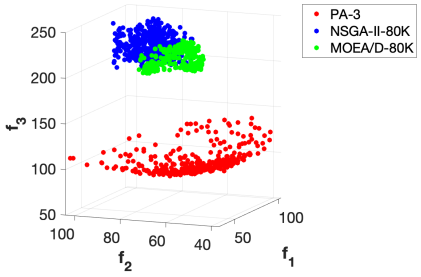
ADDITIONAL EXPERIMENTAL RESULTS

Euclidean 3-objective MOTSP: PA-Net was evaluated on another 3-objective Euclidean MOTSP problem. In this case, all three objectives are $\mathbb{L}_2$-norms. The generated Pareto fronts for 200 and 500 City MOTSP are shown in Fig-5(a)-(b). It can be clearly seen that PA-Net is able to generate a good approximation Pareto frontier. The 3-D Pareto Front can be visualized at: https://sites.google.com/view/pareto-approximate-net/home.

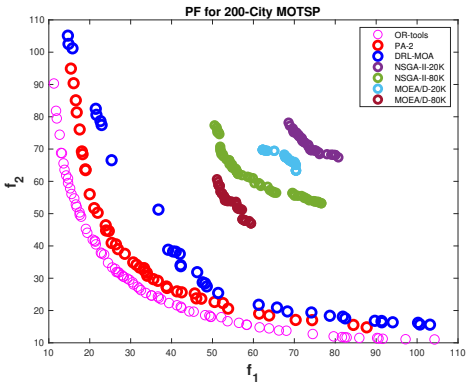
Euclidean 2-objective MOTSP: Results of 2-objective MOTSP (see Sec-4) for 200 and 1000 cities is shown in Fig-5(c)-(d)



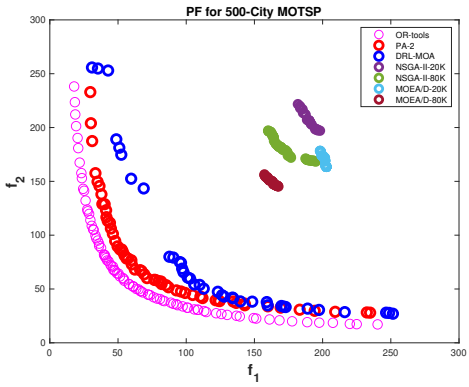
(a) 500 City MOTSP with 3 Euclidean objective



(b) 200 City MOTSP with 3 Eucledian objective



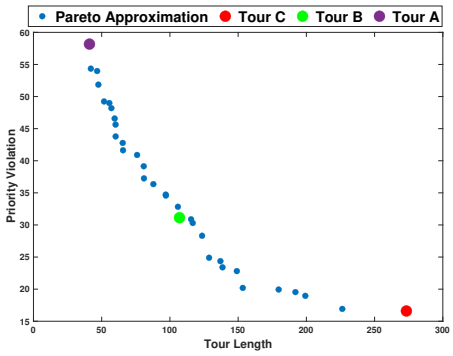
(c) 200 City MOTSP with 2-objective



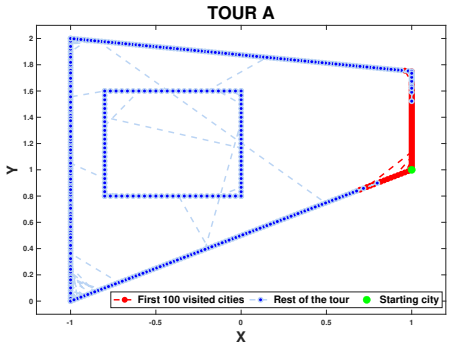
(d) 500 City MOTSP with 2-objective

Figure 5: Visualization of the dominant solutions for 3 and 2 objective MOTSP. It can be seen that our network (PA-2 and PA-3) generates significantly better objective values

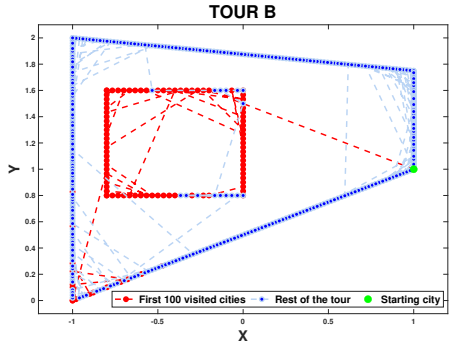
Application for Coverage Planning: This experiment represents a setup where a robot has to visit multiple locations in a building floor like a mall, airport etc (see Sec-4). The sequence of visiting these locations is dependent on the path length and adherence to a pre-computed priority. The visualization of three tours generated by PA-Net for this scenario is shown in Fig-6(b)-(d). The square section in the middle of the environment is given the highest priorities, and the rest of the depots are assigned priorities randomly. In each plot of the tour shown in Fig-6, the first 100 depots visited are marked in red. It can be seen that Tour C visits the middle section first and has the least priority violation. On the other hand, emphasis in Tour A is to minimize the total tour length.



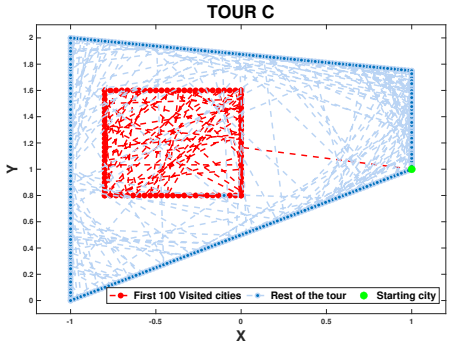
(a) Pareto front for 500 depot MOTSP for coverage planning



(b) Tour A



(c) Tour B



(d) Tour C

Figure 6: Visualization of different tours generated for the task of coverage planning. It can be seen that Tour A has the least tour length, which comes at a cost of high priority violation. Tour C has the smallest priority violation. Tour B is an intermediate tour between the other two tours.