Continuation Path Learning for Homotopy Optimization

Xi Lin [1] **Zhiyuan Yang** [1] **Xiaoyuan Zhang** [1] **Qingfu Zhang** [1]

Abstract

Homotopy optimization is a traditional method to deal with a complicated optimization problem by solving a sequence of easy-to-hard surrogate subproblems. However, this method can be very sensitive to the continuation schedule design and might lead to a suboptimal solution to the original problem. In addition, the intermediate solutions, often ignored by classic homotopy optimization, could be useful for many real-world applications. In this work, we propose a novel model-based approach to learn the whole continuation path for homotopy optimization, which contains infinite intermediate solutions for any surrogate subproblems. Rather than the classic unidirectional easy-to-hard optimization, our method can simultaneously optimize the original problem and all surrogate subproblems in a collaborative manner. The proposed model also supports real-time generation of any intermediate solution, which could be desirable for many applications. Experimental studies on different problems show that our proposed method can significantly improve the performance of homotopy optimization and provide extra helpful information to support better decision-making.

1. Introduction

Homotopy optimization (Blake & Zisserman, 1987; Yuille, 1989; Allgower & Georg, 1990), also called continuation optimization, is a general optimization strategy for solving complicated and highly non-convex optimization problems which can be found in many machine learning applications (Jain & Kar, 2017). This method first constructs a simple surrogate of the original optimization problem, and then gradually solves a sequence of easy-to-hard surrogates to approach the optimal solution of the original complicated problem. The simplest surrogate subproblem could be easily solved, and its solution will serve as a good initial one for the next subproblem. In this way, we can eventually find a good initial and then a (nearly) optimal solution for the original hard-to-solve optimization problem.

The idea of homotopy optimization is straightforward but it also suffers several drawbacks. First, the optimization performance could heavily depend on the continuation schedule of the surrogate subproblems, which is not easy to design for a new problem (Dunlavy & O'Leary, 2005). It also needs to iteratively solve each subproblem in sequence, which could lead to undesirable long run time in practice (Iwakiri et al., 2022). In addition, the existing homotopy optimization methods only focus on finding the final solution to the original problem. However, the intermediate solutions for homotopy surrogate subproblems could be useful for many real-world applications.

In this work, to tackle the drawbacks mentioned above, we propose a novel continuation path learning (CPL) method to find the whole solution path for homotopy optimization, which contains infinite solutions for all intermediate subproblems. The key idea is to build a learnable model that maps any valid continuation level to its corresponding solution, and then optimize all of them simultaneously in a collaborative manner. In this way, the whole path model can be learned in a single run without sensitive schedule design and iterative optimization. With the learned path model, we can easily generate the solution for any intermediate homotopy level, which could be useful for many applications. Our main contributions can be summarized as follows:

- We propose a novel model-based approach to learn the whole continuation path for homotopy optimization, which is significantly different from the existing methods that iteratively solve a sequence of finite subproblems.
- We develop an efficient learning method to train the path model concerning all homotopy levels simultaneously. The proposed model can generate solutions for any intermediate subproblem in real time, which is desirable for many real-world applications.
- We empirically demonstrate that our proposed CPL method can achieve promising performances on var-

[1]Department of Computer Science, City University of Hong Kong. Correspondence to: Xi Lin <xi.lin@my.cityu.edu.hk>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

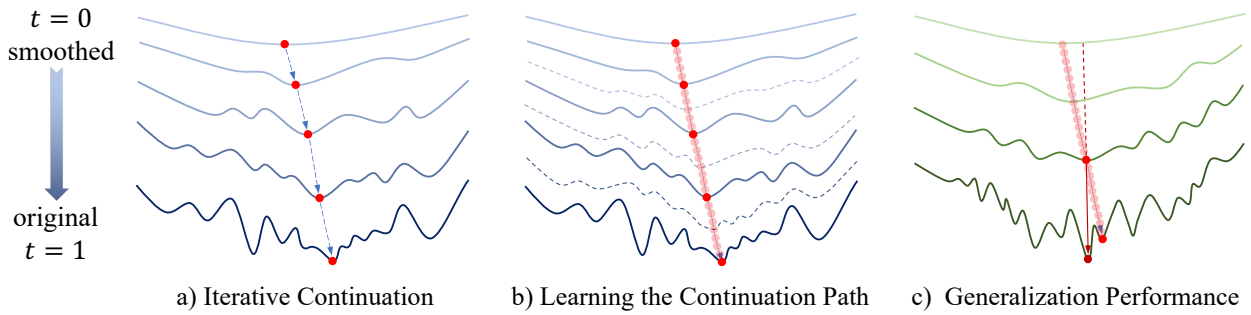


Figure 1. **Continuation Path Learning**: **a)** The classical homotopy optimization method sequentially solves a set of easy-to-hard smoothed subproblems, which helps find the optimal solution for the original problem. **b)** We propose to simultaneously learn the whole continuation path, which contains the intermediate solutions for all homotopy subproblems. **c)** The solutions for homotopy subproblems could have better generalization performance for learning-based problems.

ious problems, including non-convex optimization, noisy regression, and neural combinatorial optimization. [1]

2. Related Work

Homotopy Optimization, also called continuation or graduated optimization method (Blake & Zisserman, 1987; Yuille, 1989; Allgower & Georg, 1990), is a general optimization strategy for solving non-convex optimization problems. This method sequentially constructs and solves a set of smoothed subproblems that gradually deform from an easy-to-solve problem to the original complicated problem as shown in Figure 1(a). It would help to find a better solution for the original non-convex problem (Wu, 1996; Dunlavy & O'Leary, 2005). This method has also been widely used for solving nonlinear equations (Eaves, 1972; Wasserstrom, 1973; Allgower & Georg, 1990), and it is closely related to simulated annealing for optimization (Kirkpatrick et al., 1983; Van Laarhoven & Aarts, 1987; Ingber, 1993).

Homotopy Optimization in Machine Learning The homotopy optimization methods have been widely used in different machine learning applications over the past three decades, such as for computer vision (Terzopoulos, 1988; Gold et al., 1994; Brox & Malik, 2010; Hruby et al., 2022), statistical learning (Chapelle et al., 2006; Kim & De la Torre, 2010), curriculum learning (Bengio, 2009; Bengio et al., 2009; Kumar et al., 2010; Graves et al., 2017), and efficient model training (Chaudhari et al., 2016; Gargiani et al., 2020; Guo et al., 2020). A few works have been proposed to study its theoretical property in different settings (Mobahi & Fisher III, 2015b;a; Hazan et al., 2016; Anandkumar et al., 2017; Iwakiri et al., 2022). Although the homotopy optimization method can usually help to find a better solution, it might suffer from a long run time due to the iterative optimization structure. Recently, Iwakiri et al. (2022) have proposed a novel single loop algorithm for fast Gaussian homotopy optimization.

Most homotopy optimization methods only care about the final solution for the original problem, but the intermediate optimal solutions could also be useful for flexible decision-making. A few works have been proposed to find a single solution for a specific smooth intermediate subproblem for better generalization performance (Chaudhari et al., 2016; Gulcehre et al., 2017). In statistical learning, efficient methods have been proposed to find the whole set of finite tuning points that fully characterize the homotopy path for LASSO (Efron et al., 2004; Rosset & Zhu, 2007; Tibshirani & Taylor, 2011) and SVM (Hastie et al., 2004) by leveraging the specific piece-wise linear structure. However, these methods do not work for general homotopy optimization problems.

Model-based Optimization Many model-based methods have been proposed to improve different optimization algorithms' performance. Bayesian optimization builds a surrogate model to approximate the unknown black-box optimization problem and uses it to guide the optimization process (Shahriari et al., 2016; Garnett, 2022). Latent space modeling (Gómez-Bombarelli et al., 2018; Tripp et al., 2020) is another powerful approach for reconstructing the original complicated optimization problem into a much easier form to solve. It is also possible to accelerate the optimization algorithm by learning the problem structure (Sener & Koltun, 2020) or dividing the search space (Wang et al., 2020; Eriksson et al., 2019). Recently, a few model-based approaches have been proposed to learn the Pareto set for multi-objective optimization problems (Yang et al., 2019; Dosovitskiy & Djolonga, 2020; Lin et al., 2020; Navon et al., 2021; Lin et al., 2022a;b). In this work, we propose a novel method to learn the whole continuation path for homotopy optimization as shown in Figure 1(b), and use it to improve the optimization performance for different applications.

[1] The source code can be found in `https://github.com/Xi-L/CPL`.

3. Homotopy Optimization

In this section, we introduce the classical homotopy optimization method and a recently proposed single-loop Gaussian homotopy algorithm.

3.1. Classical Homotopy Optimization

We are interested in the following minimization problem:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ is the decision variable, and $f : \mathcal{X} \to \mathbb{R}$ is the objective function to minimize. The objective function $f(\boldsymbol{x})$ could be highly non-convex and has a complicated optimization landscape. Therefore, it cannot be easily optimized with a simple local minimization method such as gradient descent.

To tackle this problem, homotopy optimization method (Allgower & Georg, 1990; Dunlavy & O'Leary, 2005) considers a family of function $H : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$ parameterized by the continuation level $t \in \mathcal{T} = [0, 1]$ such that:

$$H(\boldsymbol{x}, t = 0) = g(\boldsymbol{x}) \quad H(\boldsymbol{x}, t = 1) = f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{X}, \tag{2}$$

where $g : \mathcal{X} \to \mathbb{R}$ is another easy-to-optimize objective function on the same decision space $\mathcal{X}$. The function $H(\boldsymbol{x}, t)$ is also called a homotopy that gradually transforms $g(\boldsymbol{x})$ to $f(\boldsymbol{x})$ by increasing t from 0 to 1. An illustration of the continuation function can be found in Figure 1(a).

The key idea of homotopy optimization is to define a suitable continuation function $H(\boldsymbol{x}, t)$ such that the minimizer for $H(\boldsymbol{x}, 0) = g(\boldsymbol{x})$ is already known or easy to find, and the $H(\boldsymbol{x}, t)$ with $t = 0 \to 1$ be a sequence of smoothed functions transforming from $g(\boldsymbol{x})$ to the target objective function $f(\boldsymbol{x})$. Rather than directly optimizing the complicated target function $f(\boldsymbol{x})$, we can progressively solve a sequence of coarse-to-fine smoothed optimization subproblems from $H(\boldsymbol{x}, 0)$ to $H(\boldsymbol{x}, 1)$ with a warm start from previously obtained solution as shown in **Algorithm 1**. In this way, we can find a better solution for the target objective function $H(\boldsymbol{x}, 1) = f(\boldsymbol{x})$.

In practice, the performance of homotopy optimization heavily depends on two crucial components:

- A proper design of the initial and continuation function for the given problem;
- A suitable schedule to progressively solve the sequence of easy-to-hard subproblems.

However, there is no clear and principled guideline for the continuation construction (Mobahi & Fisher III, 2015a). The homotopy optimization process could be time-consuming since we have to run a local search algorithm to find $\boldsymbol{x}_k$ for each subproblem (Iwakiri et al., 2022). The continuation and schedule design will become much more complicated if we want to find a solution for a specific homotopy level unknown in advance, such as for better generalization (Chaudhari et al., 2016; Gulcehre et al., 2017).

Algorithm 1 Classical Homotopy Optimization Algorithm

Input: continuation function $H(\boldsymbol{x}, t)$, a predefined sequence $0 = t_0 < t_1 < \ldots t_K = 1$
$\boldsymbol{x}_0$ = a minimizer of $H(\boldsymbol{x}, t_0)$
for $k = 1$ to K **do**
  $\boldsymbol{x}_k$ = local minimizer of $H(\boldsymbol{x}, t_k)$, initialized at $\boldsymbol{x}_{k-1}$
end for
Output: $\boldsymbol{x}_K$

3.2. Single Loop Gaussian Homotopy Algorithm

Algorithm 2 Single Loop Gaussian Homotopy Algorithm

Input: Gaussian homotopy function $GH(\boldsymbol{x}, t)$, initial solution $\boldsymbol{x}_0$, initial homotopy level t_0
for $k = 1$ to K **do**
  $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - \eta_1 \nabla_{\boldsymbol{x}} GH(\boldsymbol{x}_{k-1}, t_{k-1})$
  (optional) query $G_t = \frac{\partial GH(\boldsymbol{x}_{k-1}, t_{k-1})}{\partial t}$
  update $t_k =$

$$\begin{cases} \gamma t_{k-1} & \text{(SLGH}_r\text{)} \\ \max\{0, \min\{t_{k-1} - \eta_2 G_t, \gamma t_{k-1}\}\} & \text{(SLGH}_d\text{)} \end{cases}$$

end for
Output: $\boldsymbol{x}_K$

To alleviate the time-consuming local optimization at each homotopy iteration, Iwakiri et al. (2022) recently proposed a novel single loop algorithm for the popular Gaussian homotopy method (Blake & Zisserman, 1987).

The Gaussian homotopy function $GH(\boldsymbol{x}, t)$ with $t \in [0, 1]$ for $f(\boldsymbol{x})$ can be defined as:

$$\begin{aligned} GH(\boldsymbol{x}, t) &= \mathbb{E}_{\boldsymbol{u}\sim N(0,I_d)}[f(\boldsymbol{x} + \beta(1-t)\boldsymbol{u})] \\ &= \int f(\boldsymbol{x} + \beta(1-t)\boldsymbol{y})k(\boldsymbol{y})d\boldsymbol{y}, \end{aligned} \tag{3}$$

where parameter $\beta > 0$ controls the max range of homotopy effect, $N(0, I_d)$ is the d-dimensional standard Gaussian distribution, and $k(\boldsymbol{y}) = (2\pi)^{-d/2} \exp(-||\boldsymbol{y}||^2/2)$ is the Gaussian kernel. Instead of iteratively optimizing $GH(\boldsymbol{x}, t)$ with a sequence of t as in previous works (Wu, 1996; Mobahi & Fisher III, 2015b;a; Hazan et al., 2016),

Iwakiri et al. (2022) proposed to directly optimize:

$$\min_{\boldsymbol{x}\in\mathcal{X},t\in\mathcal{T}} GH(\boldsymbol{x},t) \tag{4}$$

with respect to both $\boldsymbol{x}$ and t at the same time. By leveraging the theoretical properties of the heat equation (Widder, 1976) and partial differential equation (Evans, 2010), they also showed that the Gaussian homotopy function $GH(\boldsymbol{x},t)$ will be always optimized at $(\boldsymbol{x}^*,1)$ where $\boldsymbol{x}^*$ is the optimal solution for $f(\boldsymbol{x})$. This property validates the approach to optimize $GH(\boldsymbol{x},t)$ with respect to t.

The single loop Gaussian homotopy (SLGH) optimization algorithm is shown in **Algorithm 2**. At each iteration, the decision variable $\boldsymbol{x}_k$ is updated by gradient descent, and the homotopy level t_k is updated by either a fixed ratio decreasing rule (e.g., SLGH$_r$ with $\gamma = 0.999$) or a derivative update rule (e.g., SLGH$_d$). This algorithm could be faster than the classical homotopy optimization algorithm with a double loop structure (Iwakiri et al., 2022). However, the performance of SLGH is still sensitive to the homotopy schedule (e.g., the setting of γ). It might not work for a general homotopy optimization problem other than Gaussian homotopy, and can not easily obtain the solutions for any intermediate subproblem or the whole homotopy path.

4. Continuation Path Learning

4.1. Why Continuation Path Learning

Both classical homotopy optimization algorithms and the recently proposed SLGH method only focus on finding a single solution for the original optimization problem. In many real-world applications, an intermediate solution $\boldsymbol{x}^{t^*}$ for a smooth homotopy subproblem $H(\boldsymbol{x}^{t^*},t^*)$ could be desirable such for robust generalization performance (Chaudhari et al., 2016; Gulcehre et al., 2017).Finding a solution with a proper regularization v.s. performance trade-off is also an important issue in machine learning (Efron et al., 2004; Hastie et al., 2004). However, the optimal homotopy level t^* is usually unknown in advance.

In this work, we propose a novel model-based approach to learn the whole continuation path that contains solutions for all gradually smooth homotopy subproblems. With the learned path model, decision-makers can easily select their preferred solution(s) on the solution path as shown in Figure 2. In addition, even if the goal is to find a single solution with a given homotopy level (e.g., $t = 1$ for the original problem), CPL can also efficiently generate a good initial solution by collaboratively learning all subproblems together. It is a strong alternative to the sequential and unidirectional information passing in classical homotopy optimization.

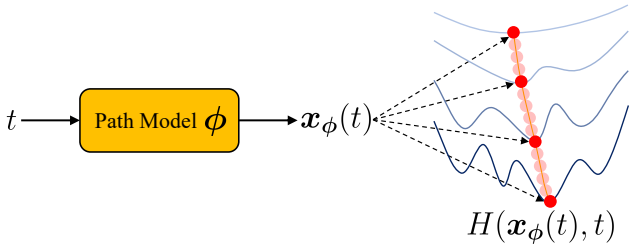


Figure 2. **Continuation Path Model** takes any valid homotopy level t as input, and generate its corresponding solution $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$ on the continuation path. Decision-makers can easily obtain any predictive homotopy solution by adjusting the input level t.

4.2. Continuation Path Model

For the homotopy function $H(\boldsymbol{x},t)$, the number of continuation levels $t \in \mathcal{T}$ and their corresponding subproblems could be infinite. Let $\boldsymbol{x}_0^*$ be the minimizer for $H(\boldsymbol{x},0) = g(\boldsymbol{x})$, we can define a path $\boldsymbol{x}^*(t)$ continuous in t such that $\boldsymbol{x}^*(0) = \boldsymbol{x}_0^*$ and $\nabla H(\boldsymbol{x}^*(t),t) = 0$ for all $t \in \mathcal{T} = [0,1]$ (Wu, 1996). This path simply goes through a set of stationary points for $H(\boldsymbol{x},t)$ from $\boldsymbol{x}_0^*$ with gradually increasing t. The homotopy optimization algorithms trace the solutions from $\boldsymbol{x}^*(0)$ to $\boldsymbol{x}^*(1)$. In practice, the solution $\boldsymbol{x}^*(1)$ could be a good local minimizer for $H(\boldsymbol{x},1) = f(\boldsymbol{x})$ if not the global one (Mobahi & Fisher III, 2015a;b). The existence and uniqueness of this path can be guaranteed for the Gaussian homotopy function (3) under mild conditions:

Theorem 4.1 (Existence of Continuation Path (Wu, 1996))**.** *Let f be a well-behaved function and $H(\boldsymbol{x},t)$ be its Gaussian homotopy function (3). Then for any stationary point x_0 of $H(\boldsymbol{x},0)$, there is a continuous and differentiable curve $\boldsymbol{x}^*(t)$ on $t \in \mathcal{T} = [0,1]$ such that $\boldsymbol{x}^*(0) = x_0$ and $x^*(t)$ is a stationary solution of $H(\boldsymbol{x},t), \forall t \in \mathcal{T}$.*

To be well-behaved, a sufficient condition is that the function f is twice continuously differentiable while f and its derivatives should all be integrable for the Gaussian homotopy function (3) (Wu, 1996). We assume such continuation path $\boldsymbol{x}^*(t)$ always exists in this work.

According to the definition, $\boldsymbol{x}^*(t)$ is a continuous curve that contains solutions for all (infinite) homotopy levels $t \in \mathcal{T}$. The discrete set of stationary solutions obtained by a classical homotopy optimization $\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_K\} = \{\boldsymbol{x}^*(t_1),\boldsymbol{x}^*(t_2),\ldots,\boldsymbol{x}^*(t_K)\}$ is a finite subset on the solution path $\{\boldsymbol{x}^*(t)|t \in \mathcal{T}\}$. In this work, we propose to build a model $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$ with learnable parameter $\boldsymbol{\phi}$ to approximate the whole continuation path $\boldsymbol{x}^*(t)$. Our goal is to find the optimal $\boldsymbol{\phi}^*$ such that:

$$\boldsymbol{x}_{\boldsymbol{\phi}^*}(t) = \boldsymbol{x}^*(t) = \arg\min_{\boldsymbol{x}} H(\boldsymbol{x},t), \forall t \in \mathcal{T}. \tag{5}$$

As shown in Figure 2, the continuation path model maps any valid homotopy level t to its corresponding solution

$\boldsymbol{x}_{\boldsymbol{\phi}}(t)$. With the ideal model parameters $\boldsymbol{\phi}^*$, the output $\boldsymbol{x}_{\boldsymbol{\phi}^*}(t)$ should be the optimal solutions for each intermediate subproblem $H(\boldsymbol{x}, t)$, and hence $\boldsymbol{x}_{\boldsymbol{\phi}^*}(t)$ well approximates the continuation path $\boldsymbol{x}^*(t)$. Once such a model is obtained, we can easily get the corresponding solution $\boldsymbol{x}_{\phi^*}(t') = \arg\min_{\boldsymbol{x}} H(\boldsymbol{x}, t')$ for any specific continuation parameter t'. In this work, we set $\boldsymbol{x}_{\phi}(t)$ as a neural network model and $\boldsymbol{\phi}$ is its model parameters.

4.3. Learning the Continuation Path

Once we have the continuation path model $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$, the next step is to find the optimal parameters $\boldsymbol{\phi}^*$ with respect to all homotopy level $t \in \mathcal{T}$. Since the number of homotopy levels is infinite, our goal is to optimize the following expectation:

$$\min_{\boldsymbol{\phi}} \mathbb{E}_t H(\boldsymbol{x}_{\boldsymbol{\phi}}(t), t), \tag{6}$$

where each term $H(\boldsymbol{x}_{\boldsymbol{\phi}}(t), t)$ is a composition of continuation path model $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$ and the homotopy function $H(\boldsymbol{x}, t)$. In this way, we reformulate the classic unidirectional homotopy optimization problem (e.g., **Algorithm 1**) into the single loop model training problem (6) that simultaneously learn the whole continuation path. *It should be noticed that our method changes the optimization variables from the original decision variable $\boldsymbol{x}$ to model parameter $\boldsymbol{\phi}$.*

It is difficult to directly optimize the problem (6) since the expectation term could be hard to compute in most cases. In this work, we propose to learn the model parameters with stochastic gradient descent as shown in **Algorithm 3**. At each step, we optimize the following stochastic optimization problem with Monte Carlo sampling:

$$\min_{\boldsymbol{\phi}} \frac{1}{M} \sum_{m=1}^{M} H(\boldsymbol{x}_{\boldsymbol{\phi}}(t_m), t_m), \quad \{t_m\}_{m=1}^{M} \sim P_{\mathcal{T}}, \tag{7}$$

where $\{t_1, \ldots, t_M\}$ are M independent identically distributed (i.i.d.) samples from distribution $P_{\mathcal{T}}$. Without any prior knowledge, we can simply set $P_{\mathcal{T}}$ to be a uniform distribution on $\mathcal{T}$. It is also possible to use other distributions or further adaptively adjust the distribution to incorporate extra information along the optimization process.

A crucial step of the proposed method is to find a valid gradient direction to update the model parameters at each iteration. We can decompose the gradient $\nabla_{\boldsymbol{\phi}} H(\boldsymbol{x}_{\boldsymbol{\phi}}(t), t)$ with the chain rule:

$$\nabla_{\boldsymbol{\phi}} H(\boldsymbol{x}_{\boldsymbol{\phi}}(t), t) = \frac{\partial \boldsymbol{x}_{\boldsymbol{\phi}}(t)}{\partial \boldsymbol{\phi}} \cdot \nabla_{\boldsymbol{x}} H(\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{\phi}}(t), t), \tag{8}$$

where $\frac{\partial \boldsymbol{x}_{\boldsymbol{\phi}}(t)}{\partial \boldsymbol{\phi}}$ is the Jacobian matrix of the path model with output vector $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$, and $\nabla_{\boldsymbol{x}} H(\boldsymbol{x}, t)$ is the gradient of the homotopy function with respect to decision variables $\boldsymbol{x}$. In this work, since the path model is a neural network, the Jacobian matrix $\frac{\partial \boldsymbol{x}_{\boldsymbol{\phi}}(t)}{\partial \boldsymbol{\phi}}$ can be easily calculated with back-propagation. If the homotopy function is also differentiable with a known gradient formulation $\nabla_{\boldsymbol{x}} H(\boldsymbol{x}, t)$, we can use standard gradient descent to optimize the model parameters.

Algorithm 3 Gradient-based Continuation Path Learning

Input: continuation function $H(\boldsymbol{x}, t)$, a path model $\boldsymbol{x}_{\phi}(t)$ with learnable parameters $\boldsymbol{\phi}$
for $i = 1$ to I **do**
  randomly sample a set of $\{t_m\}_{m=1}^{M} \sim P_{\mathcal{T}}$
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \frac{\eta}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\phi}} H(\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{\phi}}(t_m), t_m)$
end for
(optional) $\boldsymbol{x}^{t'}$ = local minimizer of $H(\boldsymbol{x}, t')$, initialized at $\boldsymbol{x}_{\boldsymbol{\phi}}(t')$ with chosen homotopy level t'
Output: path model $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$

In many real-world applications, however, the gradient of the homotopy function could be unknown or hard to compute (Iwakiri et al., 2022). In these cases, we can use a zeroth-order optimization (also called derivative-free optimization) method (Duchi et al., 2015; Nesterov & Spokoiny, 2017) with approximate gradients for model training. For a general homotopy function, we can adopt a simple evolutionary strategy (ES) (Hansen & Ostermeier, 2001; Beyer & Schwefel, 2002) to approximate the gradient:

$$\overline{\nabla}_{\boldsymbol{x}} H(\boldsymbol{x}, t) = \frac{1}{\sigma K} \sum_{k=1}^{K} (H(\boldsymbol{x} + \sigma \boldsymbol{u}^{(k)}, t) - H(\boldsymbol{x}, t)) \boldsymbol{u}^{(k)}, \tag{9}$$

where $\{\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(K)}\}$ are K i.i.d. d-dimensional Gaussian vectors sampled from $N(0, I_d)$ and σ is a fixed control parameter. This simple gradient estimation is also closely related to Gaussian smoothing (Nesterov & Spokoiny, 2017; Gao & Sener, 2022).

For the Gaussian homotopy function (3), according to (Nesterov & Spokoiny, 2017), its gradient can be written as:

$$\begin{aligned} &\nabla_{\boldsymbol{x}} GH(\boldsymbol{x}, t) \\ &= \frac{1}{\beta(1-t)} \mathbb{E}_{\boldsymbol{u} \sim N(0, I_d)} ([f(\boldsymbol{x} + \beta(1-t)\boldsymbol{u}) - f(\boldsymbol{x})]\boldsymbol{u}). \end{aligned} \tag{10}$$

Therefore, its gradient can be approximated by:

$$\begin{aligned} &\overline{\nabla}_{\boldsymbol{x}} GH(\boldsymbol{x}, t) \\ &= \frac{1}{\beta(1-t)K} \sum_{k=1}^{K} (f(\boldsymbol{x} + \beta(1-t)\boldsymbol{u}^{(k)}) - f(\boldsymbol{x}))\boldsymbol{u}^{(k)}, \end{aligned} \tag{11}$$

where $\{\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(K)}\} \sim N(0, I_d)$ are K i.i.d. Gaussian vectors as in the ES approximate gradient (9), while we now only query the value for the original function f but not the homotopy function $H(\boldsymbol{x}, t)$. A similar zeroth-order approximation with batch size $K = 1$ has been used and analyzed in Iwakiri et al. (2022) for the SLGH algorithm.

4.4. Optional Local Search

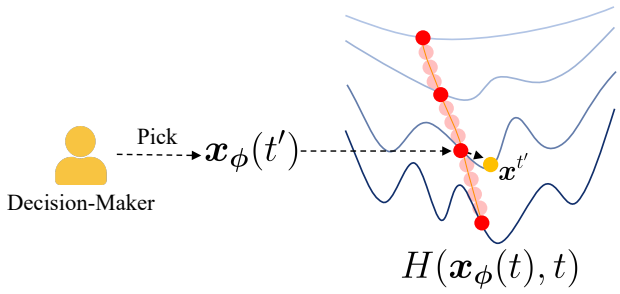


Figure 3. **Optional Local Search:** Decision-makers can easily pick their favorite solutions from the learned continuation path. An optional local search can help to find a better solution if needed.

The previous subsections mainly focus on learning the whole continuation path. In some applications, the decision-maker might only be interested in a single solution, such as $\boldsymbol{x}_{\boldsymbol{\phi}}(t=1)$ for the original optimization problem. By learning the whole continuation path $\boldsymbol{x}_{\boldsymbol{\phi}}(t)$ for all homotopy levels together, CPL actually exchanges the information among different homotopy subproblems simultaneously via the path model. In this case, our learned continuation path model can act as a warm start for any homotopy subproblem, which is more flexible than the gradual unidirectional information passing in classical homotopy optimization. An optional fast local search could help to find a better solution as in **Algorithm 3** and Figure 3. This step is equal to the final iteration of the classical homotopy optimization algorithm. In the next section, we empirically show that CPL can indeed find better initial solutions.

5. Bridging Homotopy Optimization and Parametric Optimization

Our proposed continuation path learning approach indeed provides a novel view to bridge homotopy optimization and parametric optimization. We discuss several interesting connections in this section.

5.1. CPL as Parametric Optimization

A general parametric optimization problem is defined as:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}, \boldsymbol{\beta}), \tag{12}$$

where $\boldsymbol{x} \in \mathcal{X}$ is the decision variable, $\boldsymbol{\beta} \in \mathcal{B}$ is the problem parameter (also called the context), and $f : \mathcal{X} \times \mathcal{B} \to \mathbb{R}$. Classical works on parametric optimization mainly focus on the sensitivity of objective value to the problem parameter (Bank et al., 1983; Bonnans & Shapiro, 2013; Still, 2018). A typical metric is the value function

$$v(\boldsymbol{\beta}) = \min_{x} f(\boldsymbol{x}, \boldsymbol{\beta}) \tag{13}$$

that describes the change of optimal value with respect to the problem parameter $\boldsymbol{\beta}$.

In our work, we propose to build a model to approximate the whole continuation path $\boldsymbol{x}_{\boldsymbol{\phi}^*}(t) = \boldsymbol{x}^*(t) = \arg\min_{\boldsymbol{x}} H(\boldsymbol{x}, t)$ with every homotopy level t. If we treat the homotopy level t as the problem parameter, we can define the value function for homotopy optimization

$$v(t) = \min_{\boldsymbol{x}} H(\boldsymbol{x}, t) = H(\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{\phi}^*}(t), t) \tag{14}$$

for all valid t. With CPL's model-based reformulation, we can now use the well-studied parametric optimization approach as a novel view to analyze homotopy optimization methods.

In addition, the classic value function approach mainly addresses the change of optimal value $v(\boldsymbol{\beta})$ with respect to the parameter $\boldsymbol{\beta}$. The direct differentiation of $v(\boldsymbol{\beta})$ could be difficult since the optimal solution $\boldsymbol{x}$ is often unavailable (Mehmood & Ochs, 2020; 2021). By modeling $\boldsymbol{x}^*(t) = x_{\boldsymbol{\phi}^*}(t)$, the homotopy objective $v(t) = H(\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{\phi}^*}(t), t)$ can be easily differentiated and optimized by gradient-based optimization methods. The solution model $x_{\boldsymbol{\phi}}(t)$ (in addition to the objective value) may also provide useful information to support decision-making. It could be interesting to extend the solution model method for a general value function approach.

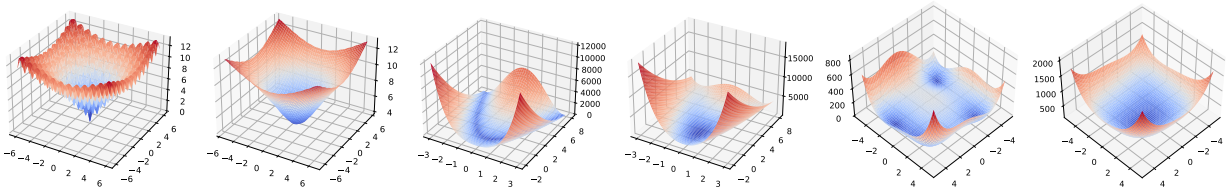
5.2. Connection to Amortized Optimization

There is an exciting and important research direction on learning to optimize (or called Amortized Optimization) (Amos, 2022; Chen et al., 2022), which focuses on making implicit or explicit inferences from a given problem context to its solution. Some recent works directly predict the optimal solution from problem parameters with specific structures (Liu et al., 2022). Conceptually, they are similar to the value function approach but also with a solution model.

A recent work (Li et al., 2023) proposes a classic iterative homotopy optimization method to accelerate the learning-to-optimize approach. Our proposed CPL method can be useful to further improve its performance. If we treat the homotopy level as an additional problem parameter, it is possible to learn the continuation path for a set of problems via a single model. The viewpoint of the value function approach may also provide useful insight for designing better methods.

5.3. Problem Reformulation and Difficulty

Our proposed CPL approach reformulates the original optimization problem into continuation model training which might have more parameters to optimize, but we believe it can deal with the homotopy optimization more easily. First of all, there could be infinite homotopy levels and corre-



(a) Ackley (b) GH Ackley (c) Rosenbrock (d) GH Rosenbrock (e) Himmelblau (f) GH Himmelblau

Figure 4. The original and Gaussian homotopy (GH) versions of the Ackley (a,b), Ronsenbrock (c,d), and Himmelblau (e,f) optimization problems. The original optimization problems are all non-convex, and the GH surrogate problems are much smoother and easier to solve.

Table 1. Results of gradient descent algorithm, Gaussian homotopy optimization algorithms, and our proposed continuation path learning (CPL) algorithm on three widely-used optimization problems. The optimal value is 0 for all problems. CPL can obtain the best solutions for all problems with the same number of function evaluations.

Algorithms		Ackley	Rosenbrock	Himmelblau
Gradient Descent	GD	12.63	0.2840	1.6×10^{-4}
Homotopy Optimization	GradOpt($\gamma = 0.5$)	0.014	0.0336	14.14
	GradOpt($\gamma = 0.8$)	0.081	0.0370	80.51
	SLGH$_r$($\gamma = 0.995$)	6.650	0.0327	6.9×10^{-5}
	SLGH$_r$($\gamma = 0.995$)	0.017	0.0419	0.21
Continuation Path Learning	CPL(95% path model training)	0.022	0.0421	1.7×10^{-3}
	CPL(path model + local search)	**0.006**	**0.0018**	$\mathbf{2.3 \times 10^{-6}}$

sponding intermediate solutions for the original problem, which is hard to handle using classic iterative optimization approaches. Our proposed CPL method can learn the whole continuation path via a single model, which is a novel and principled way to deal with this problem. In addition, the existing homotopy optimization methods could be sensitive to the progressive schedule design, and also suffer from high computational overheads due to the iterative optimization structure (Iwakiri et al., 2022). In contrast, our model-based CPL approach can be easily trained by a gradient-based optimization algorithm and obtain promising results. The reason why a large deep neural network can be easily trained by stochastic gradient descent is still an open research question. For the problem transformation in CPL, some findings from the smooth parametrization work such as Levin et al. (2022) would be useful for further studies.

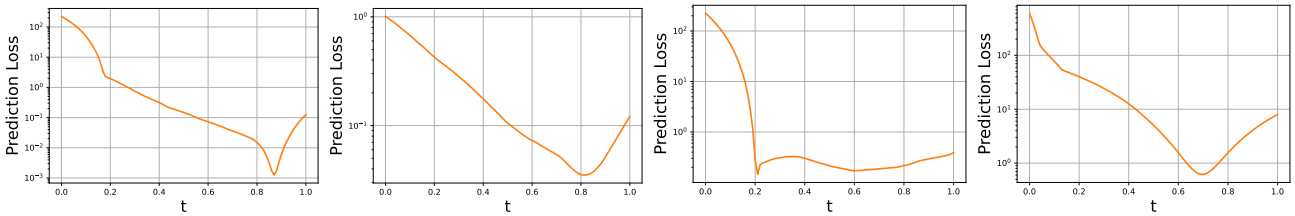
6. Experimental Studies

In this section, we empirically evaluate different aspects of our proposed continuation path learning (CPL) method for solving non-convex optimization, noisy regression, and neural combinatorial optimization problem. Due to the page limit, we only report the main results in this section. Detailed experimental settings, extra experimental results, and more discussions for each problem can be found in Appendix A,B, and C respectively.

6.1. Non-convex Optimization

We first test CPL's performance on three widely-used synthetic test benchmark problems, namely the Ackley function (Ackley, 1987), the Rosenbrock function (Rosenbrock, 1960), and the Himmelblau function (Himmelblau et al., 1972). The original and Gaussian homotopy versions of these functions are shown in Figure 4. The original optimization functions are non-convex and hence hard to be directly optimized by simple gradient descent algorithms. In contrast, their Gaussian homotopy versions are much smoother and easier to optimize.

We mainly follow the experimental setting from Iwakiri et al. (2022), and compare CPL with a simple gradient descent algorithm (GD), a classical Gaussian optimization algorithm (GradOpt) (Hazan et al., 2016) with two smoothing parameters ($\gamma = 0.5$ and 0.8), and the recently proposed single loop Gaussian homotopy algorithm with fixed ratio update (SLGH$_r$) (Iwakiri et al., 2022) with $\gamma = 0.995$ and 0.999. The total numbers of function evaluations are $1,000$, $20,000$, and $2,000$ for the Ackley, Rosenbrock, and Himmelblau optimization problem respectively. For CPL, since the goal is to optimize the original optimization problem, we use 95% function evaluations for path model training and the rest 5% for gradient-based local search with initial solution $\boldsymbol{x}_{\boldsymbol{\phi}}(t = 1)$. In other words, CPL has the same number of function evaluations as other methods.



(a) F1 (b) F2 (c) F3 (d) F4

Figure 5. Prediction performance v.s. homotopy level t on four different noisy nonlinear regression problems. CPL can successfully learn the whole continuation path for all problems. With the learned path, the decision-maker can easily locate the optimal homotopy level t with the minimum prediction loss for each problem.

According to the results reported in Table 1, the classic homotopy optimization methods are sensitive to the scheduling design and hyperparameters, which would be hard to tune for a new problem. Indeed, all the homotopy optimization methods are carefully fine-tuned as in Iwakiri et al. (2022), but no single method can consistently achieve good performance on all problems. In contrast, our proposed CPL method with only the model training step (95% evaluations) can already generate promising $\boldsymbol{x}_{\boldsymbol{\phi}}(t=1)$ solutions that have similar performances with other homotopy optimization algorithms. With extra gradient-based local search (the rest 5% evaluations), CPL can achieve significantly better solutions for all test problems. These results confirm that learning the whole continuation path in a collaborative manner with knowledge transfer can be helpful for solving the original complicated problem. It is a strong alternative to the classical homotopy optimization methods.

6.2. Noisy Regression

In this subsection, we consider the following noisy nonlinear regression problem:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} t||\hat{\boldsymbol{y}} - \psi(\hat{\boldsymbol{X}})\boldsymbol{\alpha}||_2^2 + (1-t)||\boldsymbol{\alpha}||_2, \tag{15}$$

where $\hat{\boldsymbol{X}} \in \mathbb{R}^{n\times p}$ is a matrix of predictors, $\hat{\boldsymbol{y}} \in \mathbb{R}^n$ is a noisy response vector, and $\psi : \mathbb{R}^p \to \mathbb{R}^d$ is a nonlinear mapping to the feature space. Given a noisy data set with n data points $\{\hat{\boldsymbol{X}}, \hat{\boldsymbol{y}}\}$, our goal is to find the optimal parameters $\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} ||\boldsymbol{y} - \psi(\boldsymbol{X})\boldsymbol{\alpha}||_2^2$ for the noiseless $\{\boldsymbol{X}, \boldsymbol{y}\}$. The noisy regression problem (15) is a proper homotopy surrogate controlled by the continuation level t. When $t = 0$, the problem reduces to $\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} ||\boldsymbol{\alpha}||_2$ which has a trivial solution $\boldsymbol{\alpha} = \mathbf{0}^d$. When $t = 1$, it is the standard regression problem $\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} ||\hat{\boldsymbol{y}} - \psi(\hat{\boldsymbol{X}})\boldsymbol{\alpha}||_2^2$ without the regularization term $||\boldsymbol{\alpha}||_2$, which could overfit to the noisy data. To have the best prediction performance on the noiseless $\{\boldsymbol{X}, \boldsymbol{y}\}$, we need to find the solution for the homotopy optimization problem (15) with a proper but unknown $t \in \mathcal{T} = [0, 1]$.

Our proposed CPL method can learn the whole continuation path for this problem. We build a simple fully connected neural network $\boldsymbol{\alpha}(t) = h_{\boldsymbol{\phi}}(t)$ as the path model which maps any valid homotopy level t to its solution $\boldsymbol{\alpha}(t)$, and reformulate the noisy regression problem into:

$$\min_{\boldsymbol{\phi}} t||\hat{\boldsymbol{y}} - \psi(\hat{\boldsymbol{X}})h_{\boldsymbol{\phi}}(t)||_2^2 + (1-t)||h_{\boldsymbol{\phi}}(t)||_2. \tag{16}$$

Then the path model can be trained by simple gradient descent to obtain the optimal model parameters $\boldsymbol{\phi}$. We learn the continuation path for four different noisy regression problems and report their results in Figure 5. CPL successfully learns the continuation path for all problems, which can be used to directly locate the optimal homotopy level t. The problem details can be found in Appendix B.

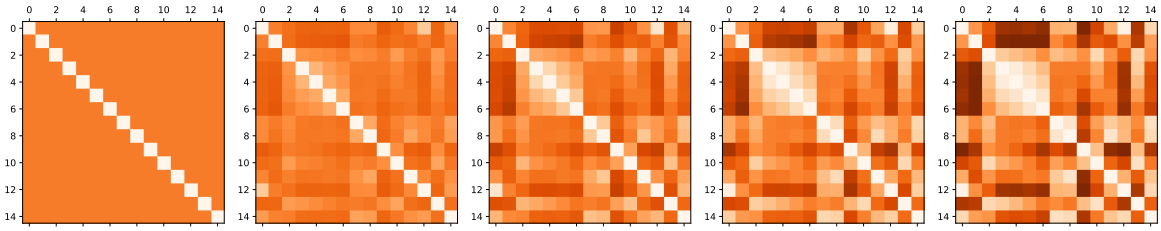
6.3. Neural Combinatorial Optimization

The CPL method can also improve the generalization performance for a neural combinatorial optimization (NCO) solver (Vinyals et al., 2015; Kool et al., 2019), which learns to directly predict the solution for a combinatorial optimization problem. We use the popular traveling salesman problem (TSP) to motivate our approach. A Euclidean TSP instance s is a fully connected graph with n nodes (cities) where each city has its own two-dimensional location $\boldsymbol{y}$. The traveling cost between two cities i and j can be defined as the distance $c_{ij} = ||\boldsymbol{y}_i - \boldsymbol{y}_j||_2$. The goal of TSP is to find a valid tour with the shortest cost to visit all cities exactly once and then return to the starting city. We can represent a valid tour as a permutation of all cities $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_i, \cdots, \pi_n), \pi_i \in \{1, \cdots, n\}$, and the objective is to find the optimal tour to minimize:

$$l(\boldsymbol{\pi}|s) = c_{\pi_n\pi_1} + \sum_{i=1}^{n-1} c_{\pi_i\pi_{i+1}}. \tag{17}$$

We can construct a smoother homotopy subproblem by gradually changing the cost between city pairs (Coy et al., 2000):

$$\hat{c}_{ij}(t) = c_{ij}^t, \quad t \in \mathcal{T} = [0, 1]. \tag{18}$$



(a) t = 0 (b) t = 0.25 (c) t = 0.5 (d) t = 0.75 (e) t = 1

Figure 6. **Homotopy Subproblems for TSP**: **(a)** When $t = 0$, we have the smoothest subproblem where all the distances are the same. In this case, all valid tours will have the same length and be equally optimal, so solving TSP could be trivial. **(b)-(d)** When t increases, we have more rugged subproblems that gradually transform back to the original problem. **(e)** When $t = 1$, we have the original problem.

Table 2. Results of generalization performance on the realistic TSPLib instances (with 51 to 200 cities). All learning-based methods are trained on synthetic and uniformly distributed TSP instances with 100 cities. The full table can be found in Table 6.

	OR-Tools	Wu et al. (Wu et al., 2021)	DACT	DACT(long)	AM-S	POMO	CPL
Optimal Gap	3.34%	4.17%	3.90%	2.07%	22.83%	2.15%	1.72%

We always normalize the original cost matrix such that $c_{ij} \in [0, 1]$ and hence all $\hat{c}_{ij}(t) \in [0, 1]$, and we also normalize the smoothed cost matrix to have the same mean with the original cost matrix $\sum \hat{c}_{ij}(t) = \sum c_{ij} = \sum \bar{c}$. The cost matrices with different values of t are shown in Figure 6. With the smoothed costs, we can define the continuation function of the tour $\boldsymbol{\pi}$ for problem instance s as:

$$H(\boldsymbol{\pi}, t|s) = c^t_{\pi_n \pi_1} + \sum_{i=1}^{n-1} c^t_{\pi_i \pi_{i+1}}, \quad t \in \mathcal{T} = [0, 1]. \tag{19}$$

For $t = 0$, all valid tours have the same total cost $H(\boldsymbol{\pi}, t = 0|s) = n\bar{c}$, and the optimization problem becomes trivial to solve. For $t = 1$, we have the original objective function $H(\boldsymbol{\pi}, t = 1|s) = l(\boldsymbol{\pi}|s)$.

With the above homotopy idea, we can build a CPL-enhanced solver for neural combinatorial optimization. As shown in Figure 7, our model can easily generate multiple solutions on the continuation path for the smoothed subproblems (e.g., $t_m \in [0, 1]$) to make a multi-shot prediction that might contain solutions with better generalization performance. By leveraging this property, CPL can obtain a robust generalization performance on unseen TSPlib instances with unseen sizes and distributions as shown in Table 2. Model details and more results can be found in Appendix. C.

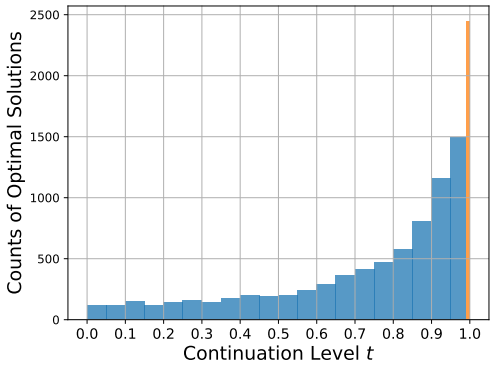


Figure 7. **CPL inference and counts of optimal solutions.** For $10,000$ random 100-city TSP instances, the model with $t = 1$ can only generate the best solutions for roughly 25% instances. We leverage solutions on the continuation path ($t \in [0, 1)$) to achieve better overall performance.

7. Conclusion and Limitation

Conclusion We have proposed a novel continuation path learning (CPL) method to approximate the whole continuation path for homotopy optimization. The experimental results have shown that CPL can successfully learn the solution path for different applications. In addition, compared with the classical homotopy optimization method, CPL can achieve similar or even better performance for the original complicated problem. We believe CPL could be a novel and promising method for homotopy optimization.

Limitation A limitation of CPL is that we need to build and train a model for learning the continuation path. The suitable model design will mainly depend on the given problem, and some domain knowledge might also be required for efficient model building. Additional theoretical analyses, such as problem transformation and the relation to the value function approach, are important future works.

Acknowledgements

This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11215622).

References

Accorsi, L., Lodi, A., and Vigo, D. Guidelines for the computational testing of machine learning approaches to vehicle routing problems. *Operations Research Letters*, 50(2):229–234, 2022.

Ackley, D. *A connectionist machine for genetic hillclimbing*. Springer Science & Business Media, 1987.

Allgower, E. L. and Georg, K. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 1990.

Amos, B. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

Anandkumar, A., Deng, Y., Ge, R., and Mobahi, H. Homotopy analysis for tensor pca. In *Conference on Learning Theory*, pp. 79–104. PMLR, 2017.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. The traveling salesman problem: A computational study, 2006.

Bank, B., Guddat, J., Klatte, D., Kummer, B., and Tammer, K. *Non-linear parametric optimization*. Springer, 1983.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations (ICLR) Workshops*, 2017.

Bengio, Y. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

Beyer, H.-G. and Schwefel, H.-P. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1): 3–52, 2002.

Blake, A. and Zisserman, A. *Visual reconstruction*. MIT press, 1987.

Bonnans, J. F. and Shapiro, A. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

Brox, T. and Malik, J. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2010.

Chapelle, O., Chi, M., and Zien, A. A continuation method for semi-supervised svms. In *International Conference on Machine Learning (ICML)*, 2006.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations (ICLR)*, 2016.

Chen, T., Chen, X., Chen, W., Wang, Z., Heaton, H., Liu, J., and Yin, W. Learning to optimize: A primer and a benchmark. *The Journal of Machine Learning Research*, 23(1):8562–8620, 2022.

Chen, X. and Tian, Y. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Coy, S. P., Golden, B. L., and Wasil, E. A. A computational study of smoothing heuristics for the traveling salesman problem. *European Journal of Operational Research*, 124(1):15–27, 2000.

d O Costa, P. R., Rhuggenaath, J., Zhang, Y., and Akcay, A. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian Conference on Machine Learning*, pp. 465–480. PMLR, 2020.

Dosovitskiy, A. and Djolonga, J. You only train once: Loss-conditional training of deep networks. *International Conference on Learning Representations (ICLR)*, 2020.

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Wibisono, A. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Dunlavy, D. M. and O'Leary, D. P. Homotopy optimization methods for global optimization. In *Technical Report*, 2005.

Eaves, B. C. Homotopies for computation of fixed points. *Mathematical Programming*, 3(1):1–22, 1972.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Evans, L. C. *Partial differential equations*, volume 19. American Mathematical Society, 2010.

Gao, K. and Sener, O. Generalizing gaussian smoothing for random search. In *International Conference on Machine Learning (ICML)*, 2022.

Gargiani, M., Zanelli, A., Dinh, Q. T., Diehl, M., and Hutter, F. Transferring optimality across data distributions via homotopy methods. In *International Conference on Learning Representations (ICLR)*, 2020.

Garnett, R. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

Gold, S., Rangarajan, A., and Mjolsness, E. Learning with preknowledge: clustering with point and graph matching distance measures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1994.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Gulcehre, C., Moczulski, M., Visin, F., and Bengio, Y. Mollifying networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Guo, Y., Feng, S., Le Roux, N., Chi, E., Lee, H., and Chen, M. Batch reinforcement learning through continuation method. In *International Conference on Learning Representations (ICLR)*, 2020.

Ha, D., Dai, A. M., and Le, Q. V. Hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

Hansen, N. and Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5(Oct):1391–1415, 2004.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. On graduated optimization for stochastic non-convex problems. In *International Conference on Machine Learning (ICML)*, 2016.

Helsgaun, K. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

Helsgaun, K. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, pp. 24–50, 2017.

Himmelblau, D. M. et al. *Applied nonlinear programming*. McGraw-Hill, 1972.

Hottung, A. and Tierney, K. Neural large neighborhood search for the capacitated vehicle routing problem. In *European Conference on Artificial Intelligence (ECAI)*, 2020.

Hottung, A., Kwon, Y.-D., and Tierney, K. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations (ICLR)*, 2022.

Hruby, P., Duff, T., Leykin, A., and Pajdla, T. Learning to solve hard minimal problems. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ingber, L. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

Iwakiri, H., Wang, Y., Ito, S., and Takeda, A. Single loop gaussian homotopy method for non-convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jain, P. and Kar, P. Non-convex optimization for machine learning. *Foundations and Trends in Machine Learning*, 10(3-4):142–363, 2017.

Jayakumar, S. M., Menick, J., Czarnecki, W. M., Schwarz, J., Rae, J., Osindero, S., Teh, Y. W., Harley, T., and Pascanu, R. Multiplicative interactions and where to find them. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Kim, M. and De la Torre, F. Gaussian processes multiple instance learning. In *International Conference on Machine Learning (ICML)*, 2010.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019.

Kumar, M., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., and Min, S. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Levin, E., Kileel, J., and Boumal, N. The effect of smooth parametrizations on nonconvex optimization landscapes. *arXiv preprint arXiv:2207.03512*, 2022.

Li, S., Drgona, J., Tuor, A. R., Pileggi, L., and Vrabie, D. L. Homotopy learning of parametric solutions to constrained optimization problems. `https://openreview.net/forum?id=GdimRqV_S7`, 2023.

Lin, X., Yang, Z., Zhang, Q., and Kwong, S. Controllable pareto multi-task learning. *arXiv preprint arXiv:2010.06313*, 2020.

Lin, X., Yang, Z., and Zhang, Q. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations (ICLR)*, 2022a.

Lin, X., Yang, Z., Zhang, X., and Zhang, Q. Pareto set learning for expensive multiobjective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Liu, X., Lu, Y., Abbasi, A., Li, M., Mohammadi, J., and Kolouri, S. Teaching networks to solve optimization problems. *arXiv preprint arXiv:2202.04104*, 2022.

Ma, Y., Li, J., Cao, Z., Song, W., Zhang, L., Chen, Z., and Tang, J. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Mehmood, S. and Ochs, P. Automatic differentiation of some first-order methods in parametric optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Mehmood, S. and Ochs, P. Differentiating the value function by using convex duality. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Mobahi, H. and Fisher III, J. W. On the link between gaussian homotopy continuation and convex envelopes. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 43–56. Springer, 2015a.

Mobahi, H. and Fisher III, J. W. A theoretical analysis of optimization by gaussian continuation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015b.

Mobahi, H. and Ma, Y. Gaussian smoothing and asymptotic convexity. *Coordinated Science Laboratory Report no. UILU-ENG-12-2201, DC-254*, 2012.

Navon, A., Shamsian, A., Fetaya, E., and Chechik, G. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2021.

Nesterov, Y. and Spokoiny, V. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Perron, L. and Furnon, V. Or-tools, 2019. URL `https://developers.google.com/optimization/`.

Rosenbrock, H. An automatic method for finding the greatest or least value of a function. *The computer journal*, 3 (3):175–184, 1960.

Rosset, S. and Zhu, J. Piecewise linear regularized solution paths. *The Annals of Statistics*, pp. 1012–1030, 2007.

Schmidhuber, J. Learning to control fast-weight memories: an alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Sener, O. and Koltun, V. Learning to guide random search. In *International Conference on Learning Representations (ICLR)*, 2020.

Shahriari, B., Swersky, K., Wang, Z., Adams, R., and De Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Still, G. Lectures on parametric optimization: An introduction. *Optimization Online*, 2018.

Terzopoulos, D. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, 1988.

Tibshirani, R. J. and Taylor, J. The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335–1371, 2011.

Tripp, A., Daxberger, E., and Hernández-Lobato, J. M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

Van Laarhoven, P. J. and Aarts, E. H. Simulated annealing. In *Simulated annealing: Theory and applications*, pp. 7–15. Springer, 1987.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Wang, L., Fonseca, R., and Tian, Y. Learning search space partition for black-box optimization using monte carlo tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Wasserstrom, E. Numerical solutions by the continuation method. *SIAM Review*, 15(1):89–119, 1973.

Widder, D. V. *The heat equation*, volume 67. Academic Press, 1976.

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

Wu, Y., Song, W., Cao, Z., Zhang, J., and Lim, A. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Wu, Z. The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. *SIAM Journal on Optimization*, 6(3):748–768, 1996.

Xin, L., Song, W., Cao, Z., and Zhang, J. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Yang, R., Sun, X., and Narasimhan, K. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Yuille, A. Energy functions for early vision and analog networks. *Biological Cybernetics*, 61(2):115–123, 1989.

We provide detailed experimental settings, extra experimental results, and more discussions in this Appendix.

A. Synthetic Test Benchmarks

A.1. Problem Definition

In this experiment, we evaluate the proposed continuation path learning (CPL) method with other homotopy optimization algorithms on the following three widely-used non-convex optimization problems.

Ackley Optimization Problem (Ackley, 1987):

$$f(x,y) = -20e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5(\cos 2\pi x + \cos 2\pi y)} + e + 20. \tag{20}$$

Rosenbrock Optimization Problem (Rosenbrock, 1960):

$$f(x,y) = 100(y-x^2)^2 + (1-x)^2. \tag{21}$$

Himmelblau Optimization Problem (Himmelblau et al., 1972):

$$f(x,y) = (x^2+y-11)^2 + (x+y^2-7)^2. \tag{22}$$

A.2. Experimental Setting

We compare CPL with a simple gradient descent algorithm (GD), a classical Gaussian optimization algorithm (GradOpt) (Hazan et al., 2016) with two smoothing parameters ($\gamma = 0.5$ or 0.8), and the recently proposed single loop Gaussian homotopy algorithm with fixed ratio update (SLGH$_r$) (Iwakiri et al., 2022) with $\gamma = 0.995$ or 0.999. We report the results of these algorithms with fine-tuned hyperparameters from Iwakiri et al. (2022).

For our proposed CPL method, we build a simple fully connected (FC) neural network as the continuation path model. It has two hidden layers each with 128 hidden nodes. Since the model's gradient can be decomposed with the simple chain rule as in Section 4.3, it can be easily optimized by the (zeroth-order) gradient descent algorithm similar to other homotopy optimization algorithms. CPL can learn to approximate the whole continuation path simultaneously, hence it does not require any predefined continuation schedule or smoothing parameter.

We use Gaussian homotopy (GH) as the homotopy method and mainly follow the experimental setting from Iwakiri et al. (2022) for each optimization problem.

Ackley The Ackley optimization problem does not have an analytical form for its Gaussian homotopy function, and hence we use the zeroth-order method to approximate its Gaussian homotopy gradient (11). The total number of function evaluations is $1,000$ for all algorithms. CPL uses 950 evaluations for path model training and 50 evaluations for final local search with homotopy level $t = 1$.

Rosenbrock The Gaussian homotopy function of the Rosenbrock optimization problem has the following analytical form (Mobahi & Ma, 2012; Iwakiri et al., 2022):

$$\begin{aligned} GH(x,y,t) =& \mathbb{E}_{u_x,u_y}[f(x+\beta(1-t)u_x, y+\beta(1-t)u_y)] \\ =& 100x^4 + [-200y + 600\beta^2(1-t)^2 + 1]x^2 - 2x + 100y^2 - 200\beta^2(1-t)^2 y \\ &+ 300\beta^4(1-t)^2 + 101\beta^2(1-t)^2 + 1. \end{aligned} \tag{23}$$

Therefore, we can use the simple first-order gradient method for CPL and all the other homotopy optimization algorithms. The total number of function evaluations is $20,000$. For CPL, $19,000$ evaluations are used for path model training, and the rest $1,000$ is for the final local search with homotopy level $t = 1$. We set $\beta = 1.5$ according to Iwakiri et al. (2022).

Himmelblau Similar to the Ronsebrock problem, since the himmelblau optimization problem is polynomial, it has an analytical Gaussian homotopy function:

$$\begin{aligned} GH(x,y,t) =& \mathbb{E}_{u_x,u_y}[f(x+\beta(1-t)u_x, y+\beta(1-t)u_y)] \\ =& x^4 + (2y + 6\beta^2(1-t)^2 - 21)x^2 + (2y^2 + 2\beta^2(1-t)^2 - 14)x \\ &+ y^4 + (6\beta^2(1-t)^2 - 13)y^2 + (2\beta^2(1-t)^2 - 22)y \\ &+ 6\beta^4(1-t)^4 - 34\beta^2(1-t)^2 + 170 \end{aligned} \tag{24}$$

We also use the first-order gradient method to optimize this problem for all algorithms. The total number of function evaluations is $2,000$. CPL has $1,900$ evaluations for model training and 100 for the final local search with $t = 1$. The parameter β is set to 2 as in Iwakiri et al. (2022).

A.3. Computational Cost

Table 3. Runtime of gradient, descent, SLGH, and CPL with CPU or GPU.

Problem # Iteration	Ackley 1,000	Himmelblau 2,000	Rosenbrock 20,000
Gradient Descent	0.9s	1.6s	11.2s
SLGH	0.9s	1.7s	12.6s
CPL (CPU)	0.9s	1.8s	13.2s
CPL (GPU)	1.6s	2.8s	15.4s

The CPL model training can be easily done by highly efficient deep learning frameworks such as PyTorch. For these non-convex optimization benchmarks, depending on the number of iterations, CPL typically needs 1 to 15 seconds to train the model while the run times for other model-free methods are 1 to 13 seconds. It should be pointed out that CPL can learn the whole continuation path while the other methods are to find a single final solution. In addition, the CPL training on GPU (RTX-3080) is actually slower than its counterpart on CPU. For such a small model, the reason could be the cost of data transformation from RAM to GPU is larger than the speedup of GPU over CPU.

A.4. Effect of the Model Size

Table 4. CPL performance on the original problem with different model sizes.

	Baseline	Single Hidden Layer			Two Hidden Layers			Three Hidden Layers		
Model Size	SLGH	16	128	1024	16-16	128-128 (this paper)	1024-1024	16-16-16	128-128-128	1024-1024-1024
Ackley	6.650	2.605	0.478	0.261	0.089	0.006	0.005	0.024	0.007	0.006
Himmelblau	6.9e-5	84.25	11.92	3.8e-05	19.46	2.3e-6	2.8e-6	8.74	2.1e-6	2.6e-6
Rosenbrock	0.0327	0.0548	0.0409	0.0282	0.0371	0.0018	0.0023	0.0220	0.0016	0.0019

The performance of CPL depends on the neural network architectures. Indeed, different optimization problems and applications could require different CPL models. A general guideline is that the model should be large enough to learn the continuation path for the given problem. Therefore, we should care about the model size and also problem-specific structure.

We report the results for different models with various sizes in Table 4. Based on the results, a very small model (such as those with single hidden layers) is not able to learn the whole continuation path, and hence has poor performance. On the other hand, once the model has sufficient capacity (such as those three-layer models with more than 128 hidden units), further increasing the model size will not lead to significantly better performance. Therefore, it is important to choose a suitable model size for a given problem.

B. Noisy Regression

In this experiment, we test our proposed CPL method on the following four different noisy regression problems.

F1 This problem has the following ground truth function relation between x and y:

$$y = 0.5\sin(x) + 0.3\cos(2x) + 2\cos(3x), \tag{25}$$

where $x \in [-5, 5]$. For the noisy regression, we report $\hat{y} = y + 0.1\varepsilon$ where $\varepsilon \sim N(0, 1)$ as the noise response value, and set $\psi(x) = [\sin(x), \cos(2x), \cos(3x)]$. Therefore, the optimal $\boldsymbol{\alpha} = [0.5, 0.3, 2]$.

F2 This problem has the following ground truth function relation between x and y:

$$y = \cos(x) + 0.2\sin(2x) + 0.5\sin(3x), \tag{26}$$

where $x \in [-5, 5]$. For the noisy regression, we report $\hat{y} = y + 0.1\varepsilon$ where $\varepsilon \sim N(0, 1)$ as the noise response value, and set $\psi(x) = [\cos(x), \sin(2x), \sin(3x)]$. Therefore, the optimal $\boldsymbol{\alpha} = [1, 0.2, 0.5]$.

F3 This problem has the following ground truth function relation between x and y:

$$y = e^{0.25x} - 0.2\cos(x) + 0.5\sin(4x), \tag{27}$$

where $x \in [-5, 5]$. For the noisy regression, we report $\hat{y} = y + 0.1\varepsilon$ where $\varepsilon \sim N(0, 1)$ as the noise response value, and set $\psi(x) = [e^{0.25x}, \cos(x), \sin(4x)]$. Therefore, the optimal $\boldsymbol{\alpha} = [1, -0.2, 0.5]$.

F4 This problem has the following ground truth function relation between x and y:

$$y = 2 * \ln 0.25|x| + 3\sin(6x) + 4\cos(0.5x), \tag{28}$$

where $x \in [-5, 5]$. For the noisy regression, we report $\hat{y} = y + 0.1\varepsilon$ where $\varepsilon \sim N(0, 1)$ as the noise response value, and set $\psi(x) = [\ln 0.25|x|, \sin(6x), \cos(0.5x)]$. Therefore, the optimal $\boldsymbol{\alpha} = [2, 3, 4]$.

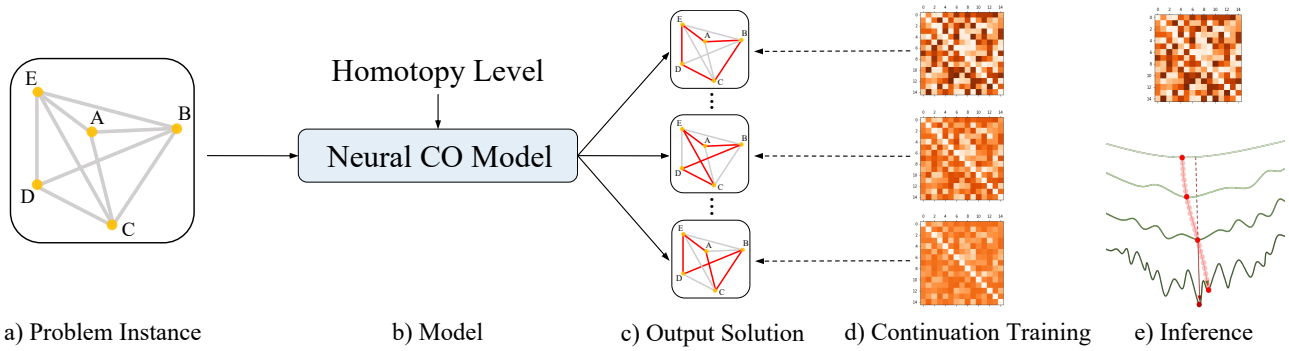


Figure 8. **Continuation Path Learning Model for Neural Combinatorial Optimization**: Our proposed CPL model can learn to construct different solutions for the smoothed subproblems on the continuation path for a given problem instance. In inference, it leverages these solutions to make a multi-shot prediction for better performance.

C. Continuation Path Learning for Neural Combinatorial Optimization

C.1. Continuation-based constructive NCO model

In this experiment, we focus on learning the continuation path for constructive neural combinatorial optimization (NCO) (Vinyals et al., 2015; Bello et al., 2017). This approach learns the policy model with parameter $\boldsymbol{\theta}$ to construct a solution (e.g., a tour $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_i, \cdots, \pi_n), \pi_i \in \{1, \cdots, n\}$) for a combinatorial optimization problem instance s (e.g., TSP) in an auto-regressive manner:

$$p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\pi_i|s, \pi_{1:i-1}). \tag{29}$$

In contrast to a policy with constant parameters $\boldsymbol{\theta}$, we propose to build a policy model with dynamic parameters $\boldsymbol{\theta}(t)$ conditioned on the continuation level t:

$$p_{\boldsymbol{\theta}(t)}(\boldsymbol{\pi}|s) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}(t)}(\pi_i|s, \pi_{1:i-1}), \tag{30}$$

such that it can generate different tours for different smoothed subproblems as in Figure 8. By assigning different t, we can easily obtain solutions of different smoothed subproblems on the continuation path for a given instance.

The proposed continuation path learning idea and the policy model framework are general and can be used for any constructive NCO model. We use the seminal Attention Model (Kool et al., 2019) as our constructive model. The recent work shows that only changing (part of) the AM decoder parameters is sufficient to construct solutions with significantly better performance (Hottung et al., 2022) or with very different trade-offs among different objectives (Lin et al., 2022a). Therefore, we also propose to let only part of the AM decoder parameters depend on the continuation level t:

$$[W^Q(t), W^{\text{Proj}}(t)] = \textbf{MLP}(t), \tag{31}$$

where $W^Q(t)$ and $W^{\text{proj}}(t)$ are the query and projection parameters for the Multi-Head Attention (MHA) (Vaswani et al., 2017) layer in the AM decoder. More advanced model structures such as multiplicative interactions (Jayakumar et al., 2020) and hypernetwork (Schmidhuber, 1992; Ha et al., 2017) can be used to learn the conditioned parameters, but we find a simple MLP model is good enough for our model.

To construct a valid tour, our proposed model first tasks the problem instance s (e.g., the graph with n fully connected nodes for TSP) as input to the AM encoder and obtains the n d-dimensional node embedding $[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n]$ for each city. For selecting the i-th city into the tour, the AM decoder combines the embedding of the first selected node $\boldsymbol{h}_{\pi_1}$ and the most current selected node $\boldsymbol{h}_{\pi_{i-1}}$ to obtain the query embedding $\boldsymbol{h}_Q(t) = W^Q(t)[\boldsymbol{h}_{\pi_1}, \boldsymbol{h}_{\pi_{i-1}}]$. Following the setting of POMO (Kwon et al., 2020), we do not include an extra graph embedding. The query embedding will be further updated by the multi-head attention with all node embedding:

$$\hat{\boldsymbol{h}}_Q = \textbf{MHA}(Q = \boldsymbol{h}^Q(t), K = W^K[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n], V = W^V[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n])W^{\text{proj}}(t), \tag{32}$$

where $W^{\text{proj}}(t)$ projects the multi-head output of MHA into an d-dimensional embedding. With the query embedding $\hat{\boldsymbol{h}}_Q$ and the embedding $\boldsymbol{h}_i$ for each city, we can calculate the logit for each city:

$$\text{logit}_j = \begin{cases} C \cdot \tanh(\frac{\hat{\boldsymbol{h}}_Q^T \boldsymbol{h}_j}{\sqrt{d}}) & \text{if } j \neq \boldsymbol{\pi}_{p'} \quad \forall p' < i, \\ -\infty & \text{otherwise.} \end{cases} \tag{33}$$

The logits are further clipped into $[-C, C]$ for all non-selected nodes with $C = 10$ as in (Kool et al., 2019) and all already selected nodes are masked in $-\infty$. The policy model can autoregressively construct a valid tour by following the probability $p_{\boldsymbol{\theta}(t)}(\boldsymbol{\pi}_t = j | s, \boldsymbol{\pi}_{1:i-1}) = e^{\text{logit}_j} / \sum_{k \in \{j \neq \boldsymbol{\pi}_{p'} \forall p' < i\}} e^{\text{logit}_k}$.

In the proposed model, the node embedding $[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n]$ and the key K and value V of MHA are shared by all continuation level $t \in \mathcal{T} = [0, 1]$. Therefore, we only need to calculate them once and then can repeatedly reuse them to construct solutions for different continuation levels t. The whole model structure is similar to multi-objective optimization model proposed in (Lin et al., 2022a) but is to learn the continuation path for a single objective function.

C.2. Model training

We have proposed the smoothed subproblems (e.g., (19)) and the policy model to construct feasible solutions (e.g.,(30)) for different continuation levels. Now the goal is to properly train the model so that the generated solutions are on the continuation path (e.g., the optimal solution for each subproblem) of the original problem (17). The training goal can be defined as:

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi} \sim p_{\boldsymbol{\theta}(t)}(\cdot|s), t \sim \mathcal{T}, s \sim \mathcal{S}} H(\boldsymbol{\pi}, t | s), \tag{34}$$

where $\mathcal{S}$ is the set of problem instances, $\mathcal{T} = [0, 1]$ is the valid continuation level, and $\boldsymbol{\pi} \sim p_{\boldsymbol{\theta}(t)}(\cdot|s)$ is the tour generated by the stochastic policy model with respect to the sampled s and t.

We follow our proposed gradient-based continuation path learning method in **Algorithm 3** to train the policy model. The gradient is approximated with REINFORCE (Williams, 1992) and POMO (Kwon et al., 2020) rollout:

$$\nabla \mathcal{J}(\boldsymbol{\theta}) \approx \frac{1}{MBN} \sum_{m=1}^{M} \sum_{i=1}^{B} \sum_{j=1}^{N} [(H(\boldsymbol{\pi}_i^j, t_m | s_i) - b(s_i | t_m)) \nabla_{\boldsymbol{\theta}(t_m)} \log p_{\boldsymbol{\theta}(t_m)}(\boldsymbol{\pi}_i^j | s_i)], \tag{35}$$

with M sampled continuation levels, B problem instances, and N solutions with diverse started nodes for each instance. We use the shared baseline $b(s_i | t_m) = \frac{1}{N} \sum_{j=1}^{N} H(\boldsymbol{\pi}_i^j, t_m | s_i)$ for each instance as in POMO (Kwon et al., 2020).

C.3. Inference from the continuation path

Algorithm 4 CPL Inference

1: **Input:** instance s, continuation path model $p_{\boldsymbol{\theta}(t)}(\boldsymbol{\pi}|s)$, number of sampled solutions M
2: $t_1, t_2 \ldots, t_M \sim \mathcal{T} = [0, 1]$
3: $\boldsymbol{\pi}_{t_m} \leftarrow$ **GreedyRollout**$(p_{\boldsymbol{\theta}(t_m)}(\cdot|s)) \quad \forall t_m$
4: $\boldsymbol{\pi}_{\text{best}} = \arg\min_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}_{t_1}, \ldots, \boldsymbol{\pi}_{t_M}\}} H(\boldsymbol{\pi}, t = 1 | s)$
5: Output: $\boldsymbol{\pi}_{\text{best}}$

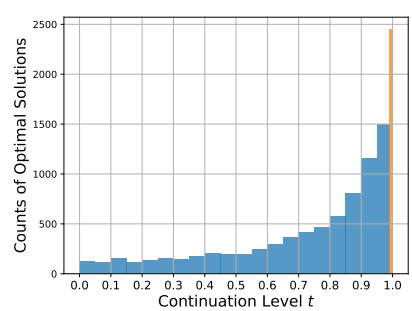


Figure 9. **CPL inference and counts of optimal solutions.** For $10,000$ random 100-city TSP instances, the model with $t = 1$ can only generate the best solutions for roughly 25% instances. We leverage solutions on the continuation path ($t \in [0, 1)$) to achieve better overall performance.

Due to the one-shot prediction mechanism and generalization gap, for a new encountered problem instance s, the generated solution $\boldsymbol{\pi}_1$ with $t = 1$ might not be the optimal solution for the original objective $H(\boldsymbol{\pi}, t = 1|s)$ as shown in Figure 9. With our proposed model, we can easily generate multiple solutions on the continuation path for the smoothed subproblems (e.g., $t_m \in \mathcal{T} = [0, 1]$) to make a multi-shot prediction that might contain solutions with better generalization performance.

The inference method is summarized in **Algorithm 4**. For each problem instance s, we sample a set of M continuation levels t_m, and construct their solutions with POMO rollout (Kwon et al., 2020). The solution with the best original objective value (not the smoothed value) will be chosen as the final solution.

C.4. Experimental Setting

Problem Setting We evaluate the proposed continuation path learning (CPL) method on both randomly generated TSP and CVRP instances that are widely used in neural combinatorial optimization (Kool et al., 2019), as well as more realistic benchmark problems following the suggestion in (Accorsi et al., 2022).

Model setting We use the Attention Model (AM) (Kool et al., 2019) as our policy network for construction-based neural combinatorial optimization. The model has a computation-heavy encoder to produce embedding for each node (e.g., city for TSP), and a lightweight decoder to autoregressively generate a valid solution (e.g., tour) based on the node embeddings. The same model structure is used for solving different problems.

We follow the hyperparameters setting from POMO (Kwon et al., 2020). For the encoder, there are 6 multi-head attention layers with 128-dimensional embedding, and each has 8 heads with 16-dimensional key/value/query embeddings. The fully connected linear sub-layer has dimension 512 in each attention layer. We build our CPL model based on the new POMO (Kwon et al., 2020) codebase [2] under the MIT License, where it uses InstanceNorm for the normalization layers and does not include graph embedding as an output for the decoder.

Only the parameters for the query token and projection operator are conditioned on the continuation level, and the other parameters are all fixed and shared for all smooth subproblems. In this way, all the key/value embeddings for each node need to be calculated only once, and can be repeatedly reused to construct solutions for different smoothed subproblems.

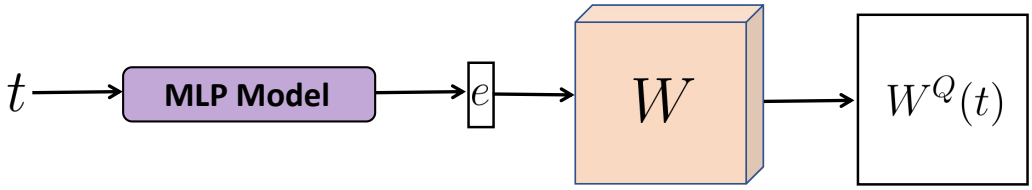


Figure 10. Conditional parameters generation.

We use a hypernetwork (Ha et al., 2017) to generate the continuation conditional parameters $W^Q(t)$ and $W^{\text{proj}}(t)$ as illustrated in Figure 10. It first takes the continuation level t as input for a MLP model to generate a low dimensional embedding $e \in \mathbb{R}^d$. Then the embedding e is linearly projected with a parameter tensor $W \in \mathbb{R}^{n1 \times n2 \times d}$ to the target parameters $W^Q(t) \in \mathbb{R}^{n1 \times n2}$. The number of trainable parameters (e.g., MLP parameters and W) is much smaller than a full hypernetwork. In this work, we use a simple 2-layer MLP with 128 neurons at each hidden layer, and let $d = 4$ for the embedding.

Training and Inference. The optimizer we use is Adam with learning rate $\eta = 10^{-4}$, weight decay $\omega = 10^{-6}$ and batch size $B = 64$. At each training epoch, we randomly generate $100,000$ problem instances on the fly as training data, and train the model for $1,000$ epoch. At each batch (e.g., with batch size $B = 64$), we randomly sample $M = 2$ continuation levels, and draw N (e.g., the number of nodes) trajectories for each problem instance with POMO rollout. In this way, the total update step is matched with $2,000$ in POMO without continuation path learning. We train our model on a single RTX 3080 GPU, which takes roughly 16 minutes for a training epoch with TSP100. For inference with M continuation levels, we always let $t_1 = 1$ and sample the rest $M - 1$ levels from $[0, 1)$.

[2] https://github.com/yd-kwon/POMO/tree/master/NEW_py_ver

C.5. Travelling salesman problem (TSP)

Problem setup A TSP instance contains the locations of N cities. We need to find a tour with the shortest length to visit all cities once and return to the starting city.

Instance generation We follow the setting in AM (Kool et al., 2019) to randomly generate cities from $[0, 1]^2$ with uniform distribution.

C.6. Capacitated vehicle routing problem (CVRP)

Problem setup For a CVRP instance, in addition to the locations of N nodes, each node (city/customer) has a demand d_i to be served. A vehicle with a fixed capacity C from an extra depot node will make multiple round trips to handle the demand for each customer. No demand split is allowed and each city can be visited only once. For a nontrivial and solvable problem instance, we have $\max d_i \leq C < \sum_{i=1}^{N} d_i$. Therefore, the vehicle needs to go back to the depot node to refill its capacity multiple times. The optimal solution is a tour with the shortest length that satisfies the demands of all cities.

Instance generation Similar to the TSP instance, we uniformly generate the customer nodes and depot nodes from the unit square $[0, 1]^2$. For the demands, following AM (Kool et al., 2019), we first randomly sample $\hat{d}_i$ from the discrete set $\{1, 2, \ldots, 9\}$ and then normalize it to $d_i = \frac{\hat{d}_i}{D}$, where $D = 30, 40, 50$ for $N = 20, 50, 100$. The vehicle has a capacity 1.

A series of smoothed subproblems can be constructed similarly to the TSP instances in Section 6.3. Now we have an $(N + 1) \times (N + 1)$ matrix that contains the distance among nodes (N customer nodes and 1 depot node). The capacity and all demands are unchanged in the smoothed subproblems.

C.7. Resutls on Random TSP and CVRP Instances

The experimental results on random TSP and CVRP instances with different sizes are shown in Table 5. Following the setting in (Kool et al., 2019), for each problem with different sizes, $10,000$ randomly generated instances are used as the testing data. We compare CPL with (1) an exact solver Concorde (Applegate et al., 2006); (2) two widely-used heuristic solvers LKH3 (Helsgaun, 2000; 2017) and OR-Tools (Perron & Furnon, 2019); (3) five learning-based improvement methods NeuRewriter (Chen & Tian, 2019), Neural 2-Opt (d O Costa et al., 2020), Neural Large Neighborhood Search (NLNS) (Hottung & Tierney, 2020), Learning Improvement Heuristics in Wu et al. (Wu et al., 2021), and DACT (Ma et al., 2021); and (4) four constructive neural combinatorial optimization methods Graph Convolutional Network (GCN) with beam search (Joshi et al., 2019), AM (Kool et al., 2019) with greedy and sampling rollout, MDAM (Xin et al., 2021) with beam search and POMO (Kwon et al., 2020). Similar to other NCO works, we report the average results (Cost), optimal gap (Gap), and the total run time (Time) for solving all $10,000$ instances. It should be noticed that, even averaging over $10,000$ random instances, the baseline performance could still be slightly different. Therefore we mainly use the gap for comparison as in the most related works (Kwon et al., 2020; Ma et al., 2021).

According to the results in Table 5, our proposed CPL method can consistently improve the POMO performance by leveraging the multi-shot prediction on the continuation path. For TSP, it achieves nearly 0% optimality gap to the exact Concorde solver for instances with 20 and 50 nodes, and a 0.08% gap for TSP100. These results are comparable with the powerful DACT solver with much faster running time, and outperform other learning-based solvers. For the challenging CVRP, CPL's improvements over POMO are much more significant. The 0.08% to 0.16% optimality gaps to the well-developed LKH3 are promising given the fast inference time and minimal specific domain knowledge required by CPL.

CPL needs a longer run time than POMO due to the multi-shot prediction, which is the cost for its better performance. However, it still only takes tens of seconds to a few minutes to solve $10,000$ instances on a single GPU, which is much faster than the learning-based improvement method on multiple GPUs (Ma et al., 2021). How to design a better continuation path construction and sampling method to further improve the performance-time ratio for CPL could be interesting future work.

Table 5. Experimental results on TSP and CVRP with random instances

TSP									
	TSP20			TSP50			TSP100		
Method	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
Concorde	3.83	-	(5m)	5.69	-	(13m)	7.76	-	(1h)
LKH3	3.83	0.00%	(42s)	5.69	0.00%	(6m)	7.76	0.00%	(25m)
OR Tools	3.86	0.94%	(1m)	5.85	2.87%	(5m)	8.06	3.86%	(23m)
Neural 2-Opt	3.83	0.00%	(15m)	5.79	0.12%	(29m)	7.83	0.87%	(41m)
Wu et al.(T=5k)	3.83	0.00%	(1h)	5.70	0.20%	(1.5h)	7.97	1.42%	(2h)
DACT (T=10k)	3.83	0.00%	(10m)	5.70	0.00%	(1h)	7.77	0.09%	(2.5h)
GCN-beam search	3.83	0.01%	(12m)	5.69	0.01%	(18m)	7.87	1.39%	(40m)
AM-greedy	3.84	0.19%	(1s)	5.76	1.21%	(1s)	8.03	3.51%	(2s)
AM-sampling	3.83	0.07%	(1m)	5.71	0.39%	(5m)	7.92	1.98%	(22m)
MDAM-beam search	3.84	0.00%	(3m)	5.70	0.03%	(14m)	7.79	0.38%	(44m)
POMO	3.83	0.00%	(3s)	5.69	0.03%	(16s)	7.78	0.15%	(1m)
CPL (4 sols.)	3.83	0.00%	(10s)	5.69	0.01%	(1m)	7.77	0.09%	(3m)
CPL (8 sols.)	3.83	0.00%	(19s)	5.69	0.00%	(2m)	7.77	0.08%	(7m)
CVRP									
	CVRP20			CVRP50			CVRP100		
Method	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
LKH3	6.12	-	(2h)	10.38	-	(7h)	15.68	-	(12h)
OR Tools	6.42	4.84%	(2m)	11.22	8.12%	(12m)	17.14	9.34%	(1h)
NeuRewriter	6.16	-	(22m)	10.51	-	(18m)	16.10	-	(1h)
NLNS	6.19	-	(7m)	10.54	-	(24m)	15.99	-	(1h)
Wu et al.(T=5k)	6.12	0.39%	(2h)	10.45	0.70%	(4h)	16.03	2.47%	(5h)
DACT (T=10k)	6.13	-0.08%	(35m)	10.39	0.14%	(1.5h)	15.71	0.19%	(4.5h)
AM-greedy	6.40	4.45%	(1s)	10.93	5.34%	(1s)	16.73	6.72%	(3s)
AM-sampling	6.24	1.97%	(3m)	10.59	2.11%	(7m)	16.16	3.09%	(30m)
MDAM-beam search	6.14	0.18%	(5m)	10.48	0.98%	(15m)	15.99	2.23%	(1h)
POMO	6.14	0.21%	(5s)	10.42	0.45%	(26s)	15.73	0.32%	(2m)
CPL (4 sols.)	6.13	0.14%	(16s)	10.40	0.26%	(1m)	15.72	0.21%	(7m)
CPL (8 sols.)	6.13	0.08%	(33s)	10.39	0.12%	(3m)	15.71	0.16%	(14m)

C.8. Generalization Performance on Realistic TSP Benchmark

We test our proposed continuation path learning (CPL) method's performance on the TSPLib benchmark problems as shown in Table 6. These problem instances have different city distributions and different sizes from 50 to 200. We mainly compare CPL with the heuristic solver OR-Tools (Perron & Furnon, 2019), two improvement-based NCO methods Wu et al. (Wu et al., 2021) and DACT (Ma et al., 2021), and two construction-based NCO methods AM (Kool et al., 2019) with sampling and POMO (Kwon et al., 2020). All these methods except POMO need to continuously interact with the problem instances (e.g., with $3,000$ to $10,000$ steps) or generate a large number of candidate solutions for selection (e.g., $10,000$ for AM with sampling). For all methods, we follow the setting in DACT (Ma et al., 2021) to report results on the first five instances (have sizes 51 to 76) with models trained on 50 cities, and the results for the rest instances (have sizes 99 to 200) with models trained on 100 cities.

According to the results, CPL with 8 continuation solutions can further improve the POMO's already very competitive overall performance from 2.15% to 1.72%. This overall result is even better than those for DACT with the strong setting (e.g., with T = $10k$ steps and 4 augmentations). Following DACT (Ma et al., 2021), we also report the average optimality gap for instances with different sizes. CPL generally has good performance for instances with sizes from 50 to 150, which is close to the training size (50 and 100). However, its performance drastically decreases for instances with larger numbers of cities that are far from the training set. This limitation is shared among other construction-based and improvement-based neural combinatorial optimization methods.

C.9. Generalization Performance on Realistic CVRP Benchmark

Table 7 shows the results on 22 realistic CVRPLIB benchmark problems (Uchoa et al., 2017). They have quite different sizes and depot/nodes distributions to the instances our model has learned (uniformly distributed depot and 100 nodes), so the generalization ability is important for a good performance. In this case, CPL with 8 continuation solutions can further improve POMO and achieve a 4.95% average optimality gap. This result is better than the powerful DACT improvement solver with $t = 5k$ improvement steps for each instance, but is outperformed by DACT with the strongest setting ($t = 10k$ with 6 augmentations). The construction, improvement, and search methods are not necessarily competitors.

Table 6. Experimental results on TSPLib with 50 to 200 cities.

Instance	OR-Tools	Wu et al. (T = 3k, M=1k)	DACT (T=3k)	DACT (T=10k, Aug. $\times 4$)	AM-S (T=10K)	POMO	CPL (T=8)
eil51	2.35%	1.17%	1.64%	0.00%	2.11%	0.23%	0.00%
berlin52	5.34%	2.57%	0.03%	0.03%	1.67%	0.00%	0.00%
st70	1.19%	0.89%	0.44%	0.30%	2.22%	0.00%	0.00%
eil76	4.28%	4.65%	2.42%	1.67%	3.35%	0.74%	0.19%
pr76	2.72%	1.37%	1.02%	0.03%	2.84%	0.09%	0.02%
rat99	1.73%	8.51%	4.05%	0.74%	9.50%	4.62%	2.39%
KroA100	0.78%	2.08%	0.86%	0.45%	79.49%	1.05%	0.68%
KroB100	3.91%	5.78%	0.27%	0.25%	9.30%	0.78%	0.73%
KroC100	4.02%	3.17%	1.06%	0.84%	8.04%	0.38%	0.17%
KroD100	1.61%	5.00%	3.54%	0.12%	10.02%	1.92%	1.25%
KroE100	2.40%	3.29%	2.17%	0.32%	3.10%	1.39%	1.02%
rd100	3.53%	0.06%	0.08%	0.00%	1.93%	0.00%	0.09%
eil101	5.56%	4.61%	3.66%	2.86%	3.97%	0.16%	0.00%
lin105	3.09%	2.48%	3.41%	0.69%	32.13%	0.77%	0.70%
pr107	1.74%	3.87%	5.86%	3.81%	43.26%	1.35%	0.90%
pr124	5.91%	2.97%	1.56%	1.22%	4.41%	0.08%	0.08%
bier127	3.76%	3.48%	4.08%	2.46%	1.71%	4.31%	5.00%
ch130	2.85%	4.89%	6.63%	1.93%	2.96%	0.10%	0.02%
pr136	5.62%	6.33%	5.54%	4.54%	4.90%	0.74%	0.93%
pr144	1.28%	1.40%	3.44%	2.49%	8.77%	0.50%	0.66%
ch150	3.08%	3.55%	3.60%	1.23%	3.45%	0.44%	0.41%
KroA150	4.03%	4.51%	6.93%	3.91%	9.98%	0.79%	0.69%
KroB150	5.52%	5.40%	6.10%	2.82%	9.87%	1.94%	1.49%
pr152	2.92%	2.17%	4.48%	3.59%	13.47%	1.23%	0.87%
u159	8.79%	7.67%	6.84%	3.16%	7.38%	1.00%	0.97%
rat195	2.84%	9.90%	6.93%	4.99%	16.57%	9.60%	6.50%
d198	1.16%	4.99%	12.27%	8.75%	331.58%	21.9%	20.1%
KroA200	1.27%	7.01%	3.60%	1.25%	15.64%	2.05%	1.78%
KroB200	3.67%	7.05%	10.51%	5.66%	18.54%	4.25%	2.29%
Avg. Gap for [50,100)	2.93%	3.19%	1.60%	0.46%	3.61%	0.95%	0.43%
Avg. Gap for [100,150)	3.29%	3.53%	3.01%	1.57%	15.29%	0.97%	0.87%
Avg. Gap for (150,200]	3.70%	5.81%	6.81%	3.93%	47.39%	4.80%	3.89%
Avg. Gap for All	3.34%	4.17%	3.90%	2.07%	22.83%	2.15%	1.72%

Table 7. Experimental results on CVRPLIB instances with different sizes and distributions

Instance	(Depot, Nodes) Type	OR-Tools	DACT (T=5k)	DACT (T=10k, 6 aug.)	AM-S (T=10K)	POMO	CPL (T=8)
X-n101-k25	(R, R)	6.57%	2.09%	1.47%	32.95%	6.73%	3.95%
X-n106-k14	(E, C)	3.72%	2.93%	1.87%	6.78%	2.27%	2.15%
X-n110-k13	(C, R)	7.87%	1.43%	0.13%	3.15%	1.22%	0.89%
X-n115-k10	(C, R)	4.50%	3.29%	1.68%	7.52%	9.96%	4.33%
X-n120-k6	(E, RC)	6.83%	3.50%	2.38%	4.54%	9.32%	8.47%
X-n125-k30	(R, C)	5.63%	6.51%	5.44%	35.16%	5.78%	4.17%
X-n129-k18	(E, RC)	8.37%	2.93%	2.55%	4.00%	2.16%	1.02%
X-n134-k13	(R, C)	21.61%	6.98%	2.63%	20.13%	3.98%	2.29%
X-n139-k10	(C, R)	12.02%	2.54%	2.08%	4.30%	3.24%	2.63%
X-n143-k7	(E, R)	11.27%	7.80%	3.55%	8.88%	4.34%	2.72%
X-n148-k46	(R, RC)	7.80%	2.69%	2.22%	79.53%	9.78%	4.73%
X-n153-k22	(C, C)	8.01%	11.04%	6.53%	78.11%	14.4%	11.4%
X-n157-k13	(R, C)	2.57%	4.64%	3.12%	16.30%	8.66%	7.76%
X-n162-k11	(C, RC)	6.31%	4.43%	2.62%	6.37%	6.00%	5.93%
X-n167-k10	(E, R)	9.34%	5.37%	3.47%	8.41%	3.61%	3.08%
X-n172-k51	(C, RC)	10.74%	6.23%	3.41%	85.37%	10.7%	7.01%
X-n176-k26	(E, R)	8.99%	10.29%	5.93%	20.39%	10.6%	6.84%
X-n181-k23	(R, C)	2.94%	3.41%	2.08%	6.45%	5.57%	4.42%
X-n186-k15	(R, R)	7.75%	5.99%	4.94%	6.01%	6.58%	6.03%
X-n190-k8	(E, C)	6.53%	7.97%	6.73%	46.61%	6.68%	6.15%
X-n195-k51	(C, RC)	13.76%	7.00%	4.36%	79.26%	13.7%	7.98%
X-n200-k36	(R, C)	4.15%	5.93%	5.89%	26.25%	5.97%	4.87%
Average Optimality Gap		8.06%	5.23%	3.41%	26.66%	6.88%	4.95%

* Three types of depot positions: C-Central, E-Eccenric/Corner, R-Random.
* Three types of node distributions: R-Random, C-Clustered, RC-Mixed Random and Clustered.