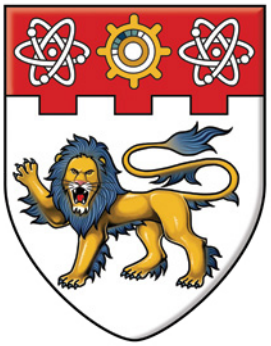


NANYANG TECHNOLOGICAL UNIVERSITY

METHODS IN MULTI-SOURCE DATA-DRIVEN TRANSFER OPTIMIZATION

BINGSHUI DA

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

METHODS IN MULTI-SOURCE DATA-DRIVEN TRANSFER OPTIMIZATION

BINGSHUI DA

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

2019

Abstract

In the global optimization literature, traditional optimization algorithms typically start their search process from scratch while facing a new problem of practical interest. That is to say, their problem-solving capabilities do not grow along with accumulated experiences or solved problems. Under the observation that optimization problems of practical interest seldom exist in isolation, ignoring the prior experience often implies the wastage of a rich pool of knowledge that can otherwise be exploited to facilitate efficient re-exploration of possibly overlapping search spaces. However, in practical settings, the ability to leverage such a rich pool of knowledge often yields substantial convergence speedup as well as cost-saving benefits. Given today's competitive need for high-quality solutions promptly, the necessity to adaptively reuse past experience is not hard to comprehend. Nevertheless, transfer learning has continuously drawn research attention during the years in the machine learning community, while only a handful of research works have been focused on knowledge transfer in optimization. Thus, in the present thesis, it is aimed to accelerate the optimization process on the task of practical interest by automatically selecting, adapting and integrating knowledge from past problems, under the recently introduced concept of *transfer optimization*.

In particular, inspired by transfer learning in supervised learning, learning generalizable probabilistic models for transfer optimization is first presented. Taking supervised signals from several related source probabilistic models, it is demonstrated that a more generalizable probabilistic model could be learned, capable of predicting high-quality solutions directly for combinatorial optimization problems drawn from different distributions. Subsequently, it is observed that in some real-world settings, some source probabilistic model can cause negative influence on the target optimization task, as it

is impractical to guarantee that all the diverse source models are beneficial to the task of practical interest. Therefore, a new transfer optimization paradigm, namely *adaptive model-based transfer*, is proposed. The proposed paradigm enables online learning and exploitation of similarities across different optimization problems. The experience on certain optimization task either takes the form of probabilistic model directly, or is encoded as a probabilistic distribution. By taking advantage of different source probabilistic models, this framework is able to automatically modulate the amount of knowledge needed to be transferred from multiple source tasks, hence the threat of negative transfer is minimized. By transforming various search spaces into a universal search space, the proposed framework can tackle discrete, continuous, as well as single- and multi-objective optimization problems. Finally, when the target optimization problem becomes computationally expensive, the aforementioned methods might not be practical to be deployed in real-world applications. Therefore, surrogate-assisted optimization is a promising optimization tool for computationally expensive problems in continuous domain. In order to adaptively take advantage of past experiences on solving various optimization tasks, a scalable multi-source surrogate-assisted transfer optimization framework is proposed, facilitating efficient global optimization on continuous optimization problems with high computational complexities.

Acknowledgements

Reaching the end of this long journey, I want to express my deepest gratitude to those who have helped me, encouraged me, and guided me during the past four years.

First of all, I want to thank my supervisor, professor Yew-Soon Ong. I would not have started this wonderful journey without this great opportunity; without his endless guidance and instructions, I would still be struggling figuring out what Ph.D research truly means. His enthusiasm to research always inspires me to go through ups and downs with energy and passion.

I would also like to express my gratitude to Abhishek Gupta, for countless inspirational and encouraging discussions. No matter what difficulties I met, he is always there to help me, and to provide constructive suggestions. It is also my honor to have a friendly and helpful research group: Feng Liang, Hou Yaqing, Zhai Yiteng, Wang Zhenkun, Yuan Yuan, Bali Kavitesh, Ray Lim, Qu Xinghua, Alvin Chan, and Lin Su Chit. Those memories of sharing office, intense discussions during group meetings, as well as countless group gatherings, are priceless to me. The knowledge they shared during discussions and group meetings are truly helpful, both in research and in daily life.

I thank SAP for the financial support throughout my entire PhD life. I am also indebted to my mentors in SAP, Shu Zhen, Lu Haiyun, Wang Chen, and Cai Jessie. Discussions with them not only motivated me towards research, but also broaden my eyes to the industry. I would also like to thank my helpful and knowledgeable colleagues in SAP for their support and encouragement: Ni Peng, Wang Wenya, Zheng Xin, Wang Jin, Li Jianshu, Guo Qing, Han Jianglei, Steve Lee, and Madhur Mayank Sharma.

Besides, I want to thank all my friends, especially Tang Jing, Xu Xin, Luo Jianping, Li Hao, Xu Qingzheng, Peng Xingguang, Zhang Zhen, etc, for the joyful times we

have been through together. I also want to thank Kesavan Asaithambi for his technical support during these four years.

Last but not least, special thanks to my beloved girlfriend, Qian Hangwei, and my family for their constant love and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In machine learning, the idea of taking advantage of *available data* from various *source* tasks to improve the learning of a related *target* task has achieved significant success, under the notion of *transfer learning* [1–3]. For example, pre-trained BERT model [4] is widely reused on various natural language processing tasks. The definition of transfer learning, according to [2], is written as follows:

Definition 1.1. Given a source domain[1] $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_S$ or $\mathcal{T}_S \neq \mathcal{T}_S$.

Under this definition, various forms of transfer learning have been studied in the literature. Notable works include but are not limited to domain adaptation [5, 6], online transfer learning [7], as well as transfer learning in deep neural networks [8] and reinforcement learning [9, 10]. Nevertheless, research progress tied to the concept of knowledge transfer has largely been restricted to the domain of predictive analytics. In contrast, for the case of optimization problems, where the search typically starts from scratch with zero prior data, similar efforts of automatic knowledge transfer have been found to be rare. That is to say, the capabilities of these optimization solvers do not automatically grow with accumulated experience.

[1] A domain $\mathcal{D}$ refers to a feature space and its density distribution

Nevertheless, humans can usually exploit a pool of knowledge gathered from various experience, and seamlessly generalize the acquired knowledge to tackle related tasks, in spite of learning or optimization tasks (as shown in Fig. 1.1). For instance, when learning to drive a car, years of past experience with basic motor skills, typical traffic patterns, etc., are drawn on [1]; when students trying to solve problems during the exams, generalizing knowledge from homework would be essential for a good grade. It is also observed that optimization problems of practical interest seldom exist in isolation [11]. Any practically useful system, especially in industrial settings, must be expected to tackle many related optimization problems over its lifetime. Many of these problems will either be repetitive, or at least share some domain-specific similarities. Therefore, in many real-world settings involving high monetary and/or temporal cost of evaluations, *ignoring the availability of related experiences on source optimization problems implies a wastage of resources in re-exploring essentially overlapping search spaces*.

The push to incorporate such cognitive capability into optimization solvers is further fueled by the ongoing era of data democratization, where technologies such as cloud computing and the Internet of things (IoT) enable direct access to and easy storage of large volumes of diverse incoming information streams over time. In such settings, effectively capturing and harnessing the fruitful *generalizable* knowledge can play a significant role in enhancing the efficacy of decision-making processes. Thus, given the present-day demands for achieving high-quality solutions within strict time constraints, the need to effectively exploit the knowledge learned from past experiences, is well recognized [12]. In this thesis, it is attempted to make a step further under the general theme of *transfer optimization*, which automatically extracts and transfers knowledge across distinct problem-solving experiences.

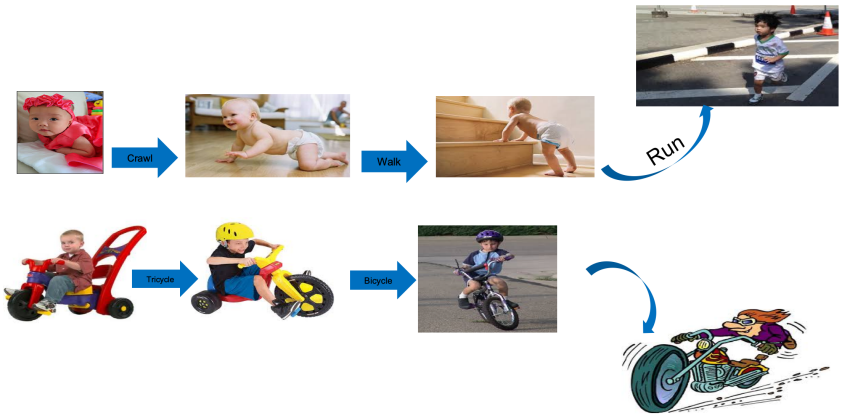


FIGURE 1.1: Anthropic example of human knowledge transfer/reuse.

1.1 A Formalization of the Transfer Optimization Paradigm

In the present thesis, knowledge transfer is studied on a different class of "optimization" problems. Without the loss of generality, the goal for optimization is defined as:

$$\max \ f(x): \ x \in \mathcal{X}, \tag{1.1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is the objective function, and $\mathcal{X}$ is the search space. For clearance, throughout the thesis, optimization problem and optimization task are used interchangeably.

Consider a series of K optimization tasks, denoted as $\mathcal{T}_1, \cdots, \mathcal{T}_K$. The key purpose is to integrate optimization solvers human-like cognitive ability to automatically learn from experience, and generalize the learned knowledge to solve related tasks more efficiently. In this thesis, knowledge transfer in optimization is defined as means to facilitate improved performance on a target task, say $\mathcal{T}_K$, given a knowledge base $\mathcal{M} = \cup_{k=1,\cdots,K-1} m_k$ gathered from other optimization efforts, where m_k can either be the function evaluations on $\mathcal{T}_k$ [13], higher-order building-blocks [14], or probabilistic models [15]. For a minimization problem defined in Eq.(1.1), *transfer optimization*

facilitating performance speedup on $\mathcal{T}_K$ can be measured as

$$Q_t(\mathcal{T}_K|\mathcal{M}) - Q_t(\mathcal{T}_K) \geq 0, \tag{1.2}$$

where $Q_t(\mathcal{T}_K)$ is the algorithmic efficiency, which can simply be defined as $Q_t(\mathcal{T}_K) = f_K(\mathbf{x}^*) : \mathbf{x}^* \in \mathbf{X}_K^t \wedge \forall \mathbf{x} \in \mathbf{X}_K^t, f_K(\mathbf{x}) \leq f_K(\mathbf{x}^*)$ in 't' time-steps, and $Q_t(\mathcal{T}_K|\mathcal{M})$ is the algorithmic efficiency conditioned on knowledge base $\mathcal{M}$. Note that as the optimization tasks can accumulate over time, the practical applicabilities of transfer optimization algorithms will be greatly enhanced with the capability of handling multiple sources at the same time. Harmful (negative) knowledge incorporation is caused if the Eq.(1.1) is not satisfied, often referred as negative transfer in this thesis.

According to this definition, two main categories of realizations that shed light on the range of ways in which transfer optimization can be put to use in practical settings are listed in the following.

- *Sequential Transfer*: For sequential transfer optimization, $\mathcal{T}_1, \cdots, \mathcal{T}_{K-1}$ have already been addressed previously, and the target optimization, $\mathcal{T}_K$, is the only task of practical interest. Therefore, the aim for sequential transfer is to speed up the optimization process on $\mathcal{T}_K$ only.

- *Multi-task*: Different from sequential transfer, multitasking caters to distinct tasks with equal priority occurring concurrently [16]. Therefore, the fruitful knowledge can be generated continuously in the common knowledge base, which is immediately accessible to all tasks in the multitasking environment. Therefore, the aim for multitasking is to enhance the search process for all the optimization tasks simultaneously. One of the metrics to quantify this performance improvement of multitasking paradigm has recently been reported in [17].

In this thesis, the focus is on the first category, i.e., *sequential transfer optimization*, with a rich pool of knowledge extracted from multiple sources. Specifically, transfer optimization is studied in following scenarios, with the aim to incorporate human-like capabilities into optimization solvers, so that the solvers can automatically select, adapt, and integrate knowledge from other problems for improved problem-solving.

1.1.1 Learning Generalizable Model for Combinatorial Optimization via Knowledge Transfer

Algorithm design is typically a laborious process, and often requires decades of efforts from domain experts. Recent research progress [18] has shown that neural networks, especially sequence-to-sequence models, are able to automatically develop their own heuristics for combinatorial optimization problems by generalizing the experience from solving millions of optimization problems drawn from a specific problem distribution [18]. To elaborate, the network learns the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence. However, it is recently reported that these well-trained models can not generalize well out of training distributions [19, 20]. Take traveling salesman problem (TSP) as an example. If a model trained on TSP instances with 20 nodes is evaluated on a TSP instance with 50 nodes, the generalization performance is reported to degrade, compared with the model trained specifically on TSPs with 50 nodes. Such performance degradation severely diminishes the practical applicability of these models in real-world scenarios, as little can be said a priori whether the new problem of practical interest is drawn from the same training distribution or not.

Research advances in transfer learning found that by utilizing supervised signals from a set of well-trained models, a more generalizable target model could be learned [8, 21]. This target model with enhanced generalization ability can either have improved performance over the source models, or is capable of tackling a wider range of machine learning tasks [22]. Motivated by this key finding, learning a generalizable target model from multiple related source models is first studied in this thesis, with the aim to enhance the practical applicability of the target model in real-world settings. Specifically, a novel neural network structure is proposed, catering to efficient knowledge transfer from diverse source models. Extensive empirical studies are conducted on various TSPs as well as knapsack problems (KPs) in Chapter 3, to demonstrate the effectiveness of the proposed model compared with other knowledge transfer based approaches.

1.1.2 Curbing Negative Transfer Online

One of the key assumptions for the previous study is the high similarity between source and the target optimization tasks (the same type of optimization problems, i.e., TSPs or KPs). When the optimization task of practical interest is drawn from a completely different environment or distribution, whether or not the learned source model is beneficial to the learning of target task remains unknown. With the rapid development in modern computing platforms, increasing number of diverse source optimization tasks are readily available for reuse. With no prior knowledge on the latent synergies between source and target optimization tasks, harmful knowledge incorporation could be a real threat, thus highlighting the importance of conducting knowledge transfer during the course of optimization process.

Initial efforts in the context of online knowledge transfer for optimization have explored a direct insertion scheme. For example, Louis and McDonnel [11] showed the benefits of periodically injecting a certain number of solutions drawn from intermediate populations of related source optimization problems into the the target evolutionary search. However, such simple insertion scheme can give rise to the danger of negative knowledge transfer, as the qualities of injected solutions remain unknown to the target task. In order to minimize the threat of negative transfer, an adaptive model-based online transfer scheme is proposed in Chapter 4. More importantly, theoretical behavior of the proposed scheme is also analyzed, substantiating its positive influences on optimization performance. The practical efficacy of an instantiation of an adaptive transfer evolutionary algorithm is demonstrated on a series of numerical examples, spanning discrete, continuous, as well as single- and multi-objective optimization.

1.1.3 Surrogate-Assisted Transfer Optimization

Real-world problems can be extremely expensive. For example, in aerodynamic design, carrying out one simulation to evaluate the performance for a given structure can take over 10 hours. Another popular example is optimally tuning the hyperparameters for a deep neural network, in which training a deep neural network using a specific

hyperparameter set can cost hours or even days. When the optimization problem of interest is computational expensive, the aforementioned methods, incorporating iterative evaluations on the optimization problem(s), become impractical to be deployed.

Taking this cue, surrogate-assisted optimization is proposed in the literature, to reduce the direct evaluations on continuous optimization problem with high computational complexity. The basic rational of a surrogate-assisted optimization algorithm is to construct a cheap-to-evaluate surrogate model f' to approximate the expensive optimization task f, hence optimization search can be conducted on the surrogate model f' in order to find a promising candidate solution to be evaluated on f. If the surrogate model f' approximates f sufficiently well, optimizing the surrogate model f' is essentially equivalent to optimizing the expensive optimization task. Therefore, the approximation quality of the surrogate model is crucial to the convergence property of the corresponding surrogate-assisted optimization algorithm.

One of the most commonly used surrogate models is Gaussian process. In order to reuse previous knowledge, multi-output/transfer Gaussian processes are often applied to increase the regression performance on the target surrogate model [23, 24] by utilizing data from potentially related source tasks. While many problem-solving exercises can accumulate over time, poor scalability of traditional transfer Gaussian process models becomes the bottleneck for efficient knowledge transfer. In order the tackle such key limitation, a scalable transfer Gaussian process model is proposed in Chapter 5, in which data from multiple source domains is re-utilized to enhance the regression performance. Further, the proposed model is deployed in surrogate-assisted optimization, with problem-solving experiences drawn from multiple source optimization tasks.

1.2 Research Contribution

The core research contributions of this dissertation are three-fold as listed below.

- Motivated by supervised learning, a novel method to learn a single generalizable probabilistic model from multiple different but related source probabilistic models is proposed. By transferring knowledge from a set of well-trained source models, the proposed target model can predict high-quality solutions for optimization problems sampled out of training distributions.
- In order to deal with transfer optimization with no prior knowledge available on the underlying synergies between optimization tasks, a novel adaptive model-based online transfer scheme is thus proposed. Theoretical analysis on this scheme is also conducted, demonstrating that the knowledge-enhancement scheme guarantees to facilitate global convergence characteristics.
- Surrogate-assisted transfer optimization is studied to deal with computationally expensive optimization tasks. With accumulated experiences on source optimization tasks, a novel scalable surrogate model in the literature of Gaussian process is proposed, in order to speed up the optimization process on the task of interest.

The contributions described above have led to the following publications:

- **Submitted to Knowledge-Based Systems**: A journal paper entitled "Generalizable Neural Optimization via Knowledge Distillation and Transfer" was submitted to Knowledge-Based Systems.
- **Accepted**: A journal paper that has been accepted for publication in IEEE Transactions on Cybernetics in 2018 entitled "Curbing Negative Influences Online for Seamless Transfer Evolutionary Optimization" [15].
- **Accepted**: A journal paper that has been accepted for publication in Knowledge-Based Systems in 2018 entitled "Fast Transfer Gaussian Process Regression with Large-Scale Sources" [25].

1.3 Thesis Organization

The structure of this thesis is organized as follows:

- In Chapter 2, a comprehensive literature review on recent advances towards transfer optimization is conducted, to provide the context of the present thesis.

- Chapter 3 presents the proposed strategy to learn a generalizable neural network model from multiple related well-trained source probabilistic models. Specifically, case studies on the popular TSPs and KPs are conducted, and it is found that the knowledge-enhanced probabilistic model can solve a wider range of TSPs and KPs compared to the source models.

- In Chapter 4, a novel adaptive model-based online transfer scheme is proposed to curb negative influences online from multiple sources. With no prior knowledge of the underlying synergies between optimization tasks, this knowledge-enhancement scheme is theoretically guaranteed to facilitate global convergence characteristics on the target optimization task. Rigorous experimental verification of the proposed scheme is conducted on a diverse test suite, showcasing the importance of online adaptive knowledge transfer during the course of optimization process.

- In Chapter 5, transfer optimization on target tasks with high computational complexity is studied. Thus, a scalable surrogate model is proposed in this chapter, to handle the issue of rapid growing complexity with increasing number of sources available.

- Summary and future directions of this thesis are presented in Chapter 6.

Chapter 2

Literature Review

In this chapter, preliminaries and a brief review of existing approaches in the literature are presented that uphold the notion of "transfer optimization".

Traditional optimization algorithms typically start their search process from scratch while facing a new problem (target problem). Hence, their problem-solving capabilities do not grow with accumulated experiences. However, optimization problems of practical interest seldom exist in isolation [11, 15], and useful information often exists between optimization problems. Ignoring prior experience implies re-exploring possibly overlapping search space and wasting a rich pool of reusable knowledge. In practical settings, the ability to leverage such a rich pool of knowledge often yields substantial convergence speedup as well as cost-saving benefits [26]. Indeed, the ability to take advantage of acquired knowledge often sets apart an expert from a novice. Therefore, the key objective of transfer optimization is to incorporate human-like capabilities into optimization solvers, so that the the solvers can automatically select, adapt, and integrate knowledge from a rich pool of knowledge accumulated over time for improved problem-solving experience.

Over the years, a handful of research advances for global optimization under the theme of transfer optimization generally fall into the following two categories: (1) store a pool of solutions from previous problems that can be subsequently reused to guide

the search on the optimization problem of interest [11, 14]; (2) directly reuse the (probabilistic) models that are built during past problem-solving exercises [15, 27]. The proposed approaches in the present thesis also fall into these two categories.

2.1 Predicting the Optimum for Combinatorial Optimization Problems

Combinatorial optimization is one of the fundamental problems in computer science [20, 28, 29]. In many of such problems, finding the exact optimal solution is often not computationally tractable. For example, the traveling salesman problem (TSP) involves finding the shortest possible route that visits each city and returns to the origin city, given a list of cities and the distances between each pair of cities. TSP is known to be NP-hard, and the complexity of finding the exact optimal by using the Held-Karp algorithm [30] is $\mathcal{O}(n^2 2^n)$, where n is the number of cities. Thus, instead of using exact solvers, heuristics or approximation algorithms are usually preferred in practical settings to tradeoff performance for computational tractability [31].

Traditional optimization algorithms for combinatorial problems, including exact solvers, heuristics, and approximation algorithms, typically start their search process from scratch while facing a new task (referred as the *target* problem). Hence, their problem-solving ability does not grow with accumulated experiences. However, optimization problems of practical interest seldom exist in isolation [11, 15], and useful information often exists between related tasks. Over recent years, there have been a handful of methods initiating cross-task learning in optimization. Different modes of knowledge transfer have been proposed in this regard, including the direct injection of raw solutions [11, 14, 16], biasing search through prior solution distribution models (that provide useful hints on where to search on a related task [15, 27]), and transfer regression in surrogate-assisted optimization [13, 32]. While positive results have been reported in various domains, for e.g., in operations research, engineering optimization, neuro controller design, etc., it is found that the learning capabilities of methods of the aforementioned type are typically limited by the relative simplicity of

the machine learning models used. In contrast however, there also exists a separate and emerging class of approaches that have attempted to unleash the expressive power of deep learning for solving combinatorial optimization problems. By learning from millions of optimization examples drawn from a specific distribution (during training), deep sequence-to-sequence models [33, 34] are essentially able to discover their own (neurally encoded) problem-solving heuristics, allowing them to directly predict high-quality solutions on-the-fly for new combinatorial instances belonging to the same distribution [18, 29, 35–37]. In this context, one of the most notable neural network models is the Pointer Network [18, 35].

While (deep) neural optimization methods have promised fast approximations to NP-hard problems, it has also been reported that these models are generally unable to generalize outside of the training distribution [20]. Taking TSP as an example, if a model trained on TSP instances with 20 nodes is tested on a TSP instance with 50 nodes, the generalization performance is poor, especially compared to a model trained specifically on TSPs with 50 nodes. The practical applicability of the approach is thus brought under question, as little can be said in advance about the true underlying distribution of a target problem. With the aim of enhancing the practicality of neural combinatorial optimization, in Chapter 3, a novel approach to distill and transfer knowledge contained in multiple models is therefore proposed, trained on different problem distributions, into a single neural network which may serve as a more general-purpose problem solver.

To this end, it is however noted that, directly transferring knowledge to a target network may still lead to poor generalization performance (as shall experimentally be shown in Chapter 3), as different source models may have discovered conflicting search behaviors, confounding optimization process. In order to cope with this issue, a novel neural network architecture is also proposed for transferring knowledge from multiple source models. To elaborate, a task-specific module is designed for each source model, as a result of which different source models are able to teach the target network through representation learning. It is also noted that complex models such as transformer [36] are demonstrated to have more generalizable capabilities compared with simple Pointer Network. Nevertheless, a different direction in the field of neural combinatorial optimization is explored in the present thesis, where knowledge from different sources is

utilized to enhance the generalization ability of a simple neural network model.

2.2 Online transfer optimization techniques

The transfer optimization strategy proposed in Chapter 3 assumes that the source models are related and will be beneficial to the learning of the target model on a new test problem drawn from an unknown distribution. However, given the rapid advancements in modern computing platforms such as cloud and the Internet of Things (IoT), which give rise to large-scale data storage and seamless communication facilities, the practical viability of such method is hard to tell. Various types of problem-solving experiences are mixed together, and negative transfer can be a real threat if we naively transfer knowledge from all the source models. This finding motivates increasing research attention towards online knowledge transfer during transfer optimization.

Initial efforts in the context of online transfer optimization have however explored a direct solution injection scheme. For example, Cunningham and Smyth [38] proposed to directly inject known good schedules into a target scheduling problem. Similarly, a family bootstrapping approach is put forward in [39] for neuro-evolutionary robot controller design, where the optimized solutions of a common source task are used to bias the initial population of a target task. Likewise, Louis and McDonnel [11] showcased the benefits of periodically injecting a certain number of solutions drawn from intermediate populations of related source optimization problems into the the target evolutionary search.

In addition to direct *genetic transfer* from source to target evolutionary searches [40], higher-order model-based transfer optimization algorithms are also explored in the literature. In [27], a heuristic criterion is used within a specific class of combinatorial problems for retrieving related source probabilistic models from a database to bias the target optimization search. In [14, 41], the model takes the form of a positive semidefinite (PSD) matrix that induces a modified (biased) distance metric for graph-based clustering cum sequencing problems. More recently, Feng *et. al.* [42] proposed to learn an iterative mapping from a continuous source domain to the target search

space through a single layer denoising autoencoder. In addition to the above, [43–45] demonstrated that building blocks of knowledge in the form of code fragments (i.e., trees or sub-trees of computer programs evolved by genetic programming) extracted from small-scale problems, could be re-used while solving more complex, large-scale problems.

Based on an initial study, it is found that that the aforementioned approaches typically make use of large databases to store past solutions. Thereafter, the *case by case assessment* required to select the most appropriate candidate solutions gradually becomes prohibitively time consuming as the database grows in size; often referred to as the *swamping problem* [46, 47]. Challenges are particularly exacerbated by the fact that little can be said *a priori*, with any degree of certainty, about the *similarity* between black-box optimization problems (due to the lack of target data available before the onset of the search). *As a result, the threat of negative transfer acts as a major impedance when dealing with multiple sources of knowledge, where some sources may be more relevant than others*. In contrast, we humans can effortlessly draw useful information from a vast pool of knowledge acquired over a lifetime of experiences. Interestingly, even if two distinct tasks appear unrelated on the surface, humans are often able to uncover any latent synergies that may be present.

The observations above serve as the main motivations for the proposed paradigm in Chapter 4. With the goal of incorporating human-like problem-solving capabilities into optimization solvers, a novel *transfer evolutionary computation* paradigm capable of *online source-target similarity learning* is proposed as a way to curb the risks of negative transfer on the fly. Given the increasing popularity of modern computing platforms such as the cloud and the IoT, the practical viability of such a paradigm (from a hardware perspective) is little in doubt. In particular, this work focuses on the use of population-based *evolutionary algorithms* (EAs) as they not only provide significant flexibility in dealing with a wide range of discrete and continuous optimization problems, but are also amenable to be hybridized with various learning strategies. In the proposed approach, the population distribution of *elite* solutions is captured from some source optimization task in the form of a probabilistic model, that is then stored for future usage. These

probabilistic knowledge building-blocks serve to bias the search on a *related* target task towards solutions that have been shown to be promising.

It is also noted that, despite sequential transfer, online knowledge transfer has also been extensively studied in multi-task optimization. The first attempt in the evolutionary computation literature explored the implicit parallelism of population-based search, and proposed a new paradigm named multifactorial optimization [16, 48, 49]. Diverse optimization problems are optimized in the same evolutionary search engine, enabling the implicit knowledge transfer between problems via the interchange of genetic materials. Inspired by this work, Yang et al. [50] proposed a multi-task evolutionary method tackling multi-objective operational indices optimization problems. Li et al. utilized evolutionary multitasking framework to simultaneously optimize multiple sparse reconstruction tasks in [51]. Enhanced performance on signal reconstruction and hyperspectral image unmixing demonstrate the effectiveness of online knowledge transfer. In [52], Tang et al. proposed a group-based multitask evolutionary algorithm that groups tasks of similar types and transfers genetic information only within the groups, to prevent potential negative transfer. Feng et al. [53] have also explored learning an iterative mapping between different optimization tasks in the evolutionary multitasking paradigm. More recently, Kavitesh et. al. [54] proposed to learn inter-task relationships online, so as to adaptively modulate the amount of transferred knowledge between tasks to prevent negative transfer.

2.3 Surrogate-Assisted Transfer Optimization

In practice, when dealing with an optimization problem in real-world settings, evaluating the objective value can be non-trivial. A single function evaluation can cost hours or even days. Thus, surrogate-assisted optimization is extensively studied to deal with computationally expensive optimization problems. General procedure of a surrogate-assisted optimization algorithm is shown in Fig. 2.1, and the pseudo-code is listed in Algorithm 1.

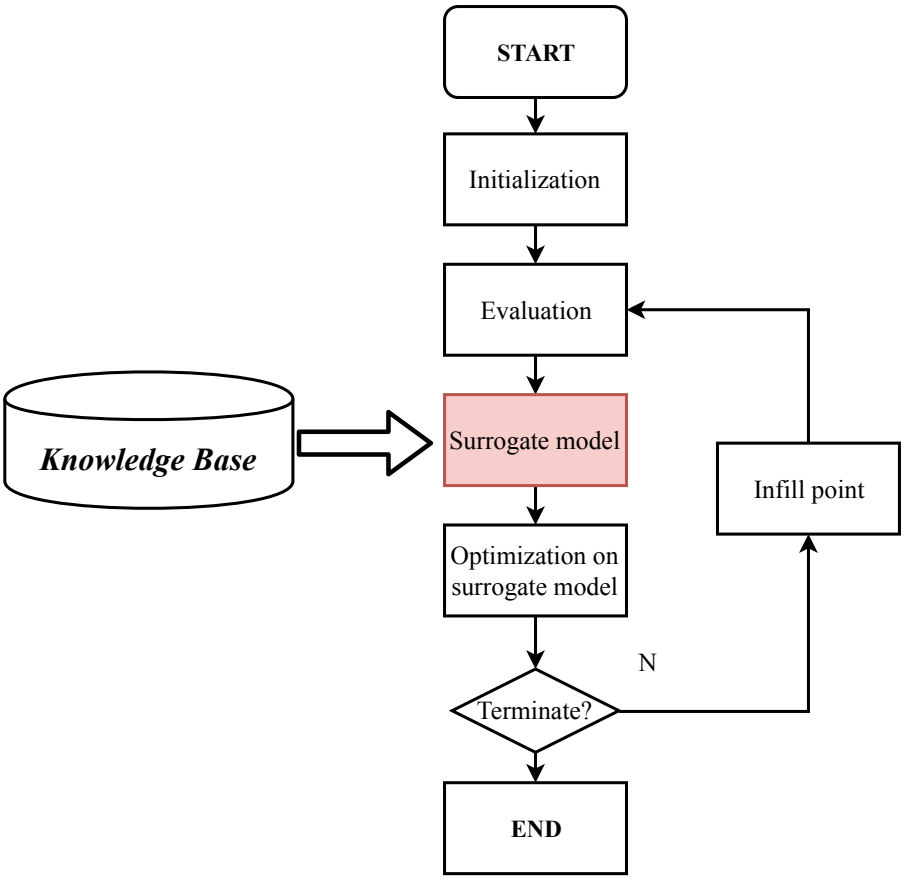


FIGURE 2.1: General process of surrogate-assisted optimization.

Algorithm 1: General pseudo-code for surrogate-assisted search algorithm

1: Sample D_0 from f according to initial space-filling experimental design
2: Construct surrogate model
3: **for** $t = 1, 2, \cdots$ **do**
4: Find the next sampling point(s) based on the surrogate model built upon D_{t-1}
5: Evaluate the original objective function on the sampling point(s)
6: Augment the dataset to D_t
7: Update the surrogate model
8: **end for**

In the literature of evolutionary computation, surrogate models (also known as meta-models) have been widely used to tackle computationally expensive optimization problems in continuous domain. The corresponding evolutionary algorithm (EA) is often known as surrogate-assisted EA (SAEA). A variety of computational models, including polynomials, Gaussian process (GP, also known as kriging), neural networks, active learning, as well as boosting have been deployed as surrogate models in SAEA [55–57]. For example, Ong et al. [58] proposed a framework of SAEA coupled with a feasible sequential quadratic programming solver in the spirit of Lamarckian learning, and local surrogate models using radial basis functions are constructed to increase the quality of approximation in local regions. A max-min SAEA for robust engineering design was proposed in [59], to search for designs that have the best worst-case performance in the presence of parameter uncertainty. Later, Zhou et al. [60] proposed to combine global and local surrogate models to accelerate evolutionary optimization. However, it is later shown in [61, 62] that an EA may benefit from approximated error from surrogate modeling. These facts partially demonstrate the importance of the tradeoff between exploration and exploitation. Exploitation means sampling the next point(s) with high expected performance, while exploration encourages sampling regions with high uncertainty. Probabilistic surrogates such as GP [32, 63] are hence becoming the most widely used surrogates when the uncertainty information is crucial. Recently, ensemble machine learning models have also been proved in [64] to be promising in providing the uncertainty information, where variance of the predictions made by different base learners can be used to estimate the degree of uncertainty. To balance exploitation and exploration, a number of individual-based strategies, namely infill sampling criteria [63] (also known as acquisition functions in Bayesian optimization), are deployed, including probability of improvement (PI), expected improvement (EI), and Gaussian process lower (upper) confidence bound (GP-LCB).

Tremendous research attention has also been focused on solving computationally expensive multi-objective optimization by utilizing SAEA. Initial work in [65] directly extends the single-objective *efficient global optimization* algorithm, which uses a design-of-experiments inspired initialization procedure, and learns a GP model to approximate the function landscape. More recently, a multi-objective infill sampling

criteria was proposed in [66], where the infill sampling is considered as a bi-objective problem that simultaneously minimizes the predicted fitness and the estimated variance of the predicted fitness. For more detailed review of recent advances of SAEA, readers may refer to [57].

It is also noted that Bayesian optimization has emerged as a powerful solution for various design problems. In Bayesian optimization, GP is widely applied as the surrogate model, as GP model provides predictive mean as well as variance during predictive analysis. Bayesian optimization on hyperparameter tuning for machine learning models has drawn increasing research attention over the years [13, 67–69]. The tuning of model hyperparameters is often a "black art" requiring expert experience, while Bayesian optimization can reach or *surpass* human expert-level optimization for many algorithms. This success has made Bayesian optimization a crucial player in the current trend of "automatic machine learning". Detailed reviews on recent progress towards Bayesian optimization have recently been published in [70].

Similar to SAEA, acquisition functions, such as PI, EI, and GP-LCB, are also designed based on the predictive mean and uncertainty to tradeoff exploitation and exploration for Bayesian optimization. In particular, the analytical behavior of GP-LCB is analyzed, proving that the cumulative regret is bounded in terms of maximal information gain [71]. Knowledge transfer is also gaining its popularity in Bayesian optimization. Function evaluations on previously solved optimization tasks can be directly reused to construct multi-task/transfer GP, as shown in Fig. 2.1, during surrogate modeling process. Improved optimization performance has been reported in [13, 72]. Nevertheless, with the accumulated experiences, the poor scalability of traditional multi-task/transfer GP becomes the bottleneck to scale multi-task/transfer Bayesian optimization. In order to deal with this issue, a scalable transfer GP model is proposed in Chapter 5, in which data from multiple source domains is re-utilized to enhance the regression performance. Further, the proposed model is applied as a surrogate model in Bayesian optimization, with problem-solving experiences on multiple source optimization tasks available. To have a better understanding on recent progress towards transfer Gaussian process, a brief literature review is introduced in the following.

2.3.1 Transfer Gaussian Process

GP is a stochastic process wherein any finite subset of random variables follows a joint multivariate Gaussian distribution. It selects a prior distribution over the underlying function, conditions this distribution over the observations (training data), and uses the posterior to make predictions. An attractive property of GP is that it provides predictive mean as well as the associated uncertainty during predictive analysis. For details of GP, readers may refer to [73].

The idea of transfer learning in GP is that information shared between the tasks leads to improved generalization performance on the target task in comparison to learning the target task individually [2, 74, 75]. When using a GP for multiple distinct but related outputs, the problem often reduces to developing a prior (mainly determined by the covariance function) that expresses correlations between the outputs. A number of different covariance functions for multi-task GPs have been proposed in [74, 76–78]. For example, in [74, 75], the authors encode the inter-task correlations in a PSD matrix, with the entry in the ith and jth column capturing the degree of relatedness between the ith and jth tasks. Detailed reviews on the subject have recently been published in [79, 80].

As opposed to symmetric transfer in multi-task learning, less research attention has been focused on asymmetric transfer via transfer Gaussian process (TGP). In [81], Cao et al. proposed TGP to adaptively transfer knowledge from a single source task to improve the performance of the target task by learning source-target similarity. Leen et al. [78] combines the latent decision margins of multiple GPs operating on the source tasks with the latent decision margin of the target task. In [82], Wang et al. proposed to model the source task, the target task, and the offset between, using GP models, based on the assumption that there is some smoothness in the offset over the input domain. Taking advantage of deep GP, Kandemir [83] adopted a two-layer feed-forward deep GP [84] as the task learner of source and target domains. More recently, Wagle and Frew [85] proposed forward adaptive TGP, in which the training of source task is decoupled. In [24], Wei et al. studied multi-source transfer learning problems by *stacking* all the source and target models. A similar stacking procedure was also adopted in [32],

with the transfer learning GP model applied to enhance the efficiency of Bayesian optimization. Further, Wistuba et al. [69] proposed to combine source and target GP models via ensemble techniques, thus the final model is a weighted sum of all surrogates.

However, one of the major problems of the above transfer learning methods is the computational complexity during training and prediction, which scales cubically with the number of observations. Given the large amount of observations that can be available from source tasks, practical use of such methods becomes problematic, even with a small number of target training data. To reduce the overwhelming computational burden, different acceleration methods have been proposed in the literature, primarily focusing on symmetric transfer in multi-task GP, taking advantage of low-rank approximation to the full covariance matrix. The pioneering work proposed in [74] uses Nyström approximation of the kernel matrix in the joint marginal likelihood. Later, several different low-rank approximation methods have been proposed in [77, 86], which are strongly related to the partially independent training conditional [87] and fully independent training conditional [88] approximations for a single-task GP. These approximation methods cut the computational complexity to scale linearly with the number of observations. More recently, various methods taking advantage of variational inference [89–91] are proposed in the literature.

Nevertheless, it is noted that low-rank approximation and variational inference based methods generally have a key limitation: for quick-varying functions with significant local structures, the complete set of local training observations may be the most compact representation than any low-rank approximation. Recent study in [92] shows, in single-task GP models, similar local expressiveness can be maintained by using aggregation models. However, to the best of our knowledge, no aggregation models have been proposed to tackle transfer learning problems. Therefore, in Chapter 5, a principled aggregation model is proposed to deal with transfer learning problems efficiently, especially those with large-scale source data.

Chapter 3

Learning Generalizable Models for Transfer Optimization

Combinatorial optimization is one of the fundamental problems in computer science [20, 28, 29]. In many of such problems, finding the exact optimal solution is often not computationally tractable. Traditional optimization algorithms for combinatorial problems, including exact solvers, heuristics, and approximation algorithms, typically start their search process from scratch while facing a new problem (target problem). Hence, their problem-solving capabilities do not grow with accumulated experiences. Optimization problems of practical interest seldom exist in isolation [11, 15], and useful information often exists between optimization problems. Under these key observations, recent research progress in deep learning reveals that a sequence-to-sequence model is able to automatically discover its own heuristic for combinatorial optimization by training on millions of synthetic examples drawn from a specified distribution over problems [33, 34]. By accumulating experience from solving thousands, even millions of problem instances drawn from a specific distribution, deep learning models, specifically sequence-to-sequence models, are proved to be able to directly predict high-quality solutions for new combinatorial problems drawn from the same distribution [18, 19, 29, 35, 37].

While offering a promising new avenue to NP-hard problem-solving, these models

continue to suffer from the usual shortcoming of machine learning, in that their performance expectedly degrades when faced with new test cases that lie outside the regime of the training data. As a result, the practical applicability of these methods decreases, as little can be said a priori about whether a new target optimization task belongs to the training distribution or not. Thus, in this chapter, it is aimed to increase the generalization ability of these neural optimization approaches to out of distribution examples, gradually moving towards an envisioned theme of *General Optimization Intelligence*. In particular, a simple yet efficient strategy is put forward to distill and transfer the knowledge from multiple related source models, which are trained on different problem distributions, to a single network architecture. To this end, it is however noted that, directly transferring knowledge to a single target network may still lead to poor generalization performance (as shall experimentally be shown in this chapter), as different source models may have discovered conflicting search behaviors, confounding optimization process. In order to tackle this issue, a novel neural network architecture for transferring knowledge from multiple source models is proposed in this chapter. To elaborate, a task-specific module for each source model is designed, as a result of which different source models are able to teach the target network through representation learning. It is also noted that complex models such as transformer [36] are demonstrated to have more generalizable capabilities compared with simple Pointer Network. Nevertheless, a different direction is explored in the field of neural combinatorial optimization, where knowledge from different sources is utilized to enhance the generalization ability of a simple neural network model. Superior empirical performance on different types of combinatorial optimization problems demonstrates that the proposed model has stronger generalization performance on a wider range of distributions. To conclude, the contributions of this chapter are summarized as follows:

- It is proposed to distill and transfer knowledge from multiple source models to a single target network, in order to increase the applicability of the target neural networks for solving combinatorial optimization problems.
- A novel yet simple neural network architecture is proposed, enabling the knowledge from all source models to be captured in the form of high-level representations.

- Extensive empirical studies on two classes of combinatorial optimization problems are conducted to demonstrate the effectiveness of the proposed model compared to other neural optimization approaches.

The rest of this chapter is organized as follows. In Chapter 3.1, a brief introduction on Pointer Network is presented, which serves as the key element of this work. Based on that, a novel neural network structure for efficient knowledge transfer is proposed in Chapter 3.2. Experimental studies on two types of combinatorial optimization problems are presented in Chapter 3.3.

3.1 Pointer Network

In this section, the pointer network (PN) [35] is briefly reviewed. Take traveling salesman problem (TSP) in a 2D Euclidean space as an example. A simple demonstration of TSP is shown in Fig. 3.1. Given an input graph, represented as a sequence of n cities in a two dimensional space $s = \{\mathbf{x}_i\}_{i=1}^{n}$ where each $\mathbf{x}_i \in \mathbb{R}^2$, it is aimed to find a permutation of the points $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_n)$, termed as a tour, that visits each city once and has the minimum total length. The total length of the tour is defined as

$$L(\boldsymbol{\pi}|s) = \|\mathbf{x}_{\pi_n} - \mathbf{x}_{\pi_1}\|_2 + \sum_{i=1}^{n-1} \left\|\mathbf{x}_{\pi_i} - \mathbf{x}_{\pi_{i+1}}\right\|_2, \tag{3.1}$$

where $\|\cdot\|_2$ denotes l_2 norm.

The structure of the PN is shown in Fig. 3.2a. It is applied to define a probability distribution $p_\theta(\boldsymbol{\pi}|s)$ parameterized by θ given a problem instance s. Using chain rule, $p_\theta(\boldsymbol{\pi}|s)$ can be factorized as

$$p_\theta(\boldsymbol{\pi}|s) = \prod_{i=1}^{n} p(\pi_i|s, \boldsymbol{\pi}_{1:i-1}). \tag{3.2}$$

PN is a sequence-to-sequence model, which comprises two modules, namely encoder and decoder. The encoder reads the input sequence of a TSP s, and the decoder

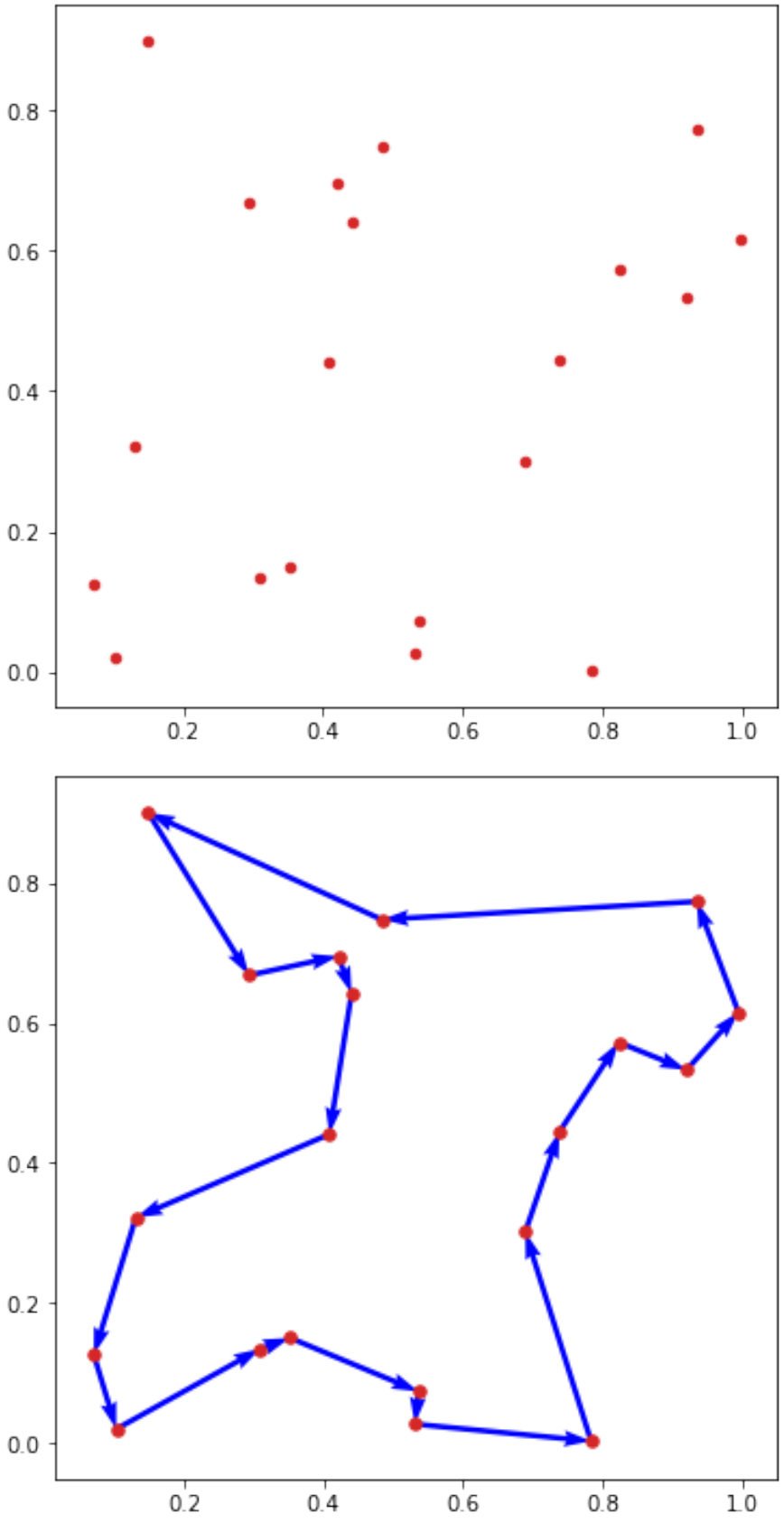
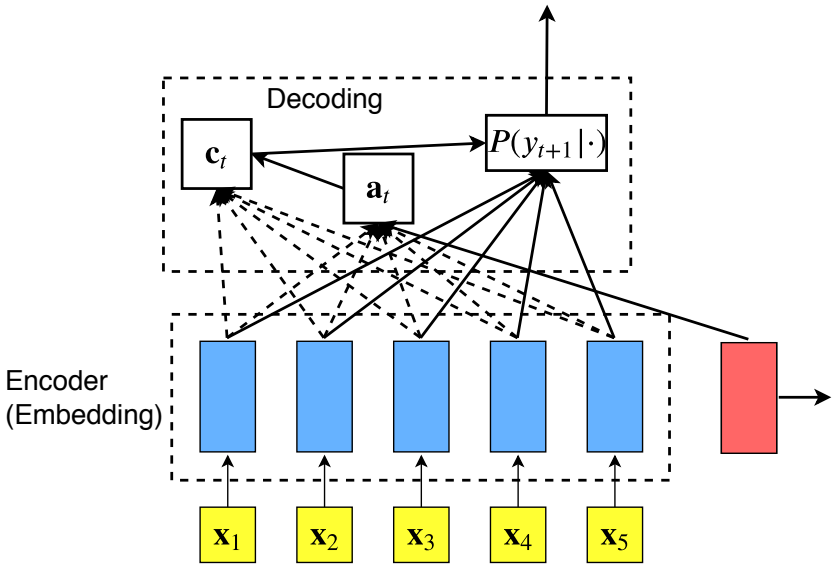
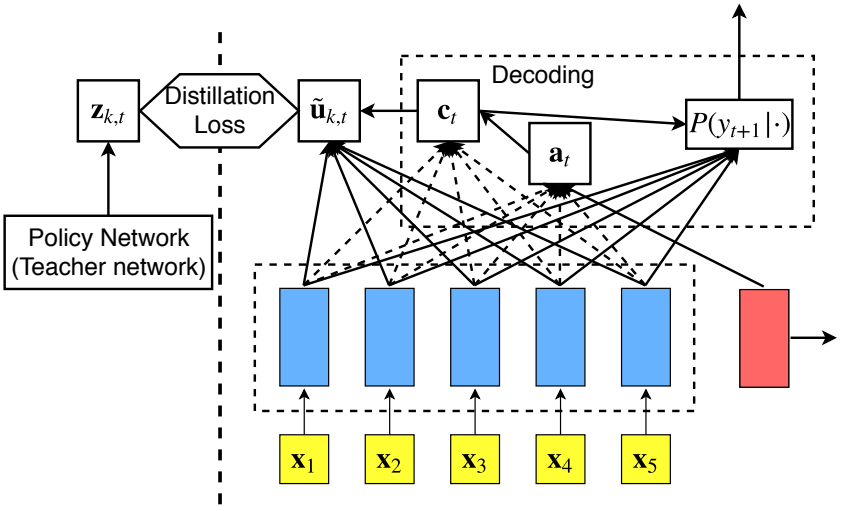


FIGURE 3.1: (**a**) A 2D TSP with 20 cities. (**b**) The optimal solution to the TSP.



(a) Pointer network.



(b) Pointer network with policy distillation.

FIGURE 3.2: (**a**) Pointer Network. The embedding layer maps the inputs to a high-dimensional embedding space; a RNN decoder stores the information of the decoded sequence; and attention is applied to produce a probability distribution over the next input; (**b**) Pointer network for efficient knowledge transfer (tPN). Different source models can be used to transfer knowledge to the target model through a task-specific attention layer.

uses a pointing mechanism to produce a conditional distribution over the next city to visit in the route. In [35], convolution layers (instead of recurrent neural network) are used for encoding, and decoder is a recurrent neural network. Specifically, the encoder produces embeddings of all input nodes, by applying 1-dimensional convolution layers, in which the in-width is the input length, and the number of in-channels is the number of cities in s. The decoder then produces the sequence $\boldsymbol{\pi}$ of input nodes, with one node at each time step. Say $\mathbf{e}_i$ is the embedded input of city $\mathbf{x}_i$. The decoder observes a mask to filter out the cities already visited, thus attention mechanism is applied to select the remaining city to be visited next. At decoding step t, a context-based attention mechanism with glimpse is utilized, extracting the relevant information from the inputs using a variable-length alignment vector $\mathbf{a}_t$. $\mathbf{a}_t$ specifies how much every input data point might be relevant at the current decoding step, and is computed as:

$$\mathbf{a}_t = \mathbf{a}_t(\mathbf{e}_i, \mathbf{h}_t) = \sigma(\mathbf{u}_t), \tag{3.3}$$

where $\mathbf{h}_t$ refers to the memory state of the RNN cell at step t, $\sigma(\cdot)$ denotes a softmax function, and ith element of $\mathbf{u}_t$ is computed as $u_t^i = \mathbf{v}_a^T \tanh(\mathbf{W}_a[\mathbf{e}_i; \mathbf{h}_t])$ ($[\cdot;\cdot]$ means concatenating two vectors). Therefore, the context vector at time step t is

$$\mathbf{c}_t = \sum_{i=1}^{n} a_t^i \mathbf{e}_i. \tag{3.4}$$

The probability for the final output is computed as follows, by using the context vector $\mathbf{c}_t$:

$$\tilde{u}_t^i = \mathbf{v}_c^T \tanh(\mathbf{W}_c[\mathbf{e}_i; \mathbf{c}_t]), i = 1, \cdots, n \tag{3.5}$$

$$P(\pi_{i+1}|s, \boldsymbol{\pi}_{1:i}) = \sigma(\tilde{\mathbf{u}}_t) = \sigma([\tilde{\mathbf{u}}_t^1, \cdots, \tilde{\mathbf{u}}_t^n]), \tag{3.6}$$

where $\mathbf{v}_a$, $\mathbf{W}_a$, $\mathbf{v}_c$, and $\mathbf{W}_c$ are all trainable parameters.

In [93], it is proposed to treat negative objective value (for minimization problem) as the reward signal, and use model-free reinforcement learning to optimize the Pointer Network parameterized by θ. For instance, given a TSP with size n described by a graph s, the training objective is to minimize the expected tour length, which could be defined

as:

$$J(\theta|s) = \mathbb{E}_{\pi \sim p_\theta(\cdot|s)} L(\pi|s), \tag{3.7}$$

where $p_\theta(\cdot|s)$ refers to joint probability predicted by the Pointer Network.

Having this training objective, policy gradient methods and stochastic gradient descent can be deployed to optimize θ. Using well-established REINFORCE algorithm [94], the gradient of (3.7) is calculated as:

$$\nabla_\theta J(\theta|s) = \mathbb{E}_{\pi \sim p_\theta(\cdot|s)} \Big[(L(\pi|s) - b(s)) \nabla_\theta \log p_\theta(\pi|s) \Big], \tag{3.8}$$

where $b(s)$ is the baseline function, which estimates the expected tour length.

During training stage, assuming within each batch, B problems $s_1, \cdots, s_B$ are sampled from a specific problem distribution. According to p_θ, a single tour π_i is sampled for problem s_i, i.e., $\pi_i \sim p_\theta(\pi_i|s_i)$. From (3.8), the gradient can be approximated with Monte Carlo sampling:

$$\nabla_\theta J(\theta|s) \approx \frac{1}{B} \sum_{i=1}^{B} (L(\pi_i|s_i) - b(s_i)) \nabla_\theta \log p_\theta(\pi_i|s_i). \tag{3.9}$$

Using a parametric baseline, for example neural network, to estimate the expected tour length $\mathbb{E}_{\pi \sim p_\theta(\cdot|s)} L(\pi|s)$ typically improves learning. Therefore, *actor-critic* algorithm [94] is applied, in which *actor* is the Pointer Network, and *critic* refers to the auxiliary network learning the expected reward given a TSP instance. Assuming the critic network is parameterized by θ_v, the objective is thus written as:

$$\mathcal{L}_v(\theta_v) = \frac{1}{B} \sum_{i=1}^{B} \| b_{\theta_v}(s_i) - \mathcal{L}(\pi_i|s_i) \|_2^2 \tag{3.10}$$

3.2 Pointer Network for Efficient Knowledge Transfer

In [35], a well-trained PN can directly predict high-quality solutions for unseen problems drawn from the training distribution. However, it has recently been reported that even a well-trained neural network model tends to have difficulty in generalizing out

of the training distribution; this is also shown in our experimental study. Bear in mind that we often have no prior knowledge of a particular target optimization problem or its distribution, and thus the capability of generalizing out of training distribution is one of the key factors to deploy these models into practical settings. To this end, we propose to reuse multiple well-trained networks as source models adaptively, to help train a target network which is capable of generalizing out of the training distribution.

Suppose we are already given K source models, each of which is capable of solving problems drawn from a specific problem distribution. One of the common strategies to teach a target network with a set of source models is referred to as knowledge distillation [8]. To elaborate, in the setting of TSP, logit vector at time t from kth source model is denoted as $\mathbf{z}_k, t \in \mathcal{R}^n, t = 1, \cdots, n$ (which can be converted to a probability distribution by a softmax function). To conduct knowledge distillation, the following Kullback-Leibler (KL) divergence with temperature τ is optimized:

$$\mathcal{L}_{\mathrm{KL}}(\theta; \tilde{\mathbf{U}}, \mathbf{Z}_k) = \sum_{t=1}^{n} \sum_{i=1}^{n} \sigma(\frac{\mathbf{z}_{k,t}^i}{\tau}) \ln \frac{\sigma(\frac{\mathbf{z}_{k,t}^i}{\tau})}{\sigma(\tilde{u}_t^i)}, \tag{3.11}$$

where $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, and $\tilde{\mathbf{U}} = \{\tilde{\mathbf{u}}_1, \cdots, \tilde{\mathbf{u}}_n\}$. To transfer more knowledge, the outputs from the source model are softened by passing the network output through a relaxed (higher temperature τ) softmax [95], as when the soft targets have higher entropy, they can provide more information.

Nevertheless, there is a possibility that this straightforward knowledge transfer strategy is confounded by contradictions in the knowledge transferred from different source models [96]. The reasoning behind this observation is perhaps that different source models may have discovered differing search behaviors under different problem distributions, such that the target network may face difficulties in finding a unified neural encoding of heuristics are suitable across problems drawn from different distributions.

In order to mitigate the direct contradiction among different source models, we introduce a novel neural network architecture based on Pointer Network, labeled as transfer Pointer Network (tPN), as shown in Figure 3.2b. In tPN, instead of learning the whole neural network directly via knowledge distillation, high-level representations are

first learned from different source models. To achieve this, a task-specific module for each source model is designed in this network. As such a module needs to handle variable-length logits from different source models, the same pointer-based attention mechanism, as previously discussed, is adopted.

In tPN, with the context vector $\mathbf{c}_t$ at step t, the corresponding logits $\mathbf{u}_{k,t}$ for the kth source model are calculated as follows:

$$\mathbf{u}_{k,t}^i = \mathbf{v}_k^T \tanh(\mathbf{W}_k[\mathbf{e}_i; \mathbf{c}_t]),\ i = 1, \cdots, n. \tag{3.12}$$

Given K source models, additional training variables, $\mathbf{v}_k, \mathbf{W}_k, k = 1, \cdots, K$, have to be learned. Note that several layers of such attention mechanism could be further stacked in the target model. Nevertheless, in the proposed network structure, one attention layer is empirically found to be sufficient to enhance the generalization performance significantly.

Across different batches, a specific source model, say the kth model, is selected to teach the target model via its corresponding task-specific attention layer, through optimizing the following KL divergence:

$$\mathcal{L}_{\text{KL}}(\theta; \tilde{\mathbf{U}}_k, \mathbf{Z}_k) = \sum_{t=1}^{n} \sum_{i=1}^{n} \sigma(\frac{\mathbf{z}_{k,t}^i}{\tau}) \ln \frac{\sigma(\frac{\mathbf{z}_{k,t}^i}{\tau})}{\sigma(\tilde{u}_{k,t}^i)}, \tag{3.13}$$

where $\tilde{\mathbf{U}}_k = \{\tilde{\mathbf{u}}_{k,1}, \cdots, \tilde{\mathbf{u}}_{k,n}\}$.

After knowledge transfer stage from K different source models, in which the training procedure generally converges in few batches, the K task-specific attention layers are then discarded. Since all the lower layers, especially embedding layers and memory state in the decoding RNN, are shared during knowledge distillation stage, generalizable representations can be learned. After that, the target model is fine-tuned with randomly initialized $\mathbf{v}_c, \mathbf{W}_c$ on TSPs with different sizes for a few epochs using standard reinforcement learning techniques.

To summarize, the whole training procedure, combined with actor-critic for fine-tuning, is recapped in Algorithm 2.

Algorithm 2: Training procedure for tPN

Input: source probabilistic models parameterized by $\theta_1, \cdots, \theta_K$, number of epochs E_p for knowledge transfer, batch size B, number of epochs E for standard training
Init θ for actor network, Φ for critic network
for $epoch = 1, \cdots, E_p$ **do**
  for $batch = 1, \cdots, T$ **do**
    $k \leftarrow \text{mod}(batch, K)$ {Select source model.}
    $s_i \leftarrow \text{RANDOMINSTANCE}(k)\ \forall\ i \in \{1, \cdots, B\}$
    $\boldsymbol{\pi}_i, \mathbf{Z}_i \leftarrow \text{SAMPLEROLLOUT}_{\theta_k}()\ \forall\ i \in \{1, \cdots, B\}$ {Rollout according to the source model.}
    $\tilde{\mathbf{U}}_i \leftarrow \text{GETLOGITS}_{\theta}(\boldsymbol{\pi}_i)\ \forall\ i \in \{1, \cdots, B\}$ {Calculate logits vector from the sampled route $\boldsymbol{\pi}_i$ on the target model.}
    $\mathcal{L}_{\text{KL}} \leftarrow \sum_{i=1}^{B} \mathcal{L}_{\text{KL}}(\theta \mathbf{Z}_i, \tilde{\mathbf{U}}_i)$
    $\theta \leftarrow \text{ADAM}(\theta, \mathcal{L}_{\text{KL}})$
  end for
end for
for $epoch = 1, \cdots, E$ **do**
  for $batch = 1, \cdots, T$ **do**
    $s_i \leftarrow \text{RANDOMINSTANCE}()\ \forall\ i \in \{1, \cdots, B\}$
    $\boldsymbol{\pi}_i \leftarrow \text{SAMPLEROLLOUT}()\ \forall\ i \in \{1, \cdots, B\}$
    $b_i \leftarrow V_{\Phi}(s_i)\ \forall\ i \in \{1, \cdots, B\}$
    $\nabla_{\theta}\mathcal{L} \leftarrow \frac{1}{B}\sum_{i=1}^{B}\big(L(\boldsymbol{\pi}_i) - b_i\big)\nabla_{\theta}\log p_{\theta}(\boldsymbol{\pi}_i|s_i)$
    $\nabla_{\Phi}V \leftarrow \frac{1}{B}\sum_{i=1}^{B}\nabla_{\Phi}\left\|b_i - \mathcal{L}(\pi_i|s_i)\right\|_2^2$
    $\theta \leftarrow \text{ADAM}(\theta, \mathcal{L})$
    $\Phi \leftarrow \text{ADAM}(\Phi, V)$
  end for
end for

3.3 Experiments

3.3.1 Traveling Salesman Problem

During each epoch in the training stage, 1,280,000 TSP instances are randomly generated, in which the node locations are chosen uniformly from the unit square $[0, 1] \times [0, 1]$. The batch size B is set to 512. The decoding process starts from a random TSP node. A masking scheme is deployed to prohibit visiting the same node twice. During training, problems with 20, 50, and 100 nodes are generated sequentially across batches, denoted as TSP20, TSP50, and TSP100. All the experiments are conducted on Tesla V100. In the test phase, greedy decoding is deployed, in which at every decoding step, the node with highest probability is chosen to be visited next.

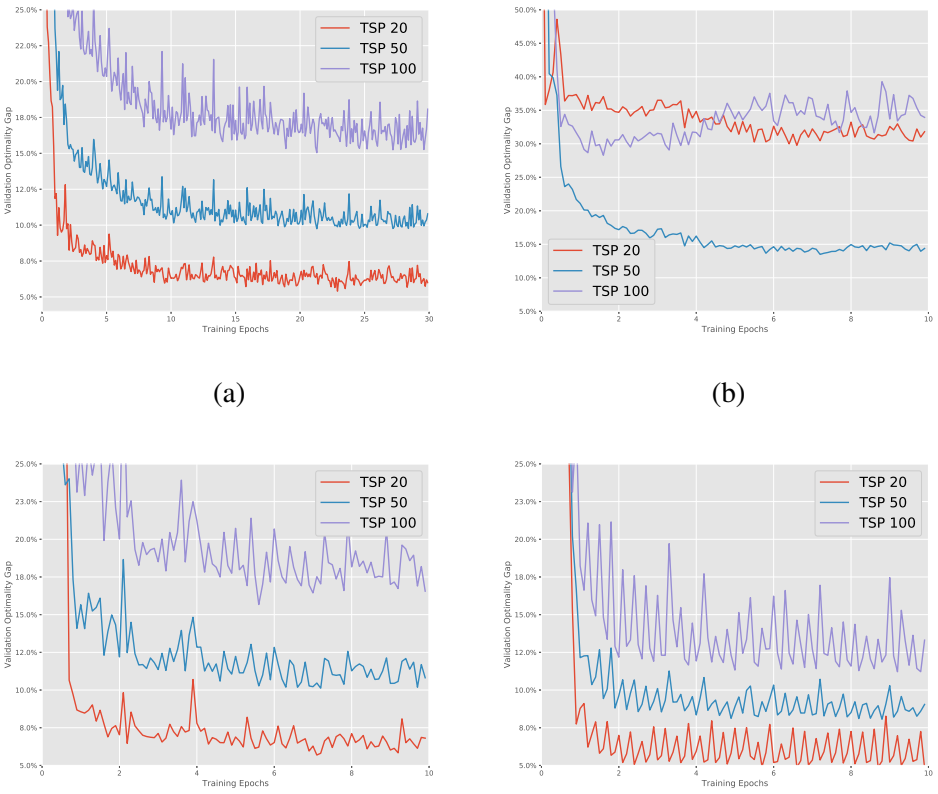


FIGURE 3.3: (a) Hold-out validation set optimality gap as a function of epochs for 'PN + Scratch' (b) Hold-out validation set optimality gap as a function of epochs for 'PN + only KD' with $E_1 = 10$; (c) Hold-out validation set optimality gap as a function of epochs for 'PN + KD'; (d) Hold-out validation set optimality gap as a function of epochs for the proposed tPN.

TABLE 3.1: Averaged tour length for TSP using greedy sampling strategy with variable problem sizes. Superior performances are highlighted in bold. Strongest generalization performance of the proposed tPN can be observed.

Method	TSP20	TSP50	TSP100	TSP150
optimal	3.83	5.69	7.76	9.35
PN, TSP20	**4.02**(4.86%)	6.69(17.51%)	10.54(35.82%)	14.37(53.62%)
PN, TSP50	4.16(8.70%)	6.31(10.79%)	9.05(16.62%)	11.37(21.51%)
PN, TSP100	5.00(30.43%)	6.66(17.05%)	9.00(15.95%)	11.07(18.40%)
PN + Scratch	4.08(6.55%)	6.30(10.70%)	9.13(17.62%)	11.57(23.74%)
PN + only KD	5.05(31.81%)	6.51(14.38%)	10.39(33.89%)	14.21(51.88%)
PN + KD	4.09(6.81%)	6.31(10.82%)	9.05(16.56%)	11.40(21.86%)
tPN	4.04(5.38%)	**6.16**(8.26%)	**8.65**(11.43%)	**10.68**(14.20%)

The proposed method is compared with various baseline methods, including the source models[1]. The specifications for different methods are listed as follows:

- 'PN, TSP20': a randomly initialized PN which is only trained on TSP20 for 10 epochs. Similarly for 'PN, TSP50' and 'PN, TSP100'. These 3 models also serve as source models for knowledge distillation;
- 'PN + Scratch': a randomly initialized PN which is trained on TSP20, TSP50, and TSP100 across different batches. This network is trained for 30 epochs for fair of comparison;
- 'PN + only KD': a randomly initialized PN is first trained using knowledge distillation for $E_1 = 10$ epochs with no fine-tuning using actor-critic;
- 'PN + KD': a randomly initialized PN is first trained using knowledge distillation for $E_1 = 1$ epochs, and fine-tuned using actor-critic for $E_2 = 9$ epochs ($E_1 + E2 = 10$);
- 'tPN': the proposed tPN is first trained using knowledge distillation for $E_1 = 1$ epochs, and fine-tuned using actor-critic for $E_2 = 9$ epochs;

For knowledge distillation, τ is set to 10. The validation performances as a function of epochs for the target model on the hold-out validation set are shown in Figure 3.3. From the figures, stronger overall generalization performance of the proposed model is achieved during training stage on the hold-out validation set. Surprisingly, from Figure 3.3b, PN with only knowledge distillation performs worst, and the contradiction between different source models is verified, thus highlighting the necessity for knowledge transfer through high-level representation learning.

The above mentioned models are tested on TSP problems with various sizes, and the corresponding results shown in Table 3.1 are averaged over 1,000 TSP instances[2]. Clearly, the proposed tPN outperforms other methods over a wide range of optimization

[1]The model are trained using the code provided in https://github.com/mveres01/pytorch-drl4vrp. Performance gap can be observed comparing to the results reported in [35].

[2]The optimal value in the table is calculated using Concorde http://www.math.uwaterloo.ca/tsp/concorde/index.html.

TABLE 3.2: Averaged tour length for dynamic TSP using greedy sampling strategy with variable problem sizes. Superior performances are highlighted in bold.

Method	TSP20	TSP50	TSP100	TSP150
PN, TSP20	4.82	8.23	12.72	16.95
PN, TSP50	4.93	7.67	11.11	13.86
PN, TSP100	5.60	8.28	11.44	14.09
PN + Scratch	4.76	7.70	11.24	14.11
PN + KD	4.78	7.58	10.91	13.61
tPN	**4.60**	**7.37**	**10.48**	**12.87**

problems. To our surprise, tPN can surpass 'PN, TSP50' on TSP50, and surpass 'PN, TSP100' on TSP100. Further, tPN shows strong generalization ability, as it outperforms other methods over a large margin. As 'PN + only KD' performs worst, this baseline will be ignored hereinafter.

To further test the generalization ability out of training distribution, part of nodes to be visited are only visible to the network after visiting certain number of nodes. To elaborate, for a TSP with n nodes, $\lfloor n * 2/3 \rfloor$ nodes are available beforehand; at decoding step $\lfloor n/3 \rfloor$, the rest $n - \lfloor n * 2/3 \rfloor$ nodes appear. Such dynamic TSPs are tested with various sizes, including 20, 50, 100, 150, on the model obtained from the previous experimental study. The experimental results are shown in Table 3.2. Superior performances on all the test cases for the proposed model highlight the importance of the learning of generalizable high-level representations. Comparing tPN with 'PN + KD', the efficacy of the proposed neural network structure for transfer learning is manifested.

3.3.2 0/1 Knapsack Problems (KP)

KP is a classical NP-hard problem in discrete (combinatorial) optimization, popularly studied in the operations research literature. The objective of the problem is to, given a knapsack of capacity C, and a set of d items, each with a weight w_i and a value q_i, find a selection of items such that the total value is maximized without violating the capacity constraint. The mathematical formulation of the KP is defined as follows:

$$\max \sum_{i=1}^{d} q_i x_i$$

TABLE 3.3: Results for KP using greedy sampling strategy with variable problem sizes(KP d/C is a KP problem with capacity C and d items). Superior performances are highlighted in bold. Strongest generalization performance of the proposed tPN can be observed.

Method	KP50/12.5	KP100/25	KP200/25	KP10/2	KP75/12.5	KP150/25	KP250/40	KP300/45
PN, KP20	**19.85**	**40.20**	55.81	**3.20**	24.29	49.00	80.22	93.02
PN, KP50	**19.85**	**40.20**	55.67	**3.20**	24.26	48.95	80.13	92.88
PN, KP100	19.55	39.56	**57.17**	3.19	24.47	49.15	81.04	94.33
PN + Scratch	19.77	40.00	57.07	**3.20**	**24.55**	**49.53**	81.24	94.44
PN + KD	19.75	39.99	57.03	3.19	**24.55**	49.49	81.19	94.45
tPN	19.77	40.00	57.12	**3.20**	**24.55**	**49.53**	**81.25**	**94.47**

$$\text{s.t.} \sum_{i=1}^{d} w_i x_i \leq C \text{ and } x_i \in \{0, 1\}, \tag{3.14}$$

Here, $x_i = 1$ indicates that the i^{th} item is selected, while $x_i = 0$ indicates that the i^{th} item is not selected.

Following the work in [93], three types of KPs are generated, KP50, KP100, and KP200, with items' weights and values sampled uniformly from [0,1]. And the capacity of the KP is set to 12.5 for KP50, and 25 for KP100 and KP200, denoted as KP50/12.5, KP100/25, and KP200/25, respectively. The averaged performances of various methods on 1,000 newly generated problem instances are presented in Table 3.3 using greedy decoding. As KP might be slightly less challenging as TSP, all the models perform comparable on KP50/12.5, KP100/25, and KP200/25. However, when the models are tested on problems drawn out of training distributions, tPN again proves its strong generalization ability.

3.4 Conclusion

In this chapter, learning a single generalizable model from multiple related source models is studied, in an attempt to enhance the applicability of neural networks for NP-hard combinatorial optimization in practical settings. Straightforward knowledge transfer into a target model can be less effective as the knowledge discovered by different source models may counteract each other, thus confounding the learning process for the target model. On the other hand, our proposed tPN is shown to have strong generalization

ability across optimization problems drawn from various unseen distributions. Experimental results have showcased that tPN is able to predict high-quality solutions for traveling salesman problems as well as knapsack problems over a wide range of instance sizes, comparing favorably against the source models as well as various baseline methods.

Chapter 4

Curbing Negative Influences Online for Seamless Transfer Optimization[1]

In the global black-box optimization literature, efforts have seldom been made to automate the re-use of knowledge acquired from past problem-solving experiences. This limitation is primarily due to the lack of problem-specific data available prior to the onset of search, which makes it difficult to ascertain (offline) the relationships across problems. Many of success stories in the literature of knowledge reuse highly depend on the prior knowledge. The transfer optimization strategy proposed in the previous chapter also assumes that the source probabilistic models have to be related and beneficial to the learning of the target model. All the source models and target model are essentially solving the same type of optimization tasks, due to the fixed neural network structure. However, in real-world settings, where various types of problem-solving experiences are mixed together, negative transfer can be a real threat if we naively transfer knowledge from all the source probabilistic models. This fact motivates increasing research attention towards online knowledge transfer for transfer optimization.

The present work in this chapter draws motivation from the remarkable ability of humans to extract useful building-blocks of knowledge from past experiences and spontaneously adaptively re-use them for new and more challenging tasks. It is contended that successfully replicating such capabilities in computational solvers, particularly global

[1]Partial resutls of the presented work have been published in [15].

black-box optimizers, can lead to significant performance enhancements over the current state-of-the-art. Given the rapid advancements in modern computing platforms such as the cloud and the IoT, which give rise to large-scale data storage and seamless communication facilities, the practical viability of such a paradigm (from a hardware perspective) is little in doubt. Thus, we focus on addressing the shortcomings that continue to exist in terms of developing suitable algorithms. In particular, this work focuses on the use of population-based *evolutionary algorithms* (EAs) as they not only provide significant flexibility in dealing with a wide range of discrete and continuous optimization problems, but are also amenable to be hybridized with various learning strategies. In the proposed approach, we capture the population distribution of *elite* solutions from some source optimization task in the form of a probabilistic model, that is then stored for future usage. These probabilistic knowledge building-blocks serve to bias the search on a *related* target task towards solutions that have been shown to be promising. Importantly, for knowledge transfer to occur in this manner, the probabilistic models must be defined in an all-encompassing *universal search space*, which creates a *common (shared) platform* for the exchange of ideas to take place [48]. The synthesis of diverse knowledge building-blocks is then realized by sampling candidate solutions from a *mixture of source + target probabilistic models*, with the mixture coefficients calculated in a manner that provides theoretical performance guarantees (see Section 4.3 for details).

At this juncture, it is worthwhile to mention that the techniques proposed in this work can be implemented within any EA of ones preference, as a way of endowing it with online knowledge transfer capabilities. Attention of the reader is primarily drawn towards the demonstrations of the explicit capture of source-target similarities for a generic transfer evolutionary optimizer, that, in principle, can be applied to any optimization problem. As an aside, we deem the proposed framework to fall within the realm of *memetic computing* [97], with the *memes* (popularly defined as *computationally encoded units of knowledge for improved problem-solving* [98]) herein taking the form of probabilistic models of elite solution distributions. In this regard, as the proposal is based on the transfer of learned models across problems, it is hereafter labeled as *adaptive model-based transfer* (AMT).

To summarize, the main contributions of this work are:

- A novel model-based transfer EA that is capable of online learning and exploitation of similarities across black-box optimization problems, in a manner that minimizes the threat of negative transfer.
- Theoretical analysis of the proposed approach, demonstrating that the knowledge-enhancement scheme guarantees to facilitate global convergence of the EA.
- Rigorous experimental verification of the algorithm on a diverse test suite. In particular, the neural network model learned in Chapter 3 also serves as one of the source probabilistic models in the case study on TSP, demonstrating the necessity of adaptive online knowledge transfer and the efficacy of the proposed AMT framework.

The remainder of the chapter is organized as follows. In Section 4.1, the key ingredients that form the crux of the AMT framework are introduced. In Section 4.2, we present an instantiation of a transfer evolutionary optimization algorithm incorporating the proposed online similarity learning strategy. We label this algorithm as AMT-*enabled* EA (or simply AMTEA). The theoretical foundations and justifications of the framework are then analyzed in Section 4.3. In Section 4.4, we present numerical results demonstrating the efficacy of the AMTEA across a range of problems spanning discrete, continuous, single-, and multi-objective optimization.

4.1 Basics of Adaptive Model-based Transfer

We consider a case where there are $K-1$ previously tackled source optimization tasks, labeled as $\mathcal{T}_1, ..., \mathcal{T}_{K-1}$, and a target task $\mathcal{T}_K$ of current interest. Each task could either have a single or multiple objectives. In the former case, we denote the objective function of the k^{th} task as f_k. In this study, we consider the building-blocks of knowledge extracted from source tasks to take the form of probabilistic models of optimized search distributions, that are subsequently used to bias the search on the target. Specifically, a model φ_k drawn from a maximization task $\mathcal{T}_k$ satisfies:

$$\int f_k(\mathbf{x})\varphi_k(\mathbf{x})d\mathbf{x} \geq f_k^* - \epsilon_k, \tag{4.1}$$

where $(^*)$ represents the global optimum, and $\epsilon_k(>0)$ is a small convergence tolerance threshold.

A favorable aspect of probabilistic models is their relatively small memory footprint. As an example, consider the case of a source task for which a large number (say N) of solutions have been evolved, each consisting of B binary bits. Naively storing the raw solution data consumes NB bits of memory. In contrast, a factored Bernoulli distribution can represent higher-order knowledge about the underlying distribution of the same population of solutions while consuming only $\mathcal{O}(B\log_2 N)$ bits of memory [99].

For the purpose of facilitating seamless transfer of knowledge building-blocks across problems, two ingredients form the crux of the proposed AMT framework, namely, (1) a *universal search space*, and (2) mixture modeling through *stacked density estimation* [100]. These are discussed next.

4.1.1 The Universal Search Space

Simply stated, a universal search space $\mathcal{X}$ serves as a common (unified) platform bringing together the individual search spaces of distinct optimization tasks. The unification is the key element that enables probabilistic models drawn from different source tasks to be directly brought to bear on the target task $\mathcal{T}_K$. To elaborate, consider the following optimization problem reformulation of $\mathcal{T}_K$ in terms of the search distribution,

$$\mathcal{T}_K : \max_{p(\mathbf{x})} \int_{\mathcal{X}} f_K(\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{4.2}$$

In the above, by approximating the latent distribution $p(\mathbf{x})$ as

$$p(\mathbf{x}) \approx \sum_{k=1}^{K} \alpha_k \varphi_k(\mathbf{x}), \tag{4.3}$$

where $\sum_{k=1}^{K} \alpha_k = 1$ and $\alpha_k \geq 0$ for $k = 1, \cdots, K$, the source models are activated to influence the target search. Note that the mixture weights (α_k's) are tunable, enabling us to adapt the extent of influence.

With this in mind, the universal search space $\mathcal{X}$ is described so as to *encode* solutions to *all* (source + target) optimization problems, such that all probabilistic models $\varphi_1, \cdots, \varphi_K$ can be built in this space. To avoid abuse of notations, $f_1, \cdots, f_K$ are considered to be defined in the universal space.

While dealing only with continuous optimization problems, a viable unification procedure is to linearly scale each variable to the common range of $[0, 1]$. Further, in [16, 48, 49], it was shown that by using an associated *random-key encoding scheme* [101], it becomes possible to unify discrete and continuous optimization problems as well, with provisions for handling search spaces of differing dimensionality. As an illustration, consider K distinct optimization problems $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_K$ with search space dimensionality $d_1, d_2, \ldots, d_K$, respectively. In such a scenario, a unified space $\mathcal{X}$ of dimensionality $d_{\text{unified}} = \max\{d_1, d_2, \ldots, d_K\}$ is defined, so that candidate solutions with respect to all optimization problems can be encoded in $\mathcal{X}$. Thereafter, while addressing the k^{th} optimization problem, a subset of d_k variables are extracted from a candidate solution vector in $\mathcal{X}$, and *decoded* (inverse of the encoding step) into a task-specific solution representation. Importantly, the cost involved in random-key encoding and decoding is practically negligible, which implies that there is little computational overhead compared to the core optimization steps.

In the context of AMT, once a probabilistic model capturing the search distribution corresponding to any source task is built (in the universal space), it can be transferred to a target optimization task of interest. Note that the possible difference in search space dimensionality of source and target optimization tasks can be addressed via a simple heuristic strategy. *If the dimensionality of the source optimization problem is greater than that of the target, the transferred probabilistic model is simply restricted to the active subset of variables by marginalizing over the distribution of all inactive variables. On the other hand, if the dimensionality of the source optimization problem is smaller, extra variables are padded to the source probabilistic model - with each new variable assigned an independent uniform distribution.*

4.1.2 Background on Mixture Modeling for Source-Target Similarity Capture

Given the premise of Eqs.(4.2) and (4.3), herein, we present a brief overview of *finite mixture modeling* for probability density estimation. The goal of finite mixture modeling can be described as the linear combination of probabilistic models for estimating an unobservable latent probability density function based on observed data. Within the context of optimization problem-solving, consider $D_t = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ to represent a dataset of N solutions in a universal search space $\mathcal{X}$ at the t^{th} generation of an EA tackling target task $\mathcal{T}_K$. Notice that since $D_0 = \emptyset$, offline source target similarity measurement is precluded. As introduced in Eq.(4.2), $p(\mathbf{x}|t)$ is the *true* latent probability density function describing the target population dataset D_t.

With this, the finite mixture model is defined as the following stacking (aggregation) of probabilistic models:

$$q(\mathbf{x}|t) = \sum_{k=1}^{K} \alpha_k \varphi_k(\mathbf{x}), \tag{4.4}$$

where $q(\mathbf{x}|t)$ is the desired *approximation* of the latent probability density function $p(\mathbf{x}|t)$, and components φ_k, $k = 1, \ldots, K$, are individual probabilistic models. In the present case, these K models include those that are drawn from the $K-1$ source optimization problems, as well as a *preliminary* model φ_K built for the target task $\mathcal{T}_K$. The coefficients in the mixture model (α_k's) are seen as capturing the *similarity* between the k^{th} source and the target. If the probabilistic model φ_k has little relevance to the target, then the corresponding *transfer* coefficient α_k will take a value close to zero once the mixture is optimized (thereby reducing the influence of the corresponding source). On the other hand, if φ_k is useful for improving the approximation of the latent probability density function $p(\mathbf{x}|t)$, then α_k will take a relatively high value (closer to one). To this end, the objective of the mixture learning algorithm (detailed in Section 4.1.3) is to deduce α_k's such that the probability of observing *out-of-sample* target data is maximized. Given an out-of-sample (test) dataset D_{test}, the mathematical program is accordingly formulated as the maximization of the following log-likelihood function:

$$\log L = \sum_{\mathbf{x}_i \in D_{test}} \log q(\mathbf{x}_i|t). \tag{4.5}$$

4.1.3 Adaptive Model-based Transfer with Online Source-Target Similarity Learning

We approach the formulation in Eq.(4.2) by successively estimating the distribution $p(\mathbf{x}|t)$ of a population evolving towards the optimum of $\mathcal{T}_K$. In this regard, a distinguishing feature of the proposed stacked density estimation procedure, as opposed to prior work [100], is that $K-1$ *pre-trained* probabilistic models originate from source tasks that are distinct from the ongoing target task of interest. Further, a source model φ_k may itself be a finite mixture model. Nevertheless, while solving $\mathcal{T}_K$, φ_k is always viewed as a single component encapsulating the knowledge acquired from the past k^{th} optimization experience.

With the learned mixture of probabilistic models capturing source-target similarities, the so-called adaptive transfer of knowledge is realized by iteratively sampling target candidate solutions from the mixture distribution. The theoretical rationale behind doing so, with regard to guiding the evolutionary search, shall be substantiated in Section 4.3.

Next, we enumerate the steps followed for learning the optimal finite mixture model $q(\mathbf{x}|t)$ in practice:

- **Step 1** Randomly partition the target population D_t into v-folds according to the standard cross-validation procedure. For each fold, a target probabilistic model is learned from the *training part* of the partition of D_t. Thereafter, the likelihood of each data point in the *test partition* of D_t (which constitutes D_{test}) is evaluated corresponding to the $K-1$ pre-trained source models and the learned target probabilistic model.

- **Step 2** By repeating Step 1 for each of the v folds, construct an $N \times K$ matrix comprising K density estimates for each of the N data points in D_t. The $(i,k)^{th}$

entry of the matrix is $\varphi_k(\mathbf{x}_i)$, representing the out-of-sample likelihood of the k^{th} model on the i^{th} data point in D_t.

- **Step 3** Using the matrix constructed from pre-trained models in Step 2, the α_k's of the mixture model are learned by maximizing the following equivalent form of Eq.(4.5):

$$\log L = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \alpha_k \varphi_k(\mathbf{x}_i). \tag{4.6}$$

  This can be (easily) solved by applying the classical expectation-maximization (EM) algorithm [102]. For the sake of brevity, details of the algorithm are not reproduced herein. Readers are referred to [103] for an intuitive description.

- **Step 4** To complete the learning process, consider the target probabilistic model trained on the entire training dataset D_t (without partitioning) to give φ_K. The final mixture model $q(\mathbf{x}|t)$ is thus the linear combination of the stored $K-1$ source probabilistic models $\varphi_1, \cdots, \varphi_{K-1}$ and the fully trained target model φ_K, with the combination given by the learned transfer coefficients (α_k's).

With regard to ascertaining the computational viability of the proposed AMT procedure, it is observed that the EM algorithm typically converges fast, i.e., within the first few iterations. In the case a *leave-one-out* cross-validation procedure is used in Step 1, the complexity of repeatedly building for example a factored Bernoulli distribution model is only $\mathcal{O}(N * d_K)$, as the model for each fold can be directly retrieved from the target model φ_K trained on the whole dataset. In other words, it is reasonable to incorporate the entire learning algorithm as a nested module within any external EA, at little computational overhead. Details of an instantiation of a generic transfer evolutionary optimization algorithm containing the aforesaid ideas are presented in the next section.

4.2 Framework for an AMT-enabled Evolutionary Algorithm: AMTEA

The main motivation behind incorporating the notion of transfer in optimization is to effectively exploit the potentially rich pool of knowledge that may be found in previous problem-solving experiences. To this end, a transfer interval (denoted by Δ generations) is first introduced in the EA, which determines the frequency at which the adaptive model-based transfer procedure (of Section 4.1.3) is launched. Notice that since the AMT procedure is repeated periodically during the course of the evolutionary search, latent synergies, those that are less apparent at the start of the search, may in fact be gradually revealed. Overall, the frequency of transfer Δ controls the rate at which the target optimization search is subjected to the influence of the source tasks. In turn, it serves to manage the computational resources allocated to the stacked density estimation procedure; even though the computational cost associated with the learning module is small when using simplistic probabilistic models φ. To summarize, the AMT framework can be implemented as a nested subroutine within any canonical or state-of-the-art EA. A conceptual illustration of the basic structure of an AMTEA is shown in Fig. 4.1, and a general pseudocode enumerating the steps of the proposal is presented in Algorithm 3.

When introduced with a new target optimization problem $\mathcal{T}_K$, with dimensionality d_K, the initial population of the AMTEA algorithm is randomly generated. The iterative fitness-based parent selection and offspring creation process is then commenced, incrementing the generation count (t) by one at each iteration. While $\mod(t, \Delta) \neq 0$, the AMTEA progresses in exactly the same manner as a standard EA, applying genetic operators such as crossover and/or mutation on the parent population P^s to produce the next generation of offspring individuals P^c.

Whenever $\mod(t, \Delta) = 0$, the AMT procedure is launched. As a first step, the parent individuals in P^s are encoded in the universal search space $\mathcal{X}$, forming the target population dataset D_t. For continuous optimization problems, the unification can be efficiently achieved via linear scaling of variables to a common range [16, 48]. Various

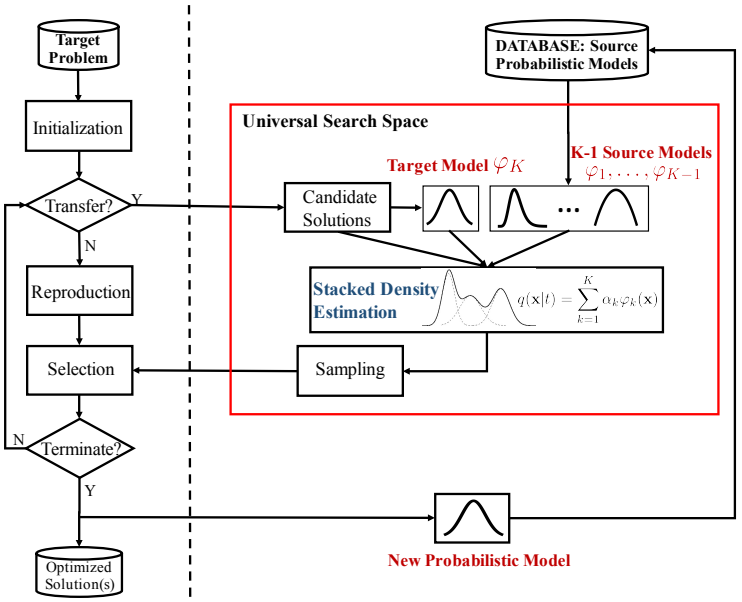


FIGURE 4.1: A conceptual illustration of the proposed AMTEA.

Algorithm 3: Pseudocode of AMTEA

Input: Pre-trained source probabilistic models; target optimization problem; transfer interval Δ

Output: Optimized solution(s) to the target optimization problem

Set $t = 1$
Randomly generate N initial solutions: $P(t)$
Evaluate target objective values of individuals in $P(t)$
Marginalize or pad all pre-trained source models so as to match the dimensionality of the target problem
while stopping condition not satisfied **do**
  Sample parent population $P^s(t)$ from $P(t)$
  if $\mod (t, \Delta) == 0$ **then**
    Encode $P^s(t)$ in the universal search space $\mathcal{X}$, to form the target dataset D_t
    Learn the mixture model $q(\mathbf{x}|t)$ describing D_t: following Steps 1 to 4 in Section 4.1.3
    Sample $q(\mathbf{x}|t)$, and decode the generated samples to form the offspring population $P^c(t)$
  else
    Generate offspring population $P^c(t)$ by genetic operators such as crossover and/or mutation
  end if
  Evaluate individuals in $P^c(t)$
  Select next generation $P(t+1)$ from $P(t) \cup P^c(t)$
  Set $t = t + 1$
end while

ways of encompassing combinatorial problems within a continuized unification scheme can also be found in [49]. Thereafter, previous optimization experiences, represented in the form of source probabilistic models, are referenced from the stored database. In order to match the search space dimensionality d_K of the target problem, the models may be adjusted by either marginalizing out all non-active variables from the source, or by padding the models with additional variables (that are assumed to follow a uniform distribution) to make up for any deficit. Next, stacked density estimation (refer Steps 1 to 4 in Section 4.1.3) is performed for optimal blending of source and target probabilistic models and effective capturing of source-target similarities online. As has been mentioned earlier, a large transfer coefficient learned for the mixture model indicates that the corresponding source model is highly correlated (and therefore relevant) to the target, while a small transfer coefficient (close to zero) reflects low similarity between the source and target. The ability to automatically attenuate the effects of such an irrelevant source model is the hallmark of the present study, as it automatically curbs the threat of negative transfer. Finally, the learned mixture model $q(\mathbf{x}|t)$ is sampled to generate the subsequent offspring population in the universal search space. The sampled offspring are decoded, yielding P^c in a task-specific solution representation. The objective function values of the offspring can then be evaluated.

The iterative parent selection and offspring creation process, interweaving evolutionary search and AMT, continues until pre-specified stopping criteria for the AMTEA are met. It is worth mentioning that the final probabilistic model built for the target problem can be incorporated back into the database of optimization experiences (as shown in Fig. 4.1), which makes it available as a new building-block of knowledge for future problem-solving exercises.

4.3 Analyzing the Theoretical Foundations of Adaptive Model-based Transfer

The asymptotic global convergence of a population-based stochastic optimization algorithm can be stated as:

$$\lim_{t\to\infty} \int_{\mathcal{X}} f_K(\mathbf{x}) p(\mathbf{x}|t) d\mathbf{x} = f_K^*, \tag{4.7}$$

where f_K is the objective function of the target optimization problem defined in the universal search space $\mathcal{X}$, f_K^* is its globally optimum value, and $p(\mathbf{x}|t)$ is the latent probability density function of the population at generation t.

In this section, we highlight that the proposed AMT framework facilitates global convergence characteristics. For simplicity of analysis, it is assumed that the population size employed in the AMTEA is large, i.e., $N \to \infty$, and f_K is continuous. While such an assumption may not hold in practice, it is considered reasonable for demonstrating the theoretical foundations of the proposal, which are found to be borne out by the experimental studies. In fact, similar assumptions are adopted in the theoretical analyses of [104], which serves as an important stepping-stone for the AMT procedure. Specifically, the main result of interest is:

Theorem 4.1 (Zhang and Muhlenbein [104])**.** *In probabilistic-modeling based evolutionary algorithms, where $p(\mathbf{x}|t=0)$ is positive and continuous in $\mathcal{X}$, asymptotic global convergence is guaranteed for continuous objective functions if $p^s(\mathbf{x}|t) = p^c(\mathbf{x}|t)$; where $p^s(\mathbf{x}|t)$ and $p^c(\mathbf{x}|t)$ are the underlying distributions of P^s and P^c, respectively, at any generation t.*

In practice, there invariably exists a *gap* between $p^c(\mathbf{x}|t)$ and the *true* latent probability density function $p^s(\mathbf{x}|t)$ of the parent population. As a result, significant efforts have been made over the years for developing increasingly sophisticated methods for improved modeling of probability density functions. In this regard, *Theorem 4.1 suggests that the role of AMT in facilitating global convergence characteristics can be established by simply showing that the proposed mixture of all available (source + target) probabilistic models guarantees superior approximations of population distributions.*

Lemma 4.2. *Maximizing the log-likelihood function in Eq.(4.6) is equivalent to minimizing the gap (measured by the Kullback-Leibler divergence) between the mixture model $q(\mathbf{x}|t)$ and the population's true latent probability density function $p(\mathbf{x}|t)$, given $N \to \infty$.*

Proof. With $q(\mathbf{x}|t)$ as the finite mixture approximation of $p(\mathbf{x}|t)$, Eq.(4.5) can be rewritten as:

$$\lim_{N\to\infty} \frac{1}{N} \cdot \log L = \lim_{N\to\infty} \frac{\sum_{i=1}^{N} \log q(\mathbf{x}_i|t)}{N}. \tag{4.8}$$

Since the N population samples are drawn from the true latent probability density function $p(\mathbf{x}|t)$, the Glivenko-Cantelli theorem [105] indicates that the empirical probability density function induced by $N(\to \infty)$ converges to $p(\mathbf{x}|t)$. Thus:

$$\lim_{N\to\infty} \frac{1}{N} \cdot \log L = \int_{\mathcal{X}} p(\mathbf{x}|t) \log q(\mathbf{x}|t) d\mathbf{x}. \tag{4.9}$$

With this, we utilize the Kullback-Leibler (KL) divergence as a common measure of evaluating the gap between two distinct probability density functions. In particular, the measure specifies the amount of information lost when $q(\mathbf{x}|t)$ is used to approximate $p(\mathbf{x}|t)$, which is given as [106]:

$$\begin{aligned} KL(p||q) &= \int_{\mathcal{X}} p(\mathbf{x}|t) \log \frac{p(\mathbf{x}|t)}{q(\mathbf{x}|t)} d\mathbf{x} \\ &= \int_{\mathcal{X}} p(\mathbf{x}|t)[\log p(\mathbf{x}|t) - \log q(\mathbf{x}|t)] d\mathbf{x} \end{aligned} \tag{4.10}$$

Therefore, maximizing $\int_{\mathcal{X}} p(\mathbf{x}|t) \log q(\mathbf{x}|t) d\mathbf{x}$ in Eq.(4.9), for a given $p(\mathbf{x}|t)$, is equivalent to minimizing $KL(p||q)$. Further, since $KL(p||q) \geq 0$ as per Gibbs' inequality [107], it can be concluded that maximizing the log-likelihood function in Eq.(4.5) minimizes the distribution gap between the mixture model $q(\mathbf{x}|t)$ and $p(\mathbf{x}|t)$, with the gap being bounded from below by zero. The same result holds for the maximization of Eq.(4.6) which is an equivalent form of Eq.(4.5). □

Lemma 4.3. *The EM algorithm for maximizing Eq.(4.6) converges to the global optimum.*

Proof. The EM algorithm converges to a stationary point of the log-likelihood function [108]. Further, it is known that the KL divergence is convex in the domain of probability distributions. With this, it can be seen that since the component models (φ_k's) of Eq.(4.6) are pre-trained, the log-likelihood function must be convex upwards with respect to α_k's. Thus, the stationary point is also the global optimum. □

Theorem 4.4. *Stacked density estimation with all available (source + target) probabilistic models guarantees $KL(p^s||p^c) \leq KL(p^s||p^c_{sub})$, where $KL(p^s||p^c_{sub})$ represents the distribution gap achievable by any proper subset of the models.*

Proof. Let $\mathcal{S}$ denote the set of all available (source + target) probabilistic models.. Thus, $|\mathcal{S}| = K$. Further, let $\mathcal{S}_{sub}$ be any proper subset of $\mathcal{S}$, i.e., $\mathcal{S}_{sub} \subset \mathcal{S}$, and $\mathcal{S}'_{sub} = \mathcal{S} \setminus \mathcal{S}_{sub}$. The mathematical program of Eq.(4.6) given $\mathcal{S}$ can be written as:

$$\max : \log L = \sum_{i=1}^{N} \log \sum_{\varphi_k \in \mathcal{S}} \alpha_k \varphi_k(\mathbf{x}_i). \tag{4.11}$$

Similarly, Eq.(4.6) given $\mathcal{S}_{sub}$ can be written as:

$$\max : \log L_{sub} = \sum_{i=1}^{N} \log \sum_{\varphi_k \in \mathcal{S}_{sub}} \alpha_k \varphi_k(\mathbf{x}_i). \tag{4.12}$$

This is equivalent to:

$$\begin{aligned} \max : \log L_{sub} &= \sum_{i=1}^{N} \log \sum_{\varphi_k \in \mathcal{S}} \alpha_k \varphi_k(\mathbf{x}_i) \\ s.t. \quad \alpha_k &= 0, \forall \varphi_k \in \mathcal{S}'_{sub}. \end{aligned} \tag{4.13}$$

Comparing Eq.(4.11) to Eq.(4.13), and given the result of Lemma 4.3, it is guaranteed that:

$$\log L^* \geq \log L^*_{sub}, \tag{4.14}$$

where (*) indicates the global maximum.

Next, notice that as the offspring population in AMT is sampled from the mixture model, we have:

$$p^c(\mathbf{x}|t) = \sum_{\varphi_k \in \mathcal{S}} \alpha_k \varphi_k(\mathbf{x}), \tag{4.15}$$

and,

$$p^c_{sub}(\mathbf{x}|t) = \sum_{\varphi_k \in \mathcal{S}_{sub}} \alpha_k \varphi_k(\mathbf{x}). \tag{4.16}$$

Further, the training dataset $D_t = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is assumed to be drawn from $p^s(\mathbf{x}|t)$. Therefore, Lemma 1 together with Eq.(4.14) imply that $KL(p^s||p^c) \leq KL(p^s||p^c_{sub})$. □

This result tells us that by combining all available (source + target) probabilistic models, we can reduce the gap between $p^s(\mathbf{x}|t)$ and $p^c(\mathbf{x}|t)$ as compared to using any subset of the models. *In fact, with increasing number of source models, we can in principle make the gap arbitrarily small.* The consequences of such a result towards guaranteeing global convergence behavior of the overall evolutionary algorithm are already substantiated by Theorem 4.1.

At this juncture, it is to be observed that if the target probabilistic model φ_K were to (hypothetically) exactly replicate the parent population's latent probability density function, i.e., $\varphi_K = p^s(\mathbf{x}|t)$, then there would occur no knowledge transfer across problems. This can be shown through Gibb's inequality, which, given $\varphi_K = p^s(\mathbf{x}|t)$, implies that:

$$\int_{\mathcal{X}} p^s(\mathbf{x}|t) \log p^c(\mathbf{x}|t) d\mathbf{x} \leq \int_{\mathcal{X}} p^s(\mathbf{x}|t) \log \varphi_K(\mathbf{x}) d\mathbf{x}. \tag{4.17}$$

Thus, based on the global convergence property of the EM algorithm (as shown in Lemma 2), $\alpha_k = 0$ for $k = 1, \ldots, K-1$, and $\alpha_K = 1$. In other words, the transfer coefficients will be zero, thereby indicating the cancellation of transfer. This suggests that while employing finite population sizes in practical applications of the AMTEA, an extra effort may be needed to prevent overfitting of the target models to the training

dataset D_t. To this end, during every iteration of AMT (refer Section 4.1.3 and Algorithm 3), we propose to artificially add a small amount of random noise to the dataset while learning target probabilistic models.

4.4 Experimental Study

In this section, numerical results are presented that demonstrate the efficacy of the AMTEA framework, with regard to online learning and exploitation of source-target similarities. We conduct experiments across a series of problem categories, ranging from discrete to continuous, as well as single-objective to multi-objective optimization. In addition, a case study on increasingly challenging variants of the double-pole balancing controller design task is carried out, to showcase the practical utility of the method.

For rigorous investigation, the performance of the AMTEA is compared against a number of baseline solvers. First of all, we consider the basic counterpart of the AMTEA, i.e., a canonical EA (CEA) or, alternatively, a canonical memetic algorithm (CMA) if some form of local solution refinement is incorporated. The solution representation scheme employed in the CEA or CMA changes depending on the underlying problem being solved. For the case of multi-objective optimization, the baseline solvers used are the popular NSGA-II [109], and MOEA/D [110]. For a representative knowledge-based (transfer) multi-objective optimizer, the recently proposed *autoencoding evolutionary* (AE) approach [42] is chosen for the comparison study. Labeled herein as AE-NSGAII, the method uses a *denoising autoencoder* to serve as a bridge between the source domain and the search space of the target optimization problem. Finally, as an instantiation of a general purpose transfer evolutionary optimization algorithm, we consider a *transfer case-injected* EA (TCIEA) in which a small number of stored solutions from the source database - those that are similar to the current best target solution - are selected (via case by case assessment) and periodically injected (at transfer interval Δ) into the target evolutionary search [11].

For the double-pole balancing controller design task, a state-of-the-art population-based stochastic optimizer, termed as *natural evolution strategies* (NES) [111], is

adopted as an additional baseline for comparison. It is worth mentioning that NES has recently garnered much attention as a powerful and scalable alternative to reinforcement learning [112].

4.4.1 Experimental Configuration

The experimental setup is outlined as follows. An *elitist* selection strategy is used throughout all experiments with the AMTEA and CEA (or CMA). The (universal) solution representation scheme, and the choice of probabilistic models, are dictated by the underlying problem specifications. For the toy examples (and the additional experiments on Knapsack problems), the following general settings are applied:

1. Representation: Binary-coded;
2. Population size (N): 50 for AMTEA, CEA (or CMA), and TCIEA;
3. Maximum function evaluations: 5,000;
4. Evolutionary operators for AMTEA, CEA (or CMA), and TCIEA:
   (a) Uniform crossover with probability (p_c) = 1;
   (b) Bit-flip mutation with probability (p_m) = 1/d;
5. Probabilistic model: Univariate marginal frequency (factored Bernoulli distribution) [113].

For traveling salesman problems (TSPs), the following settings are used:

1. Representation: $[0, 1]^d$ using random-key encoding scheme;
2. Population size (N): 100 for AMTEA, and CEA;
3. Maximum function evaluations: 10,000;
4. Evolutionary operators for AMTEA, CEA:
   (a) Order crossover;

(b) Random swap mutation;

5. Probabilistic model: Multi-variate Gaussian distribution.

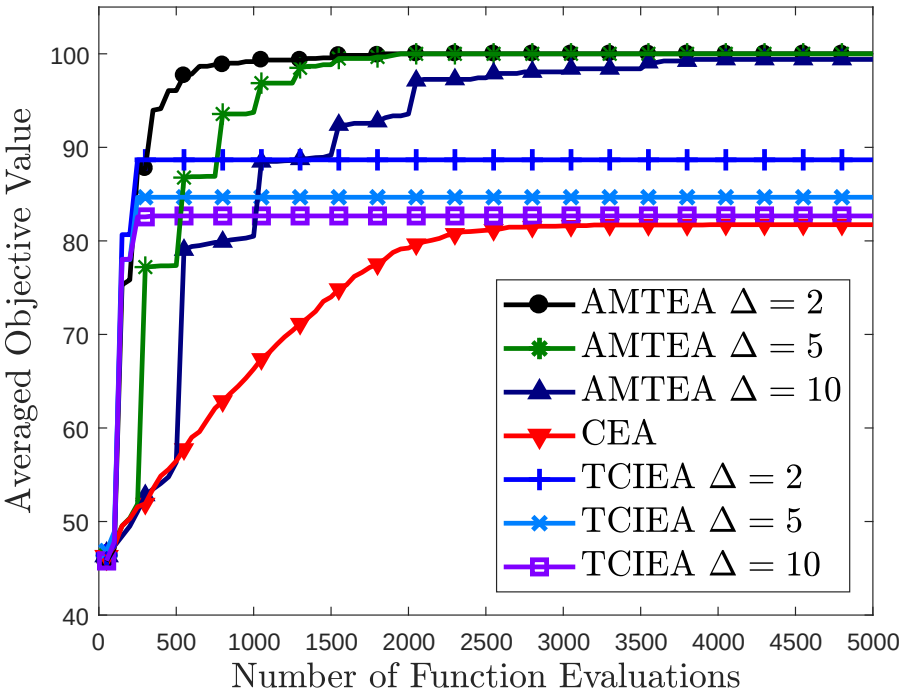
For continuous optimization problems, the experimental setup is outlined as follows. The same settings are incorporated for both single- and multi-objective problems. Note that for multi-objective cases, AMTEA, TCIEA, and AE-NSGAII are built upon the standard NSGA-II. Thus, for fairness of comparison, the genetic operators employed in these solvers are kept identical.

1. Representation: $[0, 1]^d$;
2. Population size (N): 100 for AMTEA, TCIEA, AE-NSGAII, NSGA-II, and MOEA/D;
3. Maximum function evaluations: 10,000;
4. Genetic operators for AMTEA, TCIEA, AE-NSGAII, and NSGA-II:
   (a) Simulated binary crossover (SBX) [114] with $p_c = 1$ and distribution index $\eta_c = 10$;
   (b) Polynomial mutation [115] with $p_m = 1/d$ and distribution index $\eta_m = 10$;
5. Probabilistic Model: Multi-variate Gaussian distribution.

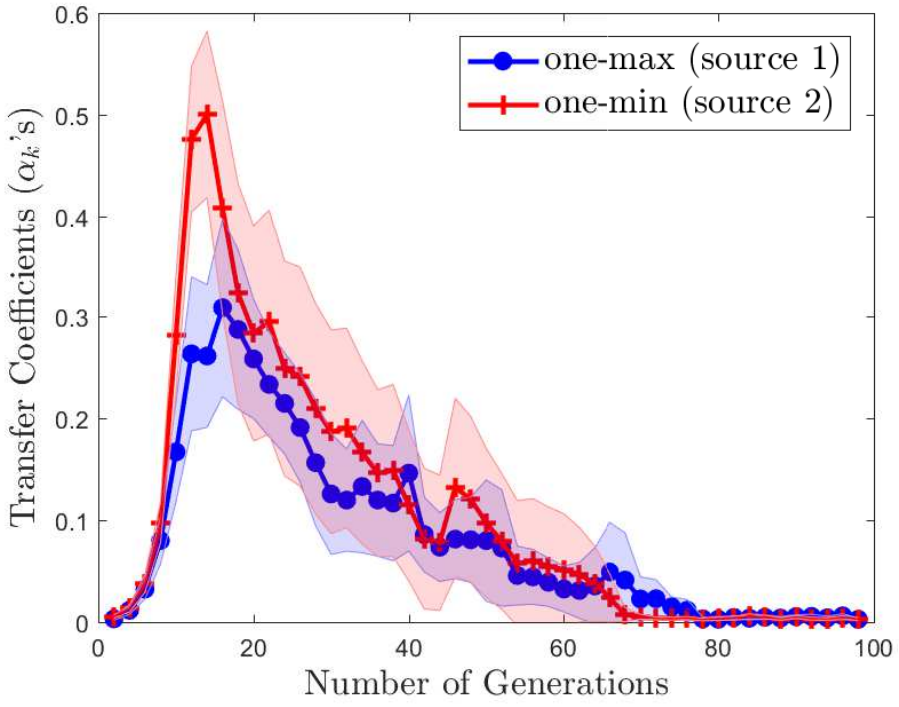
The MOEA/D algorithm incorporates traditional differential evolution operators, with $F = 0.5$ and $CR = 0.5$ [110, 116].

4.4.2 Toy Problems: Functions of Unitation

Functions of unitation encompass a class of discrete problems with binary representation for which the objective depends on the number of ones in a bitstring. For example, the commonly used one-max problem simply states to maximize the number of bits set to one in its chromosome. In contrast, the one-min problem states to maximize the number of bits set to zero (or equivalently, minimize the number of bits set to one) in its chromosome.

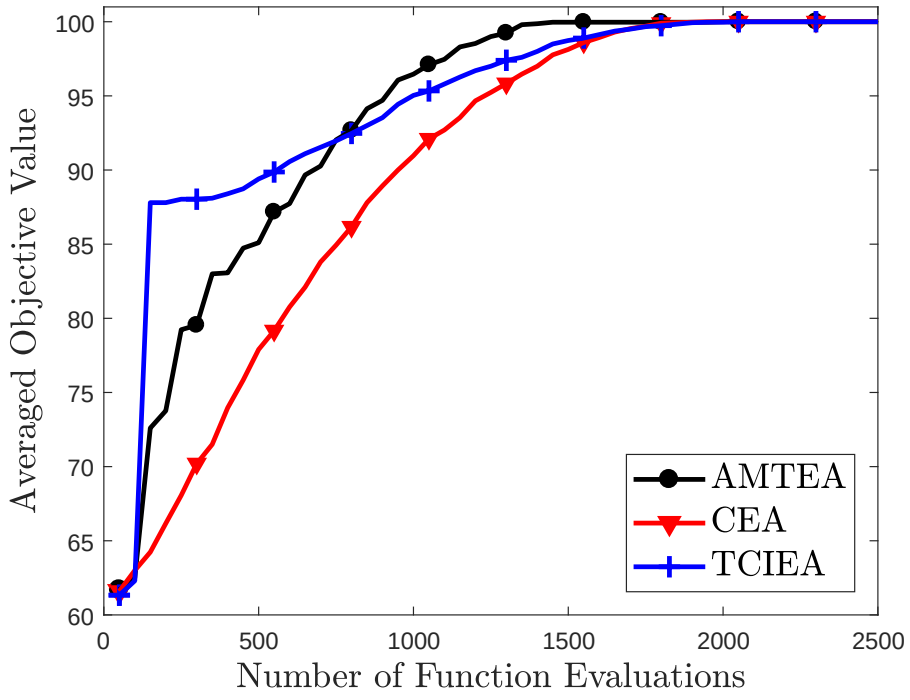


(a) Convergence trends of trap-5 using different Δ's

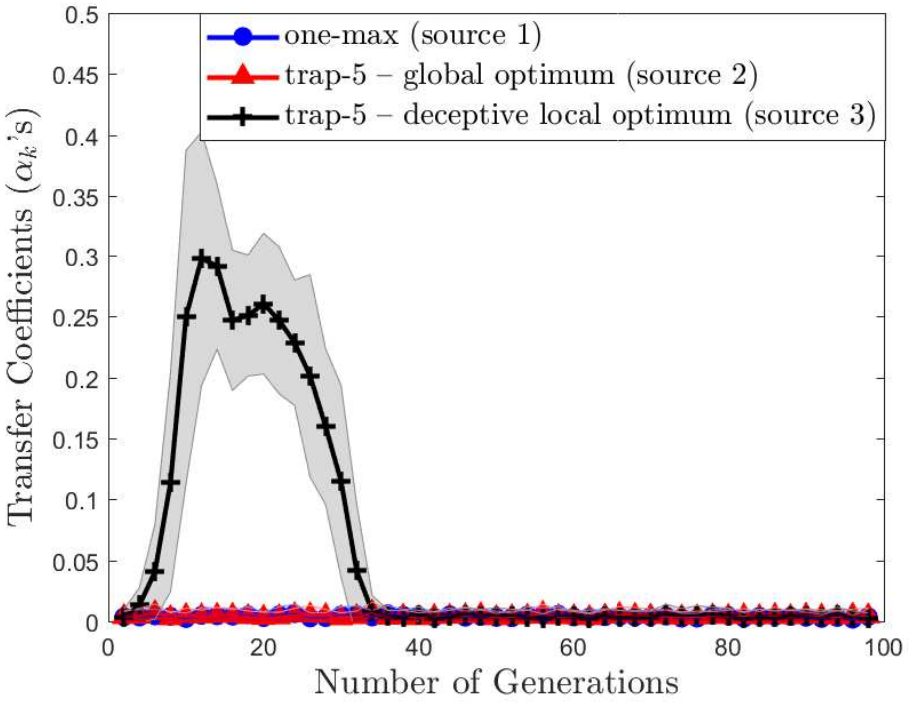


(b) Transfer coefficients learned when $\Delta = 2$

FIGURE 4.2: Convergence trends and transfer coefficients learned for trap-5 (the shaded region spans one standard deviation either side of the mean). One-max and one-min act as source tasks.



(a) Convergence trends of one-min



(b) Transfer coefficients learned

FIGURE 4.3: Convergence trends and transfer coefficients learned while solving one-min (the shaded region spans one standard deviation either side of the mean). Source probabilistic models are drawn from (1) one-max, (2) a trap-5 run where solutions reached the global optimum, and (3) a trap-5 run where solutions were trapped in the deceptive local optimum.

While both the one-max and one-min problems have a single optimum, complex (deceptive) functions of unitation with multiple local optima can also be formulated. A popular example is the trap function of order five, denoted as trap-5 [117]. In trap-5, a candidate bitstring is partitioned into groups of five bits each. The contribution of each group towards the combined objective function is calculated as:

$$f_{trap5}(u) = \begin{cases} 4-u & \text{if } u < 5 \\ 5 & \text{otherwise} \end{cases} \tag{4.18}$$

where u is the number of ones in the group. Trap-5 has one global optimum (when all the bits in the input string equal one), and $2^{d/5} - 1$ other local optima. Note that its global optimum is identical to that of one-max, while the *worst* (highly deceptive) local optimum corresponds to the global optimum of one-min.

In the experimental study, we solve 100 dimensional variants of the trap-5 function and the one-min problem. For the case of trap-5, the source probabilistic models are assumed to come from optimization experiences on (1) one-max and (2) one-min. We set the transfer interval as $\Delta = 2, 5$, and 10 to study the effect of the transfer interval on the global convergence characteristics of the AMTEA. For the case of one-min, source probabilistic models are drawn from (1) one-max, (2) a trap-5 run where solutions reached the global optimum, and (3) a trap-5 run where solutions were trapped at the deceptive local optimum.

The convergence trends for trap-5 are shown in Fig. 4.2 (results are averaged over 30 independent runs). From Fig. 4.2a, it can be seen that CEA always gets trapped at the deceptive local optimum. On the other hand, augmented with the knowledge acquired from prior problem-solving experiences on one-max and one-min, AMTEAs with different transfer intervals show consistently superior performances. Notably, for $\Delta = 2$, 5, the global optimum is achieved in every run. On the other hand, for $\Delta = 10$, the global optimum is reached in 29 out of 30 runs. The minor decrease in performance for a larger transfer interval suggests that there is a tendency that the population gets trapped in a local optimum if no external knowledge is received for a prolonged duration. Under this observation, and keeping in mind the negligible computational cost of

TABLE 4.1: Performances while solving TSPs using different solvers. Superior performance is highlighted in bold.

Methods	TSP100	TSP150	TSP200
CEA	30.2381±2.1574	52.7628±2.0917	74.9706±2.8593
neural network	7.2829	10.3276	11.8314
AMTEA	**6.6340**±0.0292	**8.2646**±0.0688	**9.2924**±0.0753

stacked density estimation given univariate marginal probabilistic models, we set $\Delta = 2$ in all subsequent discrete optimization examples.

Fig. 4.2b shows the trends of the learned transfer coefficients across generations. As indicated by the high values of the transfer coefficients, the target optimization problem can be seen to receive significant transfer from both sources, i.e., one-max and one-min. This implies that the population tends to split into two groups, one of which is inevitably drawn towards the highly deceptive local optimum (under the influence of one-min), while the other manages to converge to the global optimum (under the positive influence of one-max). One aspect of the result in Fig. 4.2a is that TCIEA often gets trapped in a local optimum. This may be attributed to the danger of negative transfer in a case by case solution assessment for transfer procedure, particularly in the absence of any automated source-target similarity modelings.

With regard to one-min as the target problem, its relative simplicity allows all three algorithms, namely, AMTEA, CEA, and TCIEA, to solve it successfully (as shown by Fig. 4.3a). The key point to highlight here is the efficacy of the AMTEA in terms of learning the source-target similarities. As shown by the learned transfer coefficients in Fig. 4.3b, the AMT procedure is able to precisely identify the overlap between the one-min problem and the deceptive local optimum of trap-5 (which is represented by one of the source probabilistic models). The benefits of deciphering such similarities on the fly, without the need for any human intervention, shall be revealed over subsequent subsections in more practically relevant scenarios.

4.4.3 A Case Study on Euclidean Traveling Salesman Problem (TSP)

A TSP is a classical combinatorial optimization problem in the field of computer science. In order to be consistent with the case study in the previous chapter, transfer optimization on TSPs in 2D Euclidean space is studied. It consists of a set of n cities $s = \{\mathbf{x}_i\}_{i=1}^{n}$, we are aiming to find a permutation of the points $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_n)$, termed as a tour, that visits each city once and has the minimum total length. The objective is defined in Eq.(3.1). Different from the previous toy example, random-key encoding scheme is applied to map a solution into the universal space. 3 TSP instances with 100, 150, 200 nodes, respectively, are created in a 2-D Euclidean space, in which the city coordinates are generated according to a Gaussian distribution instead of uniform distribution, to test the generalization ability of the model proposed in Chapter 3,

In this study, when solving one of the 3 TSP instances, the other 2 problems act as source tasks. In addition, the well-trained model proposed in Chapter 3, referred as 'neural network', also serves as a source probabilistic model, as well as a baseline method. Instead of using greedy encoding, we sample 10,000 solutions from the neural network for fair comparison. As naive TCIEA can not handle problems with heterogeneous dimensions, it is not included as baseline in this study.

The numerical results achieved at the end of function evaluations are shown in Table 4.1. Notably, AMTEA is found to consistently outperform the two baseline methods. While CEA is consistently getting stuck at local optima, the proposed AMTEA is able to benefit from fruitful knowledge transfer, perhaps from the well-trained neural network model. To provide further insights to prove our conjuncture, we plot the transfer coefficients for AMTEA. High transfer coefficients learned for the neural network source model indicate the high synergy between the neural network model and the target optimization task. However, the proposed AMTEA is able to continue to explore promising search space in the later stage, leading to consistent superior performances over the predictions of the neural network model.

TABLE 4.2: Averaged IGD values obtained by AMTEA, NSGA-II, TCIEA, AE-NSGAII, and MOEA/D over 30 independent runs. Values in brackets indicate standard deviations. Numbers with (*) indicate the best performing algorithms at 95% confidence level as per the Wilcoxon signed-rank test.

Problem	AMTEA	NSGA-II	TCIEA	AE-NSGAII	MOEA/D
ZDT1	0.0051	0.0095	0.0056	***0.0049**	0.0054
($d = 30$)	(0.0002)	(0.0014)	(0.0003)	(0.0002)	(0.0004)
ZDT2	0.0054	0.0105	0.0055	0.0050	***0.0046**
($d = 30$)	(0.0003)	(0.0015)	(0.0003)	(0.0003)	(0.0001)
ZDT3	***0.0055**	0.0088	0.0058	***0.0055**	0.0198
($d = 30$)	(0.0004)	(0.0013)	(0.0003)	(0.0002)	(0.0197)
ZDT4	0.4479	***0.2926**	0.5751	0.4189	2.0804
($d = 10$)	(0.1973)	(0.1354)	(0.2372)	(0.2285)	(0.6588)
ZDT6	0.0216	0.0723	0.0481	***0.0042**	0.0111
($d = 10$)	(0.0068)	(0.0365)	(0.0196)	(0.0003)	(0.0306)
DTLZ1	***25.8171**	94.3008	69.5432	167.3022	78.9990
($d = 30$)	(33.1267)	(19.3544)	(16.1795)	(21.5871)	(16.9633)
DTLZ2	***0.0730**	0.0932	0.0755	0.0954	0.0783
($d = 30$)	(0.0035)	(0.0063)	(0.0039)	(0.0064)	(0.0065)
DTLZ3	171.2978	225.2871	***160.5694**	477.5128	303.5445
($d = 30$)	124.4363	(57.8071)	(43.8617)	(59.8171)	(76.4256)
DTLZ4	***0.0724**	0.0940	0.0791	0.1000	0.1485
($d = 30$)	(0.0074)	(0.0126)	(0.0409)	(0.0389)	(0.0997)
DTLZ5	***0.3143**	0.4259	0.3857	0.4995	0.4389
($d = 30$)	(0.0275)	(0.0431)	(0.0489)	(0.0887)	(0.1319)
DTLZ6	8.1979	13.7044	12.5287	***1.8073**	4.4750
($d = 30$)	(0.9992)	(0.9382)	(2.4065)	(0.5296)	(1.4279)
DTLZ7	0.1152	0.1148	0.1064	***0.0806**	0.2540
($d = 30$)	(0.0094)	0.0148	(0.0496)	(0.0051)	(0.1833)
WFG1	***1.0705**	1.0807	1.0719	1.0986	1.1655
($d = 24$)	(0.0076)	(0.0099)	(0.0100)	(0.0074)	(0.0205)
WFG2	***0.0290**	0.1036	0.0697	0.0938	0.2460
($d = 24$)	(0.0224)	(0.0209)	(0.0152)	(0.0227)	(0.0826)
WFG3	***0.1464**	0.1924	0.1623	0.1945	0.1731
($d = 24$)	(0.0014)	(0.0162)	(0.0067)	(0.0111)	(0.0133)
WFG4	***0.0173**	0.0357	0.0274	0.0362	0.0794
($d = 24$)	(0.0010)	(0.0045)	(0.0023)	(0.0045)	(0.0094)
WFG5	0.0732	0.0729	0.0736	0.0725	***0.0696**
($d = 24$)	(0.0013)	(0.0014)	(0.0011)	(0.0013)	(0.0006)
WFG6	***0.0195**	0.0875	0.0380	0.0957	0.0960
($d = 24$)	(0.0010)	(0.0113)	(0.0155)	(0.0138)	(0.0211)
WFG7	***0.0185**	0.0267	0.0228	0.0294	0.0271
($d = 24$)	(0.0013)	(0.0035)	(0.0015)	(0.0032)	(0.0021)
WFG8	0.1943	0.1635	0.1708	0.1671	***0.1501**
($d = 24$)	(0.0115)	(0.0062)	(0.0080)	(0.0109)	(0.0090)
WFG9	***0.0257**	0.0918	0.0349	0.0913	0.0954
($d = 24$)	(0.0017)	(0.0437)	(0.0178)	(0.0420)	(0.0321)

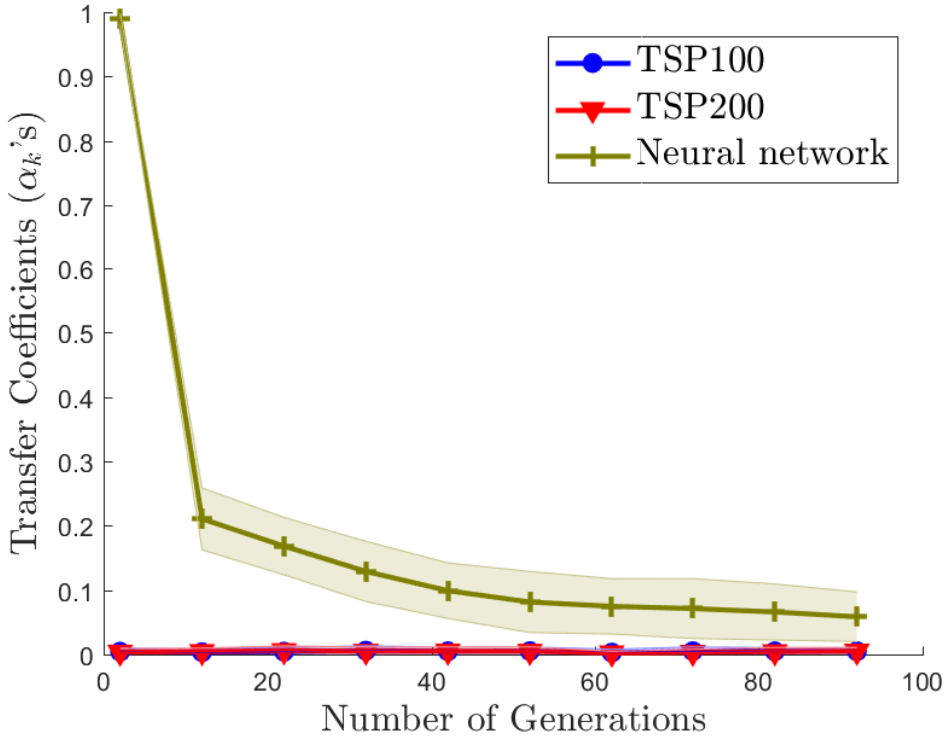
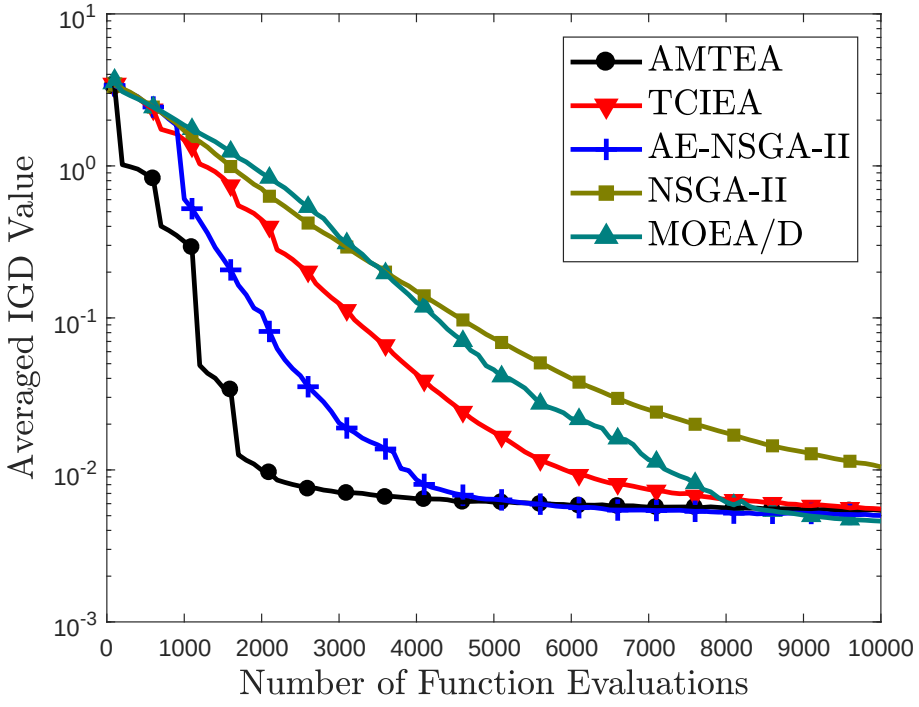


FIGURE 4.4: Transfer coefficients learned for TSP150, with TSP100, TSP200, and neural network serving as source tasks. Notice that the source-target similarities learned between TSP150 and neural network is particularly high, which partially explains the superior performance of AMTEA over CEA.

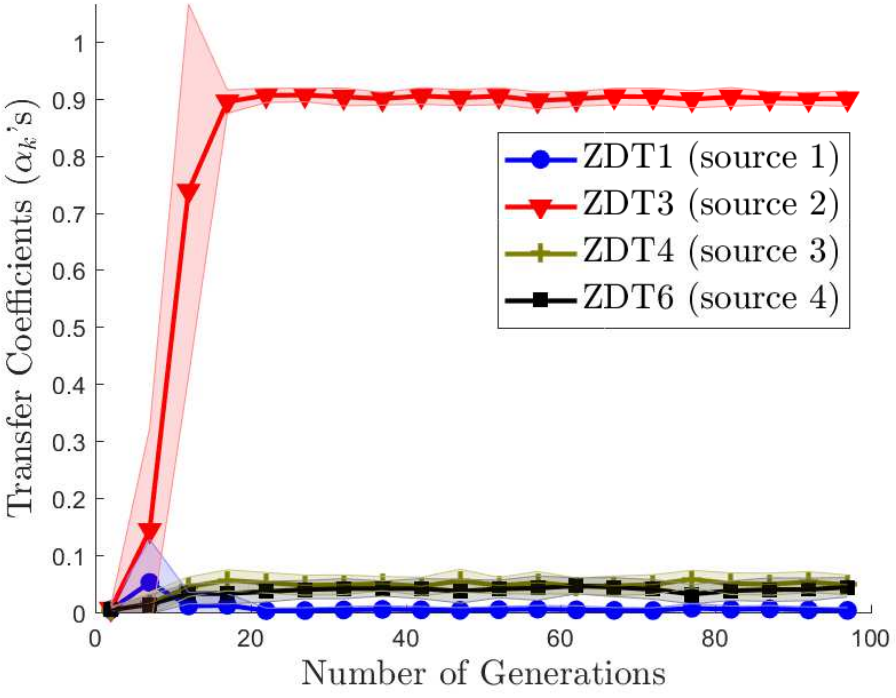
4.4.4 A Case Study in Multi-Objective Optimization (MOO)

Population-based EAs have gained popularity over the years as noteworthy solvers of MOO problems, primarily due to their ability (derived from the implicit parallelism of the population) to simultaneously converge towards the entire set of optimal solutions (commonly referred to as Pareto optimal solutions) of MOO problems [110]. Endowing multi-objective evolutionary algorithms (MOEAs) with knowledge transfer capabilities, is therefore expected to further push the envelope of evolutionary methods in this domain.

In this study, numerical experiments are carried out on three popularly used MOO benchmark test sets, namely ZDT (ZDT1-4 and ZDT6) [118], DTLZ [119], and WFG [120]. When solving any one of the problems in the three test sets, using the AMTEA, TCIEA or AE-NSGAII, all the other problems in that set act as source tasks

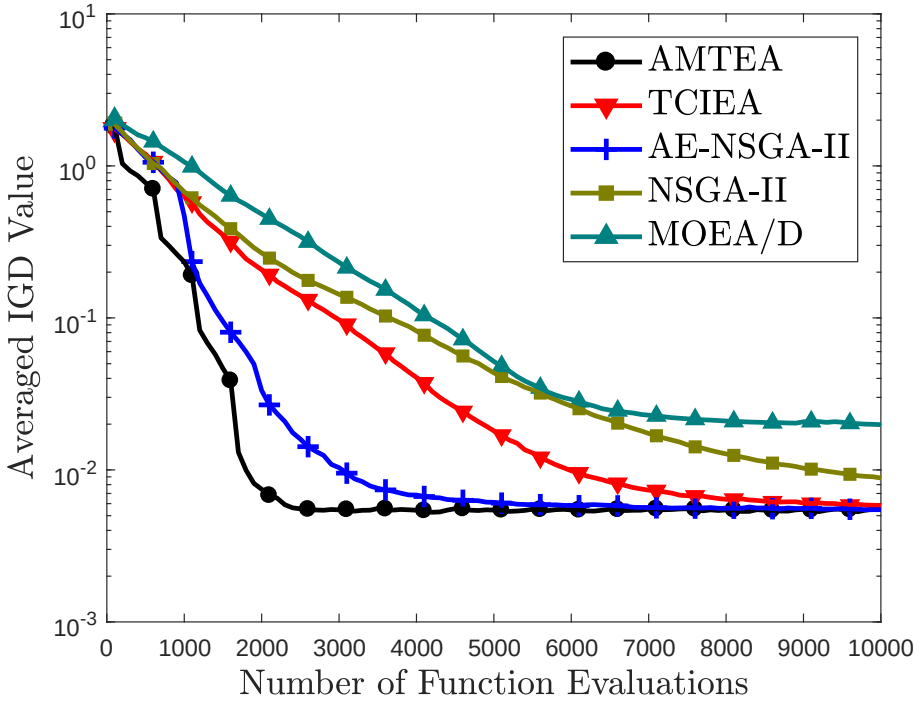


(a) ZDT2

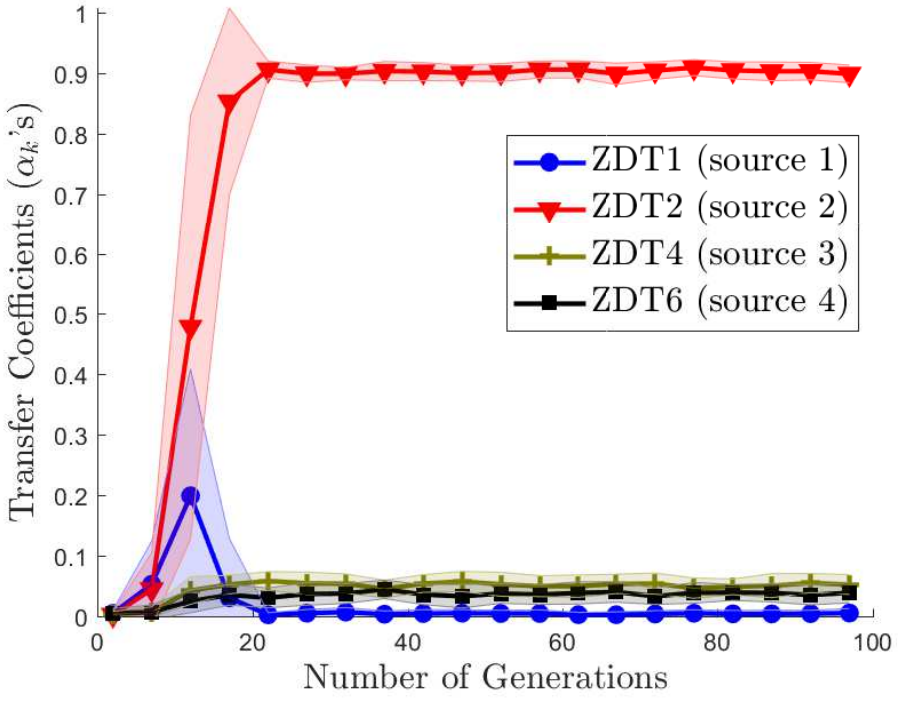


(b) Transfer coefficients learned for ZDT2

FIGURE 4.5: Convergence trends and transfer coefficients learned for ZDT2, with all other problems in the ZDT family serving as source tasks. Notice that the source-target similarities learned between ZDT2 and ZDT3 is particularly highly, which can be explained by the fact that these two problems have similar Pareto optimal solutions.



(a) ZDT3



(b) Transfer coefficients learned for ZDT3

FIGURE 4.6: Convergence trends and transfer coefficients learned for ZDT3, with all other problems in the ZDT family serving as source tasks. Notice that the source-target similarities learned between ZDT3 and ZDT2 is particularly highly, which can be explained by the fact that these two problems have similar Pareto optimal solutions.

TABLE 4.3: Performances while solving the double-pole balancing problem using different solvers. Superior performance is highlighted in bold.

Methods	Successes	Function Evaluations
CEA	0/50	NA
NES	1/50	8977
TCIEA	1/50	8900
AMTEA	**22/50**	**7918**$\pm$1241

providing source probabilistic models for transfer. Keeping in mind the modest computational cost involved in the stacking of *multivariate* Gaussian distribution models, the transfer interval is relaxed to $\Delta = 10$ in these experiments. The inverted generational distance (IGD) [121] is considered as the performance metric for comparisons.

The numerical results achieved at the end of 10,000 function evaluations are shown in Table 4.2. The non-parametric Wilcoxon signed-rank test reveals that AMTEA performs significantly the best in a majority of the test problems (precisely, in 12 out of 21 test problems). Notably, AMTEA is found to often surpass the performance of the recently proposed autoencoding-based transfer evolutionary optimizer AE-NSGAII. Fig. 4.5 and 4.6 show the accelerated convergence characteristics achieved on ZDT2 and ZDT3 by the AMTEA, in comparison to other baseline algorithms.

On further inspection, it is revealed based on the equations of ZDT2 and ZDT3 [109], that the Pareto optimal solutions of the two tasks are indeed similar to each other when mapped to the universal search space. This latent property is automatically deciphered and harnessed by the AMTEA, as can be seen in Figs. 4.5b and 4.6b where the proposed algorithm is found to consistently identify high transfer coefficients between these two tasks.

4.4.5 The Double-Pole Balancing Controller Design Task

Double-pole balancing is a controller design task popularly used as a case study for reinforcement learning algorithms. The goal is to prevent two poles, both affixed at the same point on a cart (which is restricted to a finite stretch of track), from falling over by applying a force to the cart [122].

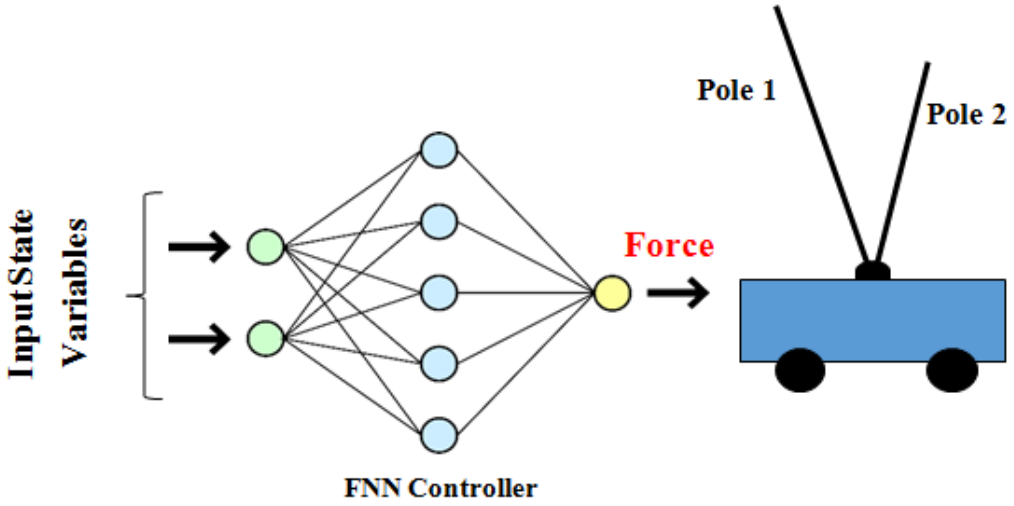


FIGURE 4.7: Setup of the double-pole balancing problem with a feedforward neural network (FNN) controller.

In the problem setup, the state of the system can be fully defined by six variables: the angle of each pole from vertical, the angular velocity of each pole, the position of the cart on the track, and the velocity of the cart (see [123] for the equations of motion and parameters used in this task). The length of the long pole is set to 1 meter, and e_l is used to denote the task where l is the length of the shorter pole (also in meters). The Runge-Kutta fourth-order method is used to numerically simulate the system, with a step size of 0.01 seconds. During simulations, a simple feedforward neural network (FNN) controller, with fixed structure, is used to output a force that acts on the cart every 0.02 seconds in the range of $[-10, 10]N$. The initial angle of the long pole is set to $1°$. The control task is considered to be a failure if the cart goes out of bounds of a 4.8 meter track, or else, one of the two poles drops beyond $\pm 36°$ from the vertical. The fitness of a candidate solution (which encodes the synaptic weights of the FNN controller) is the number of time steps taken for the system to fail. Evidently, the problem is that of fitness maximization. Note that a task is considered solved if the fitness of the corresponding solution is over 100,000 time steps, which is approximately 30 minutes in simulated time.

From previous studies, it is well-established that the double-pole system becomes more difficult to control as the poles assume similar lengths [124], i.e., as the length of the shorter pole l approaches 1 meter. A number of optimization efforts with state-of-the-art solvers were performed to verify that the problem indeed becomes progressively

harder to solve within a reasonable amount of time as the length of the shorter pole approaches 0.8 meters. Thus, in the context of AMT, it is natural to raise the question: *Is it possible to utilize previous problem-solving experiences on easier problems to help solve increasingly challenging variants of a problem at hand?*

To investigate this matter further, probabilistic models drawn from source optimization tasks with $l = 0.1, 0.2, \ldots, 0.775$ (13 source tasks in all) are used while tackling $e_{0.8}$ as the target task. A two-layer FNN controller with 10 hidden neurons is applied. Bias parameters are removed due to the symmetry property of the system. This leads to a total of 70 weights to be tuned by the optimizer.

In the experimental study, *success rate* is the performance metric used to compare AMTEA against TCIEA, CEA, and the recently proposed NES algorithm[1]. According to the results tabulated in Table 4.3, the CEA can never achieve success in any of its 50 independent runs. Even NES and TCIEA succeed only once out of their 50 runs, thereby demonstrating the considerable difficulty of $e_{0.8}$. On the other hand, the AMTEA achieves significantly higher success rate, effectively balancing the pole in 22 out of 50 runs. Notably, among the successfully completed runs, the averaged number of function evaluations consumed by AMTEA is approximately 7918, while that consumed by TCIEA and NES is more than 8900.

With regard to the transfer coefficients learned in the AMTEA, it is referred to Fig. 4.8. Note that the figure only presents transfer coefficients corresponding to 5 (out of the 13) source optimization problems, as the remaining sources showcase consistently low knowledge transfer. It can be seen that a lot of transfer occurs to the target task $e_{0.8}$ from sources $e_{0.75}$ and $e_{0.775}$. This is once again a noteworthy example where the AMTEA is automatically able to identify what are intuitively expected to be the most relevant sources of information. Furthermore, the considerably superior performance of AMTEA as compared to TCIEA substantiates the impact of the proposed approach in enhancing optimization performance in general. The benefits of online source-target similarity modeling, as opposed to a case by case solution assessment for transfer procedure, with no similarity learning capability, are amply revealed.

[1]with default parameter setting as available from http://people.idsia.ch/~tom/nes.html

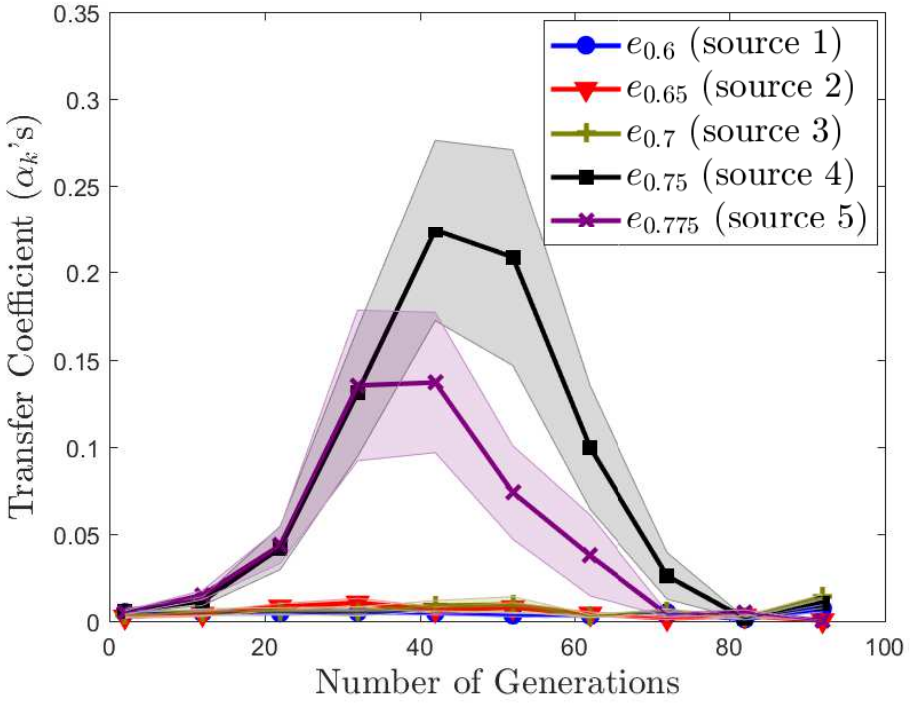


FIGURE 4.8: Transfer coefficient trends learned by AMTEA while solving $e_{0.8}$. The shaded region spans one standard deviation either side of the mean.

4.4.6 Engineering Design

Two distinct composites manufacturing techniques, that belong to the same family of rigid-tool liquid composite molding (LCM) processes [125], are considered. To elaborate, (1) resin transfer molding (RTM) and (2) injection/compression LCM (I/C-LCM) are popular techniques for high volume production of of synthetic fiber-reinforced polymer (FRP) composite parts, and are characterized by the use of high stiffness molds. Under the assumption of rigidity, the molds are expected to undergo negligible deflection in response to the large internal forces that originate from the cumulative effect of high fluid (polymer resin) pressure and compression of the fibrous reinforcement. Thus, rigid-tool LCM processes find numerous applications in areas requiring high geometrical precision, such as the automobile and aerospace industries. Often, sophisticated peripheral (such as a hydraulic press) is needed to equilibrate the internal forces [126, 127]. As a result, an important aim in the optimal design of an LCM process is to maximize throughput (by minimizing manufacturing time), while satisfying

the (often stringent) constraints placed by the availability or cost of peripheral equipment.

Next, the RTM and I/C-LCM processes are introduced briefly.

4.4.6.1 Resin Transfer Molding

The setup of the RTM process, as illustrated in Fig. 4.9, typically consists of a metallic mold machined according to the geometry of the composite part to be manufactured. The first step is to place a preform of the fibrous reinforcement into the mold cavity (see Fig 4.9a). The mold is then completely closed, fully compressing the preform to the final to the final thickness of the part (see Fig. 4.9b). Before liquid resin injection, the mold is heated to a desired (optimum) operation temperature. Then, a thermosetting resin is injected into the closed mold at high pressure until the resin reaches the vents (see Fig. 4.9c). The filled mold is then allowed to rest until the resin solidifies, followed by extraction of final part (see Fig. 4.9d). The optimization of the RTM cycle introduces introduces four design variables, namely: 1) speed of mold closure (V_{closure}); 2) resin injection pressure (P_{inj}); 3) preheated mold temperature (T_{mold}); 4) preheated resin temperature (T_{resin}). Thus a design vector for the RTM cycle can be summarized as $(V_{\text{closure}}, P_{\text{inj}}, T_{\text{mold}}, T_{\text{resin}})$.

4.4.6.2 Injection/Compression Liquid Composite Molding

While mold filling in the RTM is generally viewed as a single phase process, the same occurs in a two-phase manner in I/C-LCM. As illustrated in Fig. 4.9, during I/C-LCM, the mold is only partially closed prior to resin injection (as shown in Fig. 4.9b). After the required volume of liquid resin has been injected into the (partially) open mold (see Fig. 4.9c), the mold is fully closed to the desired part thickness using a velocity-controlled mechanism (see Fig. 4.9d). Due to the inclusion of the *in situ* mold closure phase (see Fig. 4.9d), the I/C-LCM cycle introduces two additional design variables, namely: 1) mold cavity thickness during resin injection (H_{inj}) and 2) velocity of final mold closure ($T^{\text{final}}_{\text{closure}}$). Thus, a design vector for the I/C-LCM cycle can be summarized as $(V^{\text{initial}}_{\text{closure}}, P_{\text{inj}}, T_{\text{mold}}, T_{\text{resin}}, H_{\text{inj}}, T^{\text{final}}_{\text{closure}})$.

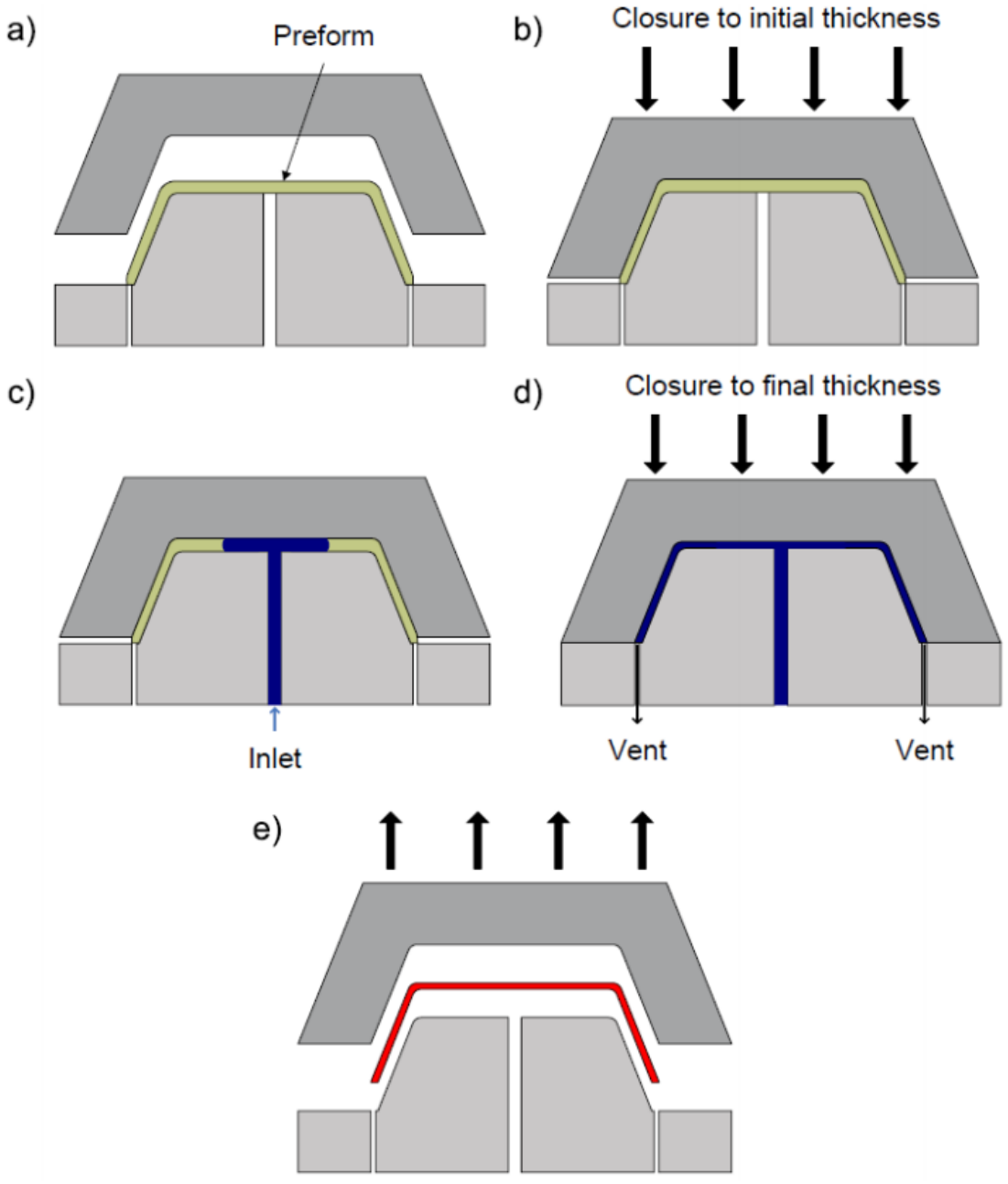


FIGURE 4.9: Workflow of I/C-LCM.

While deciding on a manufacturing technique for a particular composite part, the manufacturer must throughly investigate both processes to determine the most suitable in terms of minimizing capital layout and running costs while maximizing throughput. In most cases, investigation is performed by coupling process simulation software with an optimization engine. To this end, the formulation of the constrained optimization problem may be stated as:

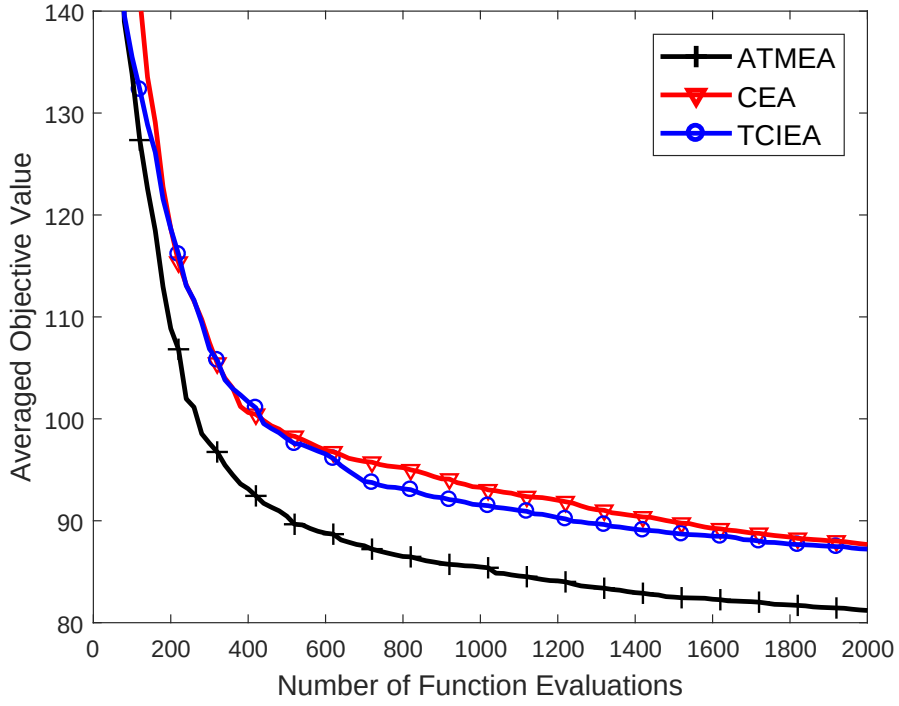
$$\begin{aligned} &\text{minimize} \ \ (\text{mold filling time}), \\ &\text{subject to: } F_{\text{fluid}} + F_{\text{fiber}} \leq F_{\text{capacity}}. \end{aligned} \tag{4.19}$$

Here, F_{fluid} represents the internal force originating from the high pressure resin injection, F_{fiber} is the response of the compressed fibrous reinforcement, and F_{capacity} is the maximum allowable internal force as dictated by the availability of peripheral equipment. The values of F_{fluid}, F_{fiber}, mold filling time, and peak internal force (for a given combination of input design variables) are obtained via a process simulation software [128, 129] which evaluates a set of partial differential equations that govern the complex non-isothermal and chemically reactive resin flow through porous media. As is well known, such simulations are generally computationally intensive, often take several minutes for a single evaluation of sufficiently high fidelity. This feature presents a considerable roadblock to efficient optimization.

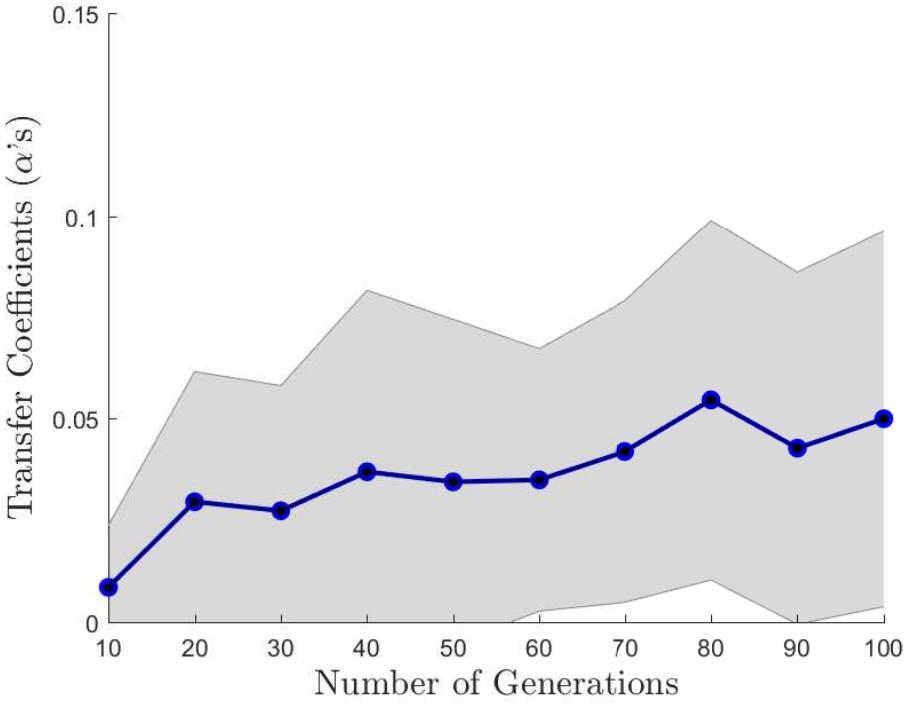
The described composites manufacturing problem provides an ideal setting for us to explore the applicability of knowledge meme transfer in real-world problems. Since RTM and I/C-LCM belong to the same family of rigid-tool LCM processes and have several recurring design variables, it is expected that there exists some knowledge transfer between the two techniques, especially when dealing with the manufacture of the same FRP composite part.

The detailed experimental setup is listed as follows:

1. Population size: 20 for ATMEA, NSGA-II and TCIEA;
2. Maximum function evaluations: 2,000;



(a) Convergence trends



(b) Coefficient trends

FIGURE 4.10: Convergence trends using different methods and coefficient trends of ATMEA during the evolution process on complex engineering design problem.

TABLE 4.4: Extent of the Search Space for the RTM and I/C-LCM Design Variables

Design Variable	**Lower Bound**	**Upper Bound**
$V_{\text{closure}}^{\text{initial/final}}$	1 mm/min	10 mm/min
P_{inj}	1 MPa	10 MPa
T_{mold}	293 K	348 K
T_{resin}	293 K	348 K
H_{inj}	0.8 cm	1 cm

3. Evolutionary operators for ATMEA, CEA and TCIEA:

   (a) SBX operator: $\eta_c = 20$;

   (b) Polynomial mutation: $p_m = 1/d_T$, $\eta_m = 20$;

4. Probabilistic Model: multi-variate Gaussian distribution;

5. Transfer interval: $\Delta = 10$.

4.5 Conclusion

In this chapter, a novel evolutionary computation paradigm is proposed, inspired by observed human-like problem-solving capabilities of seamlessly applying the knowledge acquired from previous experiences to new and more challenging tasks. As in machine learning, where the concept of transfer learning allows data from related source tasks to be re-used for improving predictive performance on the target task, the goal of the present study is to develop a theoretically principled realization of black-box *transfer optimization*.

To elaborate, the proposal enables online learning and exploitation of source-target similarities, potentially revealing latent synergies even during the course of the optimization search. In the proposal, the knowledge acquired from past optimization exercises is encoded in the form of probabilistic models that bias the search towards promising solutions. The modulation of the amount of transfer between multiple sources and the target task of interest is achieved through an adaptive model-based transfer (AMT)

procedure, where automatic learning of the optimal blend of source and target probabilistic models is carried out via *stacked density estimation*. Notably, the method eliminates the need for any human intervention or ad-hoc rules for ascertaining source-target similarities. An instantiation of the framework as a nested module within a canonical EA, labeled as an AMT-*enabled* EA (or AMTEA), is introduced. Subsequently, a theoretical analysis is provided to substantiate the impact of the proposal in facilitating improved performance of the transfer optimization algorithm.

In addition to presenting theoretical justifications, a series of numerical experiments on benchmark and practical examples, covering discrete and continuous domains, as well as single-objective and multi-objective optimization, were carried out to test the efficacy of the AMTEA[1]. The results showed that while existing techniques (including those endowed with the scope of knowledge transfer) often suffered from premature convergence at local optima of complex tasks, the AMTEA was able to automatically distinguish positive and negative transfers when faced with multiple sources, thereby leading to consistently superior performance.

[1] Some additional experiments are in the appendix

Chapter 5

Multi-Source Surrogate-Assisted Transfer Optimization for Computationally Expensive Problems[1]

The previous proposed transfer optimization techniques are demonstrated to have enhanced performance over the state-of-the-art, under the common assumption that the target optimization problems are cheap to evaluate. However, real-world optimization tasks can be extremely computational expensive. A single function evaluation can cost hours or even days. Optimization solvers involving iterative evaluations on such optimization tasks may not be practical to deploy. Therefore, various surrogate-assisted optimization techniques are proposed in the literature to reduce the number of function evaluations on the original computationally expensive problems. The basic principle of the surrogate-assisted optimization algorithms is to construct a cheap surrogate model approximating the original optimization task, hence optimization can be conducted on the surrogate model to find the next promising candidate to be evaluated on the original task. If the surrogate model approximates original function sufficiently well, optimizing the surrogate model is essentially equivalent to optimizing the original optimization task. Therefore, the approximation quality of the surrogate model is crucial to the convergence property of the corresponding surrogate-assisted optimization algorithm.

[1]Partial resutls of the presented work have been accpted in [25].

One of the most commonly used surrogate model is perhaps Gaussian process (GP), as it provides predictive mean as well as the associated variance. The posterior distribution provided by GP model is essential to design an acquisition function (or infill sampling criteria) to balance the tradeoff between exploitation and exploration. Most recently, the Gaussian process lower confidence bound (GP-LCB) [71] was proposed as a popular acquisition function with provable cumulative regret bounds. As a result, Gaussian process is applied as surrogate model throughout this chapter.

As shown in Fig. 2.1 in Chapter 2, function evaluations on previous optimization tasks can be adaptively re-used to build a more accurate surrogate model. Thus, building a more accurate transfer regression model becomes crucial. Several research papers have been recently published towards this direction [13, 72]. However, with the increasing number of source optimization tasks gathered over time, building such a surrogate model can be impractical, since the complexity of Gaussian process based multi-task/transfer learning approaches grows cubically with the total number of source+target observations.

It is found that in single-task learning, aggregation models, such as Bayesian committee machine [130], product of experts [131, 132], mixture of experts [133, 134], and so on, have been well-studied to reduce the computational burden of large-scale GPs. These methods generally involve partitioning the training inputs into local blocks or clusters, then modeling each block with an independent GP as a local expert. If the blocks are spatially localized, the overall model corresponds to a covariance function that imposes independence between output values in different regions of the input space. In comparison to the sparse approximation methods, the aggregation models (1) do not require any additional inducing or variational parameters, (2) allow straightforward parallelization to distribute the computations on individual experts, and (3) maintain similar local expressiveness as the full GP for functions with notable local structures. However, naively applying these aggregation methods into transfer Gaussian processes (TGPs) may not be practical due to insufficient target inputs. Target data is scarce and precious in transfer learning, and hence partitioning the target inputs will probably cause significant information loss globally. Therefore, in this chapter, a novel factorized training strategy is proposed for transfer learning, in which the target data is fully utilized

within each local expert, even as the advantages of lower computational complexity and straightforward parallelization of classical aggregation models is retained. In particular, armed with a set of trained local experts, a principled method, labeled as *transfer Bayesian Committee Machine* (Tr-BCM), is proposed, to combine their respective predictions. It will be demonstrated that the proposed model fully utilizes the scarce target inputs to ensure the predictive performance of each local expert is superior to the performance of a single-task GP trained on target data only. As far as is known, this is the first work introducing aggregation models in the setting of transfer learning. The efficacy of Tr-BCM is verified on toy examples as well as real-world applications.

As a notable byproduct of the aggregation model-based approach, it is found that that Tr-BCM offers enhanced expressiveness in multi-task/transfer GPs by enabling the capture of localized inter-task relationships. According to recent studies [135, 136], the efficacy of enhancing model expressiveness has been well-established. In the context of knowledge transfer in particular, while a given pair of tasks may be resolved as being globally uncorrelated, there may exist local subspaces characterized by strong correlation. Nevertheless, naively extending a full TGP model to learn localized source-target relationships is proved in this chapter to have no guarantee to produce a positive semi-definite (PSD) covariance matrix. However, such issues can be easily circumvented by applying aggregation models as shall be illustrated later on. What is more, it is revealed that this salient feature of Tr-BCM applies with little/no modification to the case of multi-source transfer learning problems as well; thereby highlighting the generality of the proposed method in practice.

Furthermore, during black-box optimization process, it is often the case that we have many already solved or cheap source optimization tasks, while the optimization problem of practical interest is very expensive to evaluate. In this case, building a full multi-task/transfer GP model as a surrogate might be implausible since the observations from source optimization problem might be abundant, causing too much computational overhead to train and inference on such a multi-task/transfer GP model. However, the proposed model can easily overcome this barrier, with little sacrifice of the expressiveness of the surrogate model.

To summarize, the following salient features make the proposed model an attractive proposition for the domain of transfer learning as well as transfer optimization:

- A new factorized training strategy and principled aggregation model are proposed, namely Tr-BCM, for transfer learning, in order to accelerate full TGP with large-scale source inputs. The theoretical behaviors of the proposed Tr-BCM model in comparison to other naive extensions of model aggregation schemes are analyzed in detail.
- Flexible/non-uniform source-target similarity capture is made possible through the proposed Tr-BCM. Therefore, the expressiveness of the proposed model is increased, and negative transfer is mitigated if the source-target similarity indeed varies drastically in the input space.
- Further, a hierarchical structure is proposed to extend Tr-BCM for dealing with transfer learning problems with multiple sources. Thus, the practical generality of Tr-BCM is greatly increased.
- When applying Tr-BCM in real-world applications, each local expert corresponds to a lightweight predictor that can be embedded in *edge* devices, thus catering to cases of online on-mote processing [137, 138].
- Finally, the proposed aggregation model is applied in the framework of Bayesian optimization. Empirical performance once again demonstrates the superiority of the proposed model.

For a detailed exposition about the proposed model and the empirical investigation of its efficacy, the rest of the chapter is organized as follows. First, a general introduction of TGP is offered in Section 5.1, following which the proposed Tr-BCM strategy is presented in order to decrease the computational burden of traditional TGP in Section 5.2. Non-uniform source-target relationship capture and multi-source transfer learning problems are studied in Section 5.3 and 5.4, respectively. In the empirical studies of Section 5.5, numerical experiments on real-world datasets highlight the benefits of the proposed method in comparison to existing transfer learning approaches. In Section 5.6,

the proposed Tr-BCM is applied in the framework of multi-source transfer Bayesian optimization.

5.1 Preliminary

In this section, a brief overview of the TGP model proposed in [81] is introduced.

5.1.1 Problem Specification

We first consider transfer regression problems with a single source task and a single target task. The dimensionality of the source and target inputs is set to d. Assume that a large source input set $\mathbf{X}_{\mathcal{S}} \in \mathbb{R}^{n_{\mathcal{S}} \times d}$ and the corresponding labels $\mathbf{y}_{\mathcal{S}} \in \mathbb{R}^{n_{\mathcal{S}}}$ are available for the source task $\mathcal{S}$, labeled as $\mathcal{D}_{\mathcal{S}} = \{\mathbf{X}_{\mathcal{S}}, \mathbf{y}_{\mathcal{S}}\}$. In contrast, the inputs $\mathbf{X}_{\mathcal{T}} \in \mathbb{R}^{n_{\mathcal{T}} \times d}$ and the corresponding labels $\mathbf{y}_{\mathcal{T}} \in \mathbb{R}^{n_{\mathcal{T}}}$ available for the target task $\mathcal{T}$ are relatively scarce (i.e., $n_{\mathcal{T}} \ll n_{\mathcal{S}}$). The overall target dataset is denoted as $\mathcal{D}_{\mathcal{T}} = \{\mathbf{X}_{\mathcal{T}}, \mathbf{y}_{\mathcal{T}}\}$. Generally, given the input $\mathbf{x}$, the source and target outputs are modeled as:

$$y_{\mathcal{S}} = f_{\mathcal{S}}(\mathbf{x}) + \epsilon_{\mathcal{S}},$$
$$y_{\mathcal{T}} = f_{\mathcal{T}}(\mathbf{x}) + \epsilon_{\mathcal{T}},$$

where the additive noise terms $\epsilon_{\mathcal{S}}$ and $\epsilon_{\mathcal{T}}$ are assumed to be independent, identically distributed (i.i.d.) Gaussian distributions with zero mean and variance $\sigma_{\mathcal{S}}^2$ and $\sigma_{\mathcal{T}}^2$, respectively; $f_{\mathcal{S}}$ and $f_{\mathcal{T}}$ are the latent functions of the corresponding tasks. The objective is to transfer knowledge from the source task $\mathcal{S}$, so as to improve the generalization performance of a predictive model over target task $\mathcal{T}$.

5.1.2 Transfer Gaussian Process

GP is a popular stochastic, nonparametric approach for regression. It describes a distribution over functions, given as $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $\mu(\mathbf{x})$ is the mean function (typically we set $\mu(\mathbf{x}) = 0$) and $k(\cdot, \cdot)$ is some valid covariance function. To

be valid, any Gram matrix derived from kernel $k(\mathbf{x}, \mathbf{x}')$ is required to be PSD. Popular kernel functions include squared exponential (SE) and Matérn kernel. GP is a stochastic process wherein any finite subset of random variables follows a joint multivariate Gaussian distribution. Therefore, for a standard single-task GP, given the observations $\mathcal{D}_{\mathcal{T}} = \{\mathbf{X}_{\mathcal{T}}, \mathbf{y}_{\mathcal{T}}\}$ on the target task, the posterior distribution at a particular test point $\mathbf{x}_*$ is efficiently obtained [73].

In order to take advantage of abundant and perhaps correlated source data, Cao *et al.* [81] proposed the TGP model to achieve adaptive knowledge transfer while retaining the advantages of a standard GP model. The key distinguishing feature of the TGP model is the description of the following transfer covariance function:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \begin{cases} \lambda k(\mathbf{x}, \mathbf{x}'), & \mathbf{x} \in \mathbf{X}_{\mathcal{S}} \ \& \ \mathbf{x}' \in \mathbf{X}_{\mathcal{T}} \\ & \text{or } \mathbf{x} \in \mathbf{X}_{\mathcal{T}} \ \& \ \mathbf{x}' \in \mathbf{X}_{\mathcal{S}} \\ k(\mathbf{x}, \mathbf{x}'), & \text{otherwise.} \end{cases} \tag{5.1}$$

Here, the additional parameter λ measures the source-target similarity. According to Theorem 1 in [81], $\tilde{k}(\cdot, \cdot)$ is a valid kernel for all $|\lambda| \leq 1$. If $|\lambda|$ is close to 1, it indicates that the source and target tasks are highly correlated.

As for the inference process of TGP, it is very similar to that of standard GP. In particular, the mean and the associated variance at an unknown target input $\mathbf{x}_*$ is given by:

$$\begin{aligned} \mu(\mathbf{x}_*) &= \tilde{\mathbf{k}}_{\mathbf{x}_*}(\widetilde{\mathbf{K}} + \Lambda)^{-1}\mathbf{y}, \\ \sigma^2(\mathbf{x}_*) &= \tilde{k}(\mathbf{x}_*, \mathbf{x}_*) - \tilde{\mathbf{k}}_{\mathbf{x}_*}^{\intercal}(\widetilde{\mathbf{K}} + \Lambda)^{-1}\tilde{\mathbf{k}}_{\mathbf{x}_*}, \end{aligned} \tag{5.2}$$

where $\tilde{\mathbf{k}}_{\mathbf{x}_*}$ is the kernel vector between $\mathbf{x}_*$ and $\mathbf{X} = \{\mathbf{X}_{\mathcal{S}}, \mathbf{X}_{\mathcal{T}}\}$ using the transfer kernel $\tilde{k}(\cdot, \cdot)$ in Eq.(5.1), $\Lambda = \begin{bmatrix} \sigma_{\mathcal{S}}^2 \mathbf{I}_{n_{\mathcal{S}}} & \mathbf{0} \\ \mathbf{0} & \sigma_{\mathcal{T}}^2 \mathbf{I}_{n_{\mathcal{T}}} \end{bmatrix}$, and $\widetilde{\mathbf{K}} = \begin{bmatrix} \widetilde{\mathbf{K}}_{\mathcal{SS}} & \widetilde{\mathbf{K}}_{\mathcal{ST}} \\ \widetilde{\mathbf{K}}_{\mathcal{TS}} & \widetilde{\mathbf{K}}_{\mathcal{TT}} \end{bmatrix}$ is the overall covariance matrix. In $\widetilde{\mathbf{K}}$, $\widetilde{\mathbf{K}}_{\mathcal{SS}}$ and $\widetilde{\mathbf{K}}_{\mathcal{TT}}$ are the kernel matrices of the data in the source task and target task, respectively; $\widetilde{\mathbf{K}}_{\mathcal{ST}} (= \widetilde{\mathbf{K}}_{\mathcal{TS}}^{\intercal})$ is the kernel matrix across source and target inputs.

During the training stage, the most commonly used approach for tuning the hyper-parameters ($\boldsymbol{\theta}$) of the transfer covariance function is the conjugate gradient algorithm for optimizing the joint likelihood $p(\mathbf{y}_{\mathcal{T}}, \mathbf{y}_{\mathcal{S}}|\mathbf{X}_{\mathcal{T}}, \mathbf{X}_{\mathcal{S}}, \boldsymbol{\theta})$. Notably, training requires the inversion of covariance matrix $\widetilde{\mathbf{K}}$, which requires $\mathcal{O}((n_{\mathcal{S}} + n_{\mathcal{T}})^3)$ computations and $\mathcal{O}((n_{\mathcal{S}} + n_{\mathcal{T}})^2)$ memory. Given that $n_{\mathcal{S}} \gg n_{\mathcal{T}}$, the time and memory complexity can be written as $\mathcal{O}(n_{\mathcal{S}}^3)$ and $\mathcal{O}(n_{\mathcal{S}}^2)$, respectively. Due to the cubically scaling computational complexity and quadratically scaling storage requirements, the practical viability of TGP rapidly diminishes with increasing amount of source data accumulated over time - regardless of the potentially small size of the target dataset. Thus, the need to propose a scalable alternative over the existing TGP model is clear. From here on, the afore-described model is denoted as full TGP to avoid possible confusions.

5.2 Model Aggregation for Fast Transfer Gaussian Processes

5.2.1 Factorized Training of Transfer Gaussian Processes

To be able to train a TGP model with large-scale source inputs using limited (or distributed) computational resources, a factorized training process is deemed as an efficient strategy. In this regard, a naive approach would be to partition all the source and target inputs $\mathbf{X}$ into M subsets, and then train every local TGP model with the corresponding local subset in parallel. Larger the choice of M, lesser is the computational burden imposed on each of the local TGPs. However, note that, given the scarcity of valuable target data, there tends to be fewer and fewer target inputs in each local subset with increasing M. Since training a good local TGP expert will require the availability of a reasonable amount of target inputs, it immediately follows that a naive extension of single-task model aggregation may not suffice in the transfer learning case.

Thus, considering that the computational complexity of TGP training is primarily dominated by the large amount of source inputs (as $n_{\mathcal{S}} \gg n_{\mathcal{T}}$), partition only the source data into M spatially disjoint subsets is proposed, i.e., $\mathcal{D}_{\mathcal{S}} = \{\mathcal{D}_{\mathcal{S}_1}, \cdots, \mathcal{D}_{\mathcal{S}_M}\}$, with

$\mathcal{D}_{\mathcal{S}_i} = \{\mathbf{X}_{\mathcal{S}_i}, \mathbf{y}_{\mathcal{S}_i}\}$ for $i = 1, \cdots, M$, to effectively decrease the computational burden on each local TGP model. In addition, each local expert is provided with the *entire* target dataset - without partitioning. That is, for the ith expert $\mathcal{M}_i$, the corresponding training inputs are $\mathcal{D}_{\mathcal{S}_i}$ and $\mathcal{D}_{\mathcal{T}}$. With this, the M 'local'[1] experts $\{\mathcal{M}_i\}_{i=1}^{M}$ can be trained in parallel.

During the hyperparameter learning stage of expert $\mathcal{M}_i$, the log marginal likelihood computed with respect to $\mathcal{D}_{\mathcal{S}_i}$ and $\mathcal{D}_{\mathcal{T}}$, i.e., $\log p(\mathbf{y}_{\mathcal{T}}, \mathbf{y}_{\mathcal{S}_i} | \mathbf{X}_{\mathcal{T}}, \mathbf{X}_{\mathcal{S}_i}, \boldsymbol{\theta})$, is optimized. Specifically, the log marginal likelihood of the expert $\mathcal{M}_i$ is given by

$$\begin{aligned} &\log p\Big(\mathbf{y}_{\mathcal{S}_i}, \mathbf{y}_{\mathcal{T}} | \mathbf{X}_{\mathcal{T}}, \mathbf{X}_{\mathcal{S}_i}, \boldsymbol{\theta}\Big) \\ &\quad = -\frac{1}{2}[\mathbf{y}_{\mathcal{S}_i}^{\intercal} \quad \mathbf{y}_{\mathcal{T}}^{\intercal}](\widetilde{\mathbf{K}}_i + \Lambda_i)^{-1} \begin{bmatrix} \mathbf{y}_{\mathcal{S}_i} \\ \mathbf{y}_{\mathcal{T}} \end{bmatrix} \\ &\qquad - \frac{1}{2}\log\Big(|\widetilde{\mathbf{K}}_i + \Lambda_i|\Big) + const, \end{aligned} \tag{5.3}$$

where $\widetilde{\mathbf{K}}_i = \begin{bmatrix} \widetilde{\mathbf{K}}_{\mathcal{S}_i\mathcal{S}_i} & \widetilde{\mathbf{K}}_{\mathcal{S}_i\mathcal{T}} \\ \widetilde{\mathbf{K}}_{\mathcal{T}\mathcal{S}_i} & \widetilde{\mathbf{K}}_{\mathcal{T}\mathcal{T}} \end{bmatrix}$ and $\Lambda_i = \begin{bmatrix} \sigma_{\mathcal{S}}^2 \mathbf{I}_{n_{\mathcal{S}_i}} & \mathbf{0} \\ \mathbf{0} & \sigma_{\mathcal{T}}^2 \mathbf{I}_{n_{\mathcal{T}}} \end{bmatrix}$. This means that while training the ith expert, the observations on all the other source subsets are considered to be marginalized. A further provision made in the present work is that the learned hyperparameters of the transfer covariance function are shared across all local TGP experts as a way to prevent individual model overfitting, hence no additional hyperparameters are needed compared to the full TGP model.

5.2.2 Principled Tr-BCM for Aggregative Model Prediction

Given an unknown target inputs $\mathbf{x}_{\mathcal{T}}^q$, it is considered M (Gaussian) predictive distributions from $\{\mathcal{M}_i\}_{i=1}^{M}$ local TGP experts to be combined for the final output. The corresponding unknown response variable is defined as $f_{\mathcal{T}}^q$. Let $p(f_{\mathcal{T}}^q | \mathbf{x}_{\mathcal{T}}^q, \mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_i})$[2] be the posterior predictive probability density at the query point for expert $\mathcal{M}_i$, and the

[1] Here, the term 'local' is defined in the context of the source data.

[2] Hereafter the dependence on $\mathbf{x}_{\mathcal{T}}^q$ is omitted in the notation for simplicity.

corresponding predictive mean and variance are denoted as $\mu_i(\mathbf{x}_{\mathcal{T}}^q)$ and $\sigma_i^2(\mathbf{x}_{\mathcal{T}}^q)$, respectively. Therefore, in what follows, an efficient strategy is proposed to combine these predictive distributions in a theoretically principled manner.

The idea of the Bayesian Committee Machine (BCM) was first introduced in [130] in the context of single-task learning. The BCM is formally equivalent to an inducing-point model in which the test points are the inducing inputs [92]. It provides a principled strategy to combining local estimators that may have been trained in parallel. Inspired by the mathematical derivations of BCM, here transfer BCM (Tr-BCM) is proposed, as a principled approach to combining predictions from local TGP experts.

Let $\mathscr{D}_{\mathcal{S}_i} = \{\mathcal{D}_{\mathcal{S}_1}, \cdots, \mathcal{D}_{\mathcal{S}_i}\}$ represent the set of all source datasets with indices smaller or equal to i, with $i = 1, \cdots, M$. For the first i source subsets $\mathscr{D}_{\mathcal{S}_i}$, we have

$$\begin{aligned} p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_i}, \mathscr{D}_{\mathcal{S}_{i-1}}) \propto\ & p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}) p(\mathscr{D}_{\mathcal{S}_{i-1}}|f_{\mathcal{T}}^q, \mathcal{D}_{\mathcal{T}}) \\ & p(\mathcal{D}_{\mathcal{S}_i}|f_{\mathcal{T}}^q, \mathcal{D}_{\mathcal{T}}, \mathscr{D}_{\mathcal{S}_{i-1}}) \end{aligned} \tag{5.4}$$

Note that $p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}})$ is the posterior predictive distribution using only target training inputs, and the corresponding expert is labeled as $\mathcal{M}_{\mathcal{T}}$[1]. To simplify the calculation, we make the following conditional independence assumptions,

$$p(\mathcal{D}_{\mathcal{S}_i}|f_{\mathcal{T}}^q, \mathcal{D}_{\mathcal{T}}, \mathscr{D}_{\mathcal{S}_{i-1}}) \approx p(\mathcal{D}_{\mathcal{S}_i}|f_{\mathcal{T}}^q, \mathcal{D}_{\mathcal{T}}) \tag{5.5}$$

Iteratively applying Bayes' rule, we obtain

$$\begin{aligned} p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}}) &\approx const \times \frac{p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_M}) p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathscr{D}_{\mathcal{S}_{M-1}})}{p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}})} \\ &\approx const \times \frac{\prod_{i=1}^{M} p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_i})}{p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}})^{M-1}}. \end{aligned} \tag{5.6}$$

[1]No extra training procedure is necessary for model $\mathcal{M}_{\mathcal{T}}$, since a common target model with shared hyperparameters can be obtained after marginalizing out the corresponding source subsets for every local TGP model.

As a consequence, the predictive distribution is still a Gaussian, with mean and variance listed as follows:

$$\begin{aligned}\mu_{\text{Tr-BCM}}(\mathbf{x}_{\mathcal{T}}^q) = \sigma^2_{\text{Tr-BCM}}(\mathbf{x}_{\mathcal{T}}^q)\Bigg(&\sum_{i=1}^{M}\sigma_i^{-2}(\mathbf{x}_{\mathcal{T}}^q)\mu_i(\mathbf{x}_{\mathcal{T}}^q) \\ &+ (1-M)\sigma_{\mathcal{T}}^{-2}(\mathbf{x}_{\mathcal{T}}^q)\mu_{\mathcal{T}}(\mathbf{x}_{\mathcal{T}}^q)\Bigg), \\ \sigma^2_{\text{Tr-BCM}}(\mathbf{x}_*) = 1/&\Bigg(\sum_{i=1}^{M}\sigma_i^{-2}(\mathbf{x}_{\mathcal{T}}^q) + (1-M)\sigma_{\mathcal{T}}^{-2}(\mathbf{x}_{\mathcal{T}}^q)\Bigg),\end{aligned} \tag{5.7}$$

where $\mu_{\mathcal{T}}$ and $\sigma^2_{\mathcal{T}}$ are the predicted mean and variance of the expert $\mathcal{M}_{\mathcal{T}}$. From the predictive distribution, it is easy to observe that the overall weight assigned to the expert $\mathcal{M}_i$ in the predictive mean is inversely proportional to its variance. This implies that those experts with more confident prediction are automatically assigned higher responsibility in an input dependent manner.

Observe that unlike single-task BCM, where the "correction" term for the predictive variance is the prior variance $k_{**} = k(\mathbf{x}_{\mathcal{T}}^q, \mathbf{x}_{\mathcal{T}}^q)$ [132], the posterior $\sigma^2_{\mathcal{T}}$ is used to rectify the predictive variance of Tr-BCM. This is caused by the fact that in Tr-BCM, the whole target data is fully utilized across every local expert, guaranteeing quality predictions from each local expert.

5.2.3 Alternative Heuristic Model Aggregations

Apart from the principled Tr-BCM, it is possible to construct alternative model aggregation schemes based on (related) heuristically defined procedures that have recently been developed for large-scale single-task GPs. A prominent example among them is the *product of experts* (PoE) [131]. In the PoE, all the predictive distributions at $\mathbf{x}_{\mathcal{T}}^q$ from a set of local estimators are directly multiplied, and the product is proportional to a Gaussian distribution if every local predictive distribution is a Gaussian. Similarly, in the case of transfer learning, the final output distribution can be directly set as proportional to the product of all the predictive distributions from the M local TGP experts, with the resultant mean and variance listed as follows:

$$\mu_{\text{PoE}}(\mathbf{x}_{\mathcal{T}}^q) = \sigma_{\text{PoE}}^2 \sum_{i=1}^{M} \beta_i \sigma_i^{-2}(\mathbf{x}_{\mathcal{T}}^q)\mu_i(\mathbf{x}_*),$$
$$\sigma_{\text{PoE}}^2(\mathbf{x}_{\mathcal{T}}^q) = 1/\left(\sum_{i=1}^{M} \beta_i \sigma_i^{-2}(\mathbf{x}_{\mathcal{T}}^q)\right), \tag{5.8}$$

where the heuristically incorporated tunable parameter β_i is set to 1 for $i = 1, \cdots, M$.

From Eq.(5.8), observe the familiar property that experts which are uncertain about their predictions are automatically weighted less than those which are more confident about their predictions. However, with an increasing number of TGP experts, Eq.(5.8) implies that the predictive variance $\sigma_{\text{PoE}}^2(\mathbf{x}_{\mathcal{T}}^q)$ monotonically decreases, leading to unreasonably overconfident predictions. Therefore, the PoE model is inconsistent in the sense that it does not fall back to the prior outside the regime of the training dataset [132]. To overcome the evident issue of the PoE aggregation approach, it has been proposed in the literature to simply set $\sum_{i=1}^{M} \beta_i = 1$. The corresponding model is labeled as *generalized PoE* (gPoE). Accordingly, throughout this chapter, we set $\beta_i = 1/M$, so that the predictive means of PoE and gPoE are identical, and only the predictive variances are adjusted.

5.2.4 Empirical Analysis of Various Aggregation Models

We analyze and compare the Tr-BCM, PoE, and gPoE methods using a 1-D toy example. In this toy example, a single source and target inputs are sampled according to a full TGP model with source-target similarity $\lambda = 0.5$ and squared exponential kernel with pre-specified hyperparameters. There are a large number (1,000) of source inputs sampled uniformly at random from the range $[-1, 1]$, and 5 target inputs sampled from the range of $[0, 1]$. The source training inputs are partitioned into two disjoint subsets. Thus, the predictive distributions of the two local TGPs, $\mathcal{M}_1$ and $\mathcal{M}_2$ are displayed in 5.1(a) and 5.1(b). The resultant predictions from the three different model aggregation schemes are presented in 5.1(c-e), with each compared against the predictions

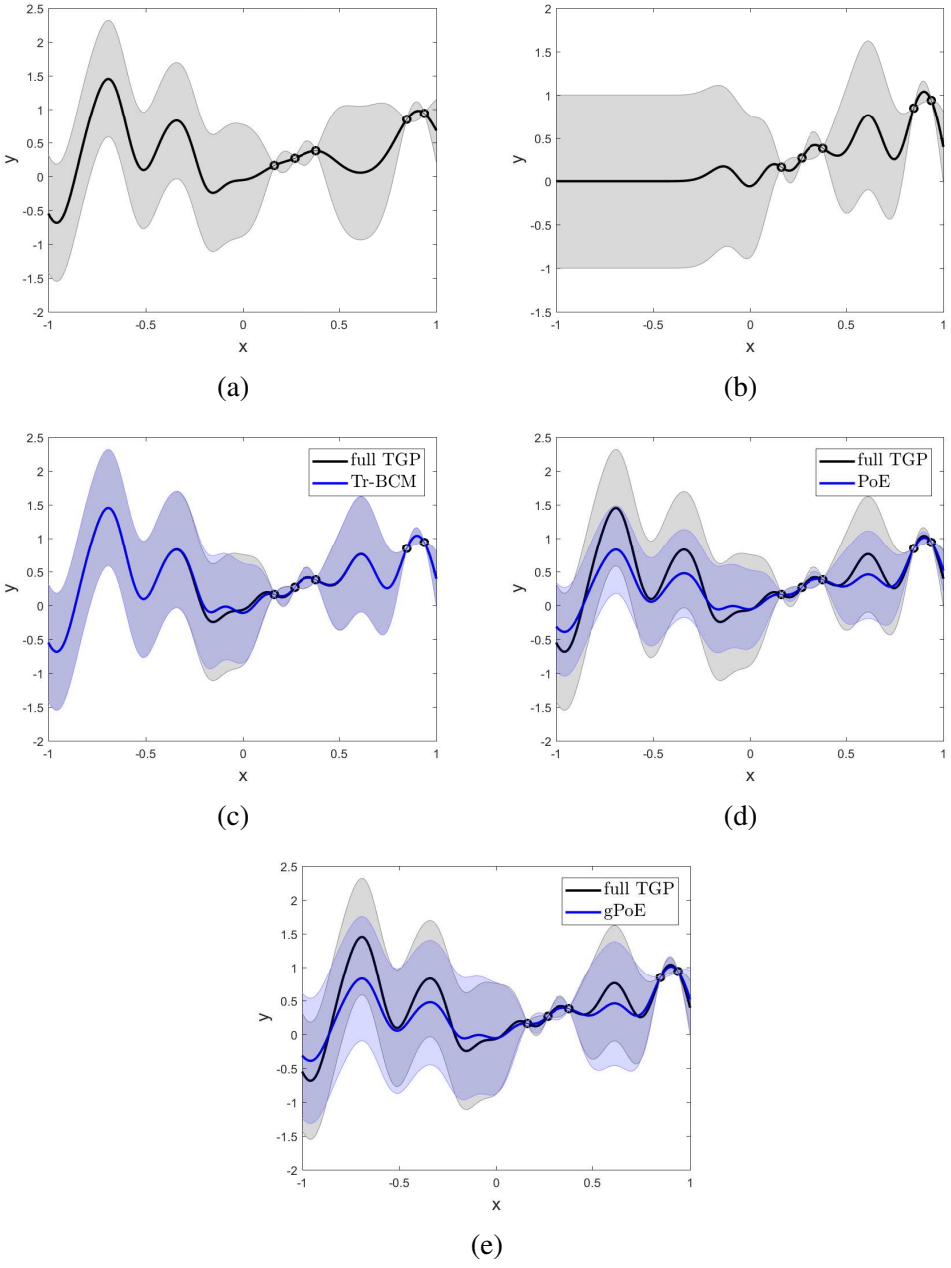


FIGURE 5.1: Toy example of aggregation of two local TGP experts. In (a) and (b), for each model, we present the predictive mean (black curve) while the gray shaded region denotes the standard deviation. The "○" symbols represent the 5 target training samples. **c-e** aggregated predictions (in blue) from Tr-BCM, PoE, and gPoE compared against the full TGP model prediction (in gray black).

made by the full TGP model. The goal is to test how closely the proposed lightweight aggregation schemes can replicate the full TGP model.

Clearly, the predictive performance of Tr-BCM as shown in Fig. 5.1(c) is the best approximation of the full TGP model among all the aggregation models. For the PoE in

Fig. 5.1(d), the problematic overconfident prediction (with unreasonably low variance) is verified. What is more, the predictive mean of PoE does not align well with the full TGP either. In contrast to PoE, gPoE seems to make more consistent predictions of the predictive variance, as displayed in Fig. 5.1(e).

5.2.5 Theoretical Analysis of Various Aggregation Models

We further analyze and compare the theoretical behavior of the proposed Tr-BCM against PoE and gPoE. To simplify the analysis, it is assumed that stationary and monotonic kernel is applied. All source subsets are spatially disjoint, such that $k(\mathbf{x}, \mathbf{x}') \approx 0$, for $\mathbf{x} \in \mathcal{D}_{\mathcal{S}_i}$ and $\mathbf{x}' \in \mathcal{D}_{\mathcal{S}_j}$, when $i \neq j$. In the following, the predictive behavior on an unknown query points $\mathbf{x}_{\mathcal{T}}^q$ will be analyzed under two circumstances. One case is that $\mathbf{x}_{\mathcal{T}}^q$ is distant from all the source subsets. The other case is that $\mathbf{x}_{\mathcal{T}}^q$ falls within the regime of a specific source subset. The proposed Tr-BCM model is proved to output consistent predictive distributions with the ones made by the full TGP model.

In the first case, $\mathbf{x}_{\mathcal{T}}^q$ is distant from all source subsets, meaning that $k(\mathbf{x}_{\mathcal{T}}^q, \mathbf{x}) \approx 0$, for all $\mathbf{x} \in \mathcal{D}_{\mathcal{S}_i}$, $i = 1, \cdots, M$. Thus, the predictive distributions of all the local TGP experts $p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_i})$ and the full TGP model $p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}})$ fall back to the prediction of $\mathcal{M}_{\mathcal{T}}$, i.e.,

$$\begin{aligned} p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}}) &\approx p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_i}) \approx p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}) \\ &= \mathcal{N}(\mu_{\mathcal{T}}(\mathbf{x}_{\mathcal{T}}^q), \sigma_{\mathcal{T}}^2(\mathbf{x}_{\mathcal{T}}^q)),\ i = 1, \cdots, M. \end{aligned} \tag{5.9}$$

According to Eq.(5.7) and Eq.(5.8), Tr-BCM and gPoE can produce the same predictive distributions as the one made by the full TGP model, while PoE makes unreasonably overconfident predictions as $\lim_{M\to\infty} \sigma_{\text{PoE}}^2(\mathbf{x}_{\mathcal{T}}^q) = 0$.

In the second circumstance, $\mathbf{x}_{\mathcal{T}}^q$ falls within the regime of a specific source subset - say the ith expert. Therefore, we have:

$$\begin{aligned} p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}}) &\approx p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_i}) = \mathcal{N}(\mu_i(\mathbf{x}_{\mathcal{T}}^q), \sigma_i^2(\mathbf{x}_{\mathcal{T}}^q)), \\ p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}, \mathcal{D}_{\mathcal{S}_j}) &\approx p(f_{\mathcal{T}}^q|\mathcal{D}_{\mathcal{T}}) = \mathcal{N}(\mu_{\mathcal{T}}(\mathbf{x}_{\mathcal{T}}^q), \sigma_{\mathcal{T}}^2(\mathbf{x}_{\mathcal{T}}^q)),\ \ j \neq i. \end{aligned} \tag{5.10}$$

According to Eq.(5.7), the aggregated predictive distributions of Tr-BCM are equivalent to the ones made by the full TGP model. The PoE aggregation scheme continues to make characteristic overconfident predictions. For the gPoE, from Eq.(5.8) - with $\beta_i = 1/M$ - it is found that with increasing number of experts, the predictive variance can be written as $\lim_{M\to\infty} \sigma^2_{\text{gPoE}}(\mathbf{x}^q_{\mathcal{T}}) = \sigma^2_{\mathcal{T}}(\mathbf{x}^q_{\mathcal{T}})$. In addition, according to Proposition 1 in [75], given the availability of related source data ($|\lambda| > 0$), we have $\sigma^2_i(\mathbf{x}^q_{\mathcal{T}}) < \sigma^2_T(\mathbf{x}^q_{\mathcal{T}})$. These facts imply that since the aggregated predictive distribution of gPoE falls back to the prediction of single-task expert $\mathcal{M}_{\mathcal{T}}$, the predictive behavior of gPoE theoretically tends to be over conservative compared to full TGP.

In summary, the theoretical behavior of the proposed model, namely Tr-BCM, is consistent with the full TGP model. In contrast, the heuristically defined PoE and gPoE aggregation schemes tend to make overconfident and over-conservative predictions, respectively.

5.2.6 Computational Complexity and Memory Consumption

As elaborated earlier, the $\mathcal{O}(n^3_{\mathcal{S}})$ computational complexity and $\mathcal{O}(n^2_{\mathcal{S}})$ storage requirement are bottlenecks to scale a full TGP model to tackle problems with large-scale sources. To overcome this issue, we have put forward a theoretically principled Tr-BCM model that partitions the source data into disjoint subsets so as to build lightweight local TGP experts. Here, we analyze the complexity of the Tr-BCM approach. We refer to Eq.(5.3) which points to the inversion of matrix $(\widetilde{\mathbf{K}}_i + \Lambda_i)$, for $i = 1, \ldots, M$. Expert $\mathcal{M}_i$ will require $\mathcal{O}((n_{\mathcal{S}_i} + n_{\mathcal{T}})^3)$ computations and $\mathcal{O}((n_{\mathcal{S}_i} + n_{\mathcal{T}})^2)$ memory space. Therefore, the overall factorized training process requires $\mathcal{O}(M \times (n_{\mathcal{S}}/M + n_{\mathcal{T}})^3)$ computations and $\mathcal{O}(M \times (n_{\mathcal{S}}/M + n_{\mathcal{T}})^2)$ memory, assuming uniform source data partitioning. In the present work, we set $M \approx n_{\mathcal{S}}/(2n_{\mathcal{T}})$ in the experimental study to enable sufficient source-target knowledge transfer. Accordingly the overall training complexity decreases to $\mathcal{O}(\frac{27}{2} n_{\mathcal{S}} n^2_{\mathcal{T}})$, and the memory cost reduces to $\mathcal{O}(\frac{9}{2} n_{\mathcal{S}} n_{\mathcal{T}})$. In other words, both quantities scale linearly with the number of source observations; thereby making Tr-BCM a viable option for lightweight online on-mote processing on edge devices.

5.3 Enhanced Expressiveness with Tr-BCM: Local Inter-Task Similarity Capture

Recent progress towards adaptive multi-task/transfer GP has shown that the expressiveness of a model can be enhanced by exploiting *spatially adaptive* inter-task relationship [135, 136]. However, most existing approaches for adaptive multi-task/transfer learning have been focused on *fixed* correlations among output variables. In other words, it has been assumed that the source-target relationship can be captured by a single scalar parameter, and is uniform everywhere in the input space. The same is seen to be true for the full TGP model, where a single parameter (λ) is used to capture source-target similarity. However, this assumption is often found to be too strict for real-world applications. Therefore, in this section, the possibility of equipping traditional transfer learning with the ability to learn non-uniform inter-task relationship is explored through a simple adjustment to Tr-BCM.

Taking advantage of the partition of source dataset into M disjoint subsets, a straightforward approach to capture localized inter-task similarity would be to apply the following localized transfer covariance function:

$$\tilde{k}(\mathbf{x},\mathbf{x}') = \begin{cases} \lambda_i k(\mathbf{x},\mathbf{x}'), & \mathbf{x}\in\mathbf{X}_{\mathcal{S}_i} \;\&\; \mathbf{x}'\in\mathbf{X}_{\mathcal{T}} \\ & \text{or } \mathbf{x}\in\mathbf{X}_{\mathcal{T}} \;\&\; \mathbf{x}'\in\mathbf{X}_{\mathcal{S}_i} \\ k(\mathbf{x},\mathbf{x}'), & \text{otherwise,} \end{cases} \tag{5.11}$$

where λ_i indicates the localized inter-task similarity between the ith source subset $\mathcal{S}_i$ and the target task $\mathcal{T}$. Using this transfer covariance function, localized inter-task relationship is learned between the different source subsets and the target data.

Let $K^f \in \mathbb{R}^{(M+1)\times(M+1)}$ represent a matrix capturing the inter-task (between source and target) and intra-task (between different source subsets) similarities. Naturally, the similarity across data subsets belonging to the same source task can be assumed to be

1. Accordingly, K^f is expressed in the following form:

$$K^f = \begin{pmatrix} 1 & 1 & \cdots & 1 & \lambda_1 \\ 1 & 1 & \cdots & 1 & \lambda_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & \lambda_M \\ \lambda_1 & \lambda_2 & \cdots & \lambda_M & 1 \end{pmatrix}. \tag{5.12}$$

In order to guarantee that the localized transfer covariance function in Eq.(5.11) is always PSD given a valid kernel $k(\cdot,\cdot)$, it suffices for us to show that K^f is a PSD matrix [74]. The following theorem gives the necessary and sufficient condition for a PSD K^f.

Theorem 5.1. *The matrix K^f is PSD if and only if $\lambda_1 = \lambda_2 = \cdots = \lambda_M$, and $|\lambda_i| \leq 1$, for all $i = 1, 2, \cdots, M$.*

Proof. Necessary condition: A principal minor of any matrix A is defined as the determinant of a principal submatrix of matrix A. Let A be an symmetric matrix. Then A is PSD if and only if every principal minor of A is nonnegative [139]. Therefore, given K^f is PSD, for a 2×2 principal submatrix $K_i^f = \begin{pmatrix} 1 & \lambda_i \\ \lambda_i & 1 \end{pmatrix}$, we have $|K_i^f| = 1 - \lambda_i^2 \geq 0$. Therefore, $|\lambda_i| \leq 1$, for $i = 1, \cdots, M$.

Further, for a 3×3 principal submatrix $K_{ij}^f = \begin{pmatrix} 1 & 1 & \lambda_i \\ 1 & 1 & \lambda_j \\ \lambda_i & \lambda_j & 1 \end{pmatrix}$, we have $|K_{ij}^f| = -(\lambda_i - \lambda_j)^2 \geq 0$. Thus, we have $\lambda_i = \lambda_j$.

Sufficiency condition: Let $\lambda_1 = \lambda_2 = \cdots = \lambda_M = \lambda$ and $|\lambda| \leq 1$. According to Theorem 1 in Cao et al. [81], it follows that K^f is PSD since a matrix of all ones is PSD. □

According to the above theorem, all the local source-target similarities take the same value in order to guarantee the validity of the transfer covariance function of Eq.(5.11). However, such a condition hampers the original intention of partitioning the source data

into local subsets to learn localized inter-task relationship between each source subset and the target task. Thus, training a full TGP with the localized transfer kernel is not guaranteed to be feasible.

In contrast, Tr-BCM can easily avoid this issue by neglecting the correlations between different source subsets, as only the correlations between the individual source subsets and the target task are considered separately under a conditional independence assumption. Taking this cue, the provision for sharing a single set of hyperparameters across all local TGP experts is relaxed in Tr-BCM, and allow localized λ_i's for each local model to be learned. All other hyperparameters of the covariance function continue to be shared. After the factorized training stage, if we marginalize out the source subsets, a common target expert $\mathcal{M}_{\mathcal{T}}$ will still be obtained. Therefore, during prediction, Eq.(5.7) for Tr-BCM can be directly applied.

It is observed that if the ith local expert learns a high source-target correlation, i.e., $|\lambda_i| \to 1$, then predictions within its local region will be highly supported by the local subset of the source data. On the contrary, if λ_i is learned to be close to 0, then there is little knowledge transferred from the corresponding source subset to the target task. As there is no restriction placed on the λ_i's to be uniform across the M subsets, the non-uniformity of the source-target similarity distribution is practically addressed.

To provide insights on the behavior of the *Tr-BCM model with localized inter-task similarity capture* (labeled as Tr-BCM-ls), we consider a toy example. The generation of the synthetic dataset is carried out as follows. 100 data points are randomly sampled from each of the two 1-D functions $f_{\mathcal{S}} = \sin(|x|)$ and $f_{\mathcal{T}} = \sin(x)$, $-5 \leq x \leq 5$, both corrupted by a zero-mean Gaussian noise with variance equal to 0.1. The first function is taken as source task, and the second function is taken as target task. 5% of the target data points are used for training, and the rest are used for testing. After training the conventional full TGP model, it is obtained that the source-task similarity is $\lambda \approx 0$, implying that the source and target tasks are nearly uncorrelated globally. As a result, the performance of TGP is somewhat similar to that of single-task GP, showing that there is nearly no knowledge transfer from source task to target task.

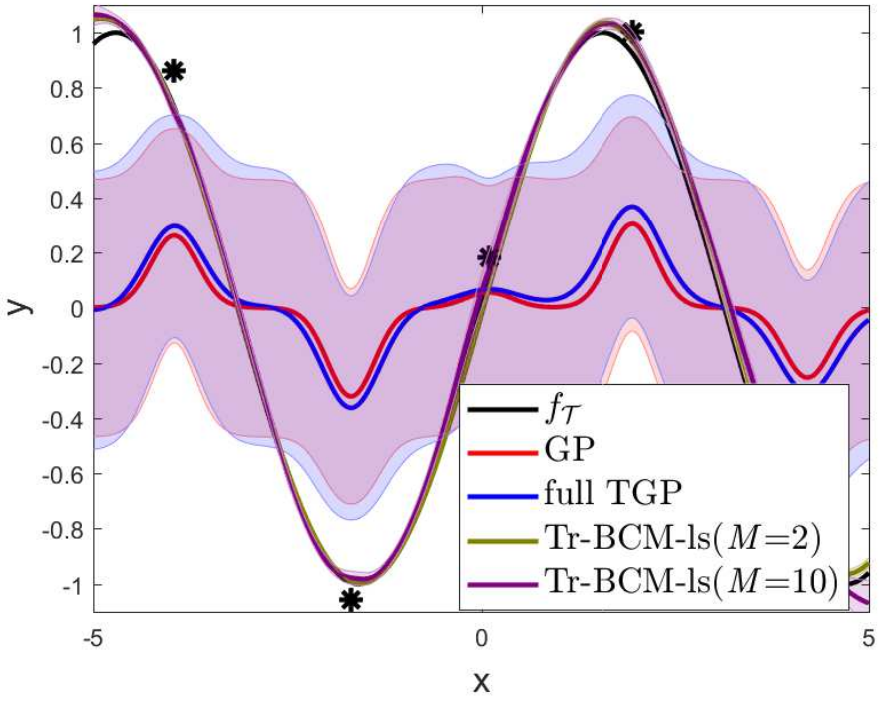


FIGURE 5.2: Predictive distribution of GP, full TGP and the proposed Tr-BCM. Shaded area denotes the predicted standard derivation of the corresponding probabilistic output. The starred points are the training data of the target task.

Nevertheless, it is apparent from the function forms of $f_{\mathcal{S}}$ and $f_{\mathcal{T}}$ that there naturally exist two regions in the input space where the source and target tasks are indeed correlated. In particular, the task pairs are strongly *positively* correlated when $x >= 0$, and strongly *negatively* correlated when $x < 0$. Accordingly, the source data is partitioned into 2 subsets, i.e., $\mathcal{D}_{\mathcal{S}_1} = \{x_i >= 0 : x_i \in \mathcal{D}_{\mathcal{S}}\}$ and $\mathcal{D}_{\mathcal{S}_2} = \{x_i < 0 : x_i \in \mathcal{D}_{\mathcal{S}}\}$. By applying the proposed factorized training with localized inter-task similarity capture, it is found that two different source-target similarities (-0.99 and 1.00) are indeed learned, closely matching the true underlying distribution of inter-task similarity. The predicted distribution of Tr-BCM-ls is shown in Fig. 5.2. Using only 5% of the target data, the proposed method can almost exactly recover the target function $f_{\mathcal{T}}$ by adaptively taking advantage of the knowledge concealed in the source task, highlighting the efficacy of the proposed method.

In order to quantify the averaged generalization performance on the test set over 10

TABLE 5.1: Averaged RMSE on toy example

Methods	RMSE
GP	0.42623 ± 0.0084
full TGP	0.45456 ± 0.0004
Tr-BCM-ls ($M = 2$)	$\mathbf{0.1050} \pm 0.0040$
Tr-BCM-ls ($M = 10$)	0.12681 ± 0.0176

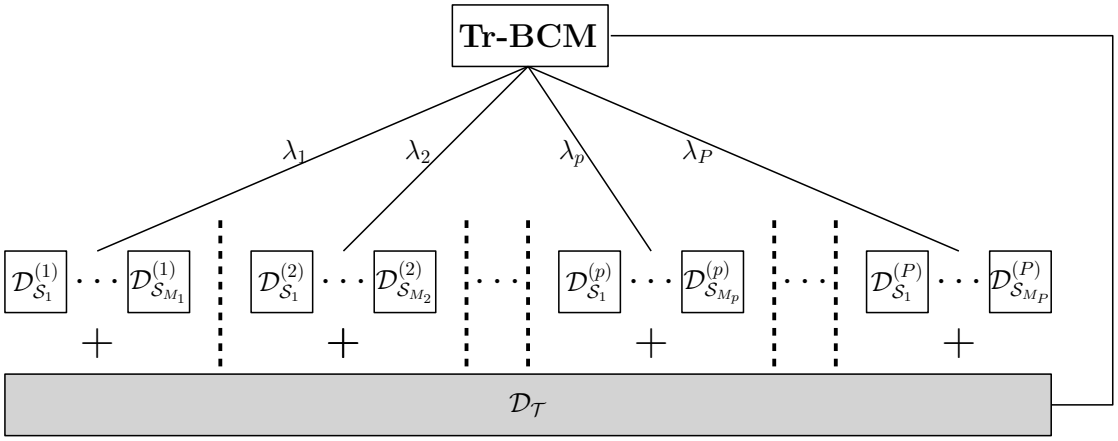


FIGURE 5.3: Hierarchical structure of multi-source Tr-BCM.

trial runs, we present root mean square error (RMSE[1]) results in Table 5.1. The results in Table 5.1 also contain the case of Tr-BCM-ls with $M = n_{\mathcal{S}}/2n_{\mathcal{S}} = 10$ (k-means is used to partition the source dataset). Tr-BCM-ls with $M = 10$ outperforms full TGP and GP with a large margin. However, Tr-BCM-ls with $M = 2$ slightly outperforms Tr-BCM-ls with $M = 10$, as in the former case prior knowledge about the underlying distribution of source-target correlation was utilized while partitioning the source datasets.

5.4 Extensions of Tr-BCM to Multi-Source Transfer Learning

Given the proposed relaxation of Tr-BCM with localized inter-task relationship capture, a similar scheme can be immediately used to deal with multi-source transfer learning problems as well.

[1]RMSE is computed as $\sqrt{\frac{\sum_{i=1}^{n_{\text{test}}}(y_i-\mu(\mathbf{x}_i))^2}{n_{\text{test}}}}$ on test samples, where y_i is the label of $\mathbf{x}_i$, and $\mu(\mathbf{x}_i)$ is the predicted mean of $\mathbf{x}_i$ on target task $\mathcal{T}$.

With research efforts largely confined to the single-source setting [2, 81], an increasing amount of studies are contributing to a realistic applicability of transfer learning by addressing the multi-source scenario - where different sources have differing degree of inter-task relationship with the target [24]. By ignoring the interactions among the different source tasks, the relaxed Tr-BCM formulation can be directly applied to tackle multi-source transfer learning problems. Accordingly, in the following, a hierarchical structure of Tr-BCM is proposed to tackle these kinds of problems.

Say there are P source tasks $\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(P)}$ and one target task $\mathcal{T}$. All the tasks are assumed to be defined in a common input space with dimensionality d. For the pth source task, the corresponding training data is labeled as $\mathcal{D}_{\mathcal{S}}^{(p)} = \{\mathbf{X}_{\mathcal{S}}^{(p)}, \mathbf{y}_{\mathcal{S}}^{(p)}\}$, where $\mathbf{X}_{\mathcal{S}}^{(p)} \in \mathbb{R}^{n_{\mathcal{S}}^{(p)} \times d}$ and $\mathbf{y}_{\mathcal{S}}^{(p)} \in \mathbb{R}^{n_{\mathcal{S}}^{(p)}}$. To accelerate the computational process, $\mathcal{D}_{\mathcal{S}}^{(p)}$ is partitioned into $M_p = n_{\mathcal{S}}^{(p)}/2n_{\mathcal{T}}$ local blocks. Hence, $M = \sum_{p=1}^{P} M_p$ TGP experts undergo factorized training in parallel. Notice that each source task possesses a unique noise level and source-target similarity, while all the other hyperparameters are shared across all the experts. Fig. 5.3 shows the hierarchical structure of the proposed model. The aggregated predictive distributions are directly calculated using Eq.(5.7).

5.5 Experimental Study

5.5.1 Medium-scale Datasets

In the following, experiments on two UCI datasets are conducted with a single source task with medium-sized source training inputs. In addition to the proposed Tr-BCM and the direct extension of (g)PoE, we present results obtained from standard GP, the full TGP model, implementations of Transfer Stacking GP (TSGP) [32], and TrAdaBoost.R2 [23] for regression transfer. For all the aggregation models, the number of experts is set as $M = n_{\mathcal{S}}/(2n_{\mathcal{T}})$, and k-means is used to partition the source data. In the following, only predictive mean is used to measure the generalization performance, therefore, PoE and gPoE serve as referred as one model.

TABLE 5.2: Results on the real-world datasets. The averaged RMSEs of different approaches for Wine and error distance (in meter) for UJIIndoorLoc are reported. Superior performance are highlighted using bold characters.

Methods	Wine	UJIIndoorLoc
Tr-BCM	**0.7074**±0.0022	**6.8197**±0.0711
(g)PoE	0.7562±0.0008	7.4277±0.0111
full TGP	0.7739±0.0207	6.9279±0.0164
GP	0.7619±0.0019	7.6545±0.0001
TSGP	0.7675±0.0056	7.5631±0.0001
TrAdaBoost.R2	0.8053±0.0111	39.448±0.0003

5.5.1.1 Wine Quality Dataset

The wine dataset [140] is related to red and white wine samples, and the goal is to model wine quality based on physicochemical tests including PH values, etc (in total 11 features). The labels are given by experts with grades between 0 (very bad) and 10 (very good). There are in total 4898 records, among which 1599 are for the red wine, and 4898 are for the white wine. In the experimental study, the quality prediction for the white wine is used as the source task, and the quality prediction for red wine is taken as the target task. 5% of the available target data is used for training, and the remaining is used for evaluation.

5.5.1.2 WiFi-based Indoor Localization

The WiFi-based indoor localization system aims to detect the location of a client device given the signals received from various access points. Given the ever-expanding scale of WiFi deployments in metropolitan areas, WiFi-based localization gains its importance and popularity due to the many AI and ubiquitous computing applications. However, most localization techniques require a training set of signal strength readings labeled against a ground truth location map. Training data of the target task is precious due to the heavy reliance on the ground truth calibration. Therefore, transfer learning becomes more appealing as fruitful knowledge from some source data can be utilized to decrease the workload of calibrating the target data. In the experimental study, we randomly choose two floors in one building as source and target task. Therefore, there are 1137 source inputs, 78 target training inputs and 1486 test samples.

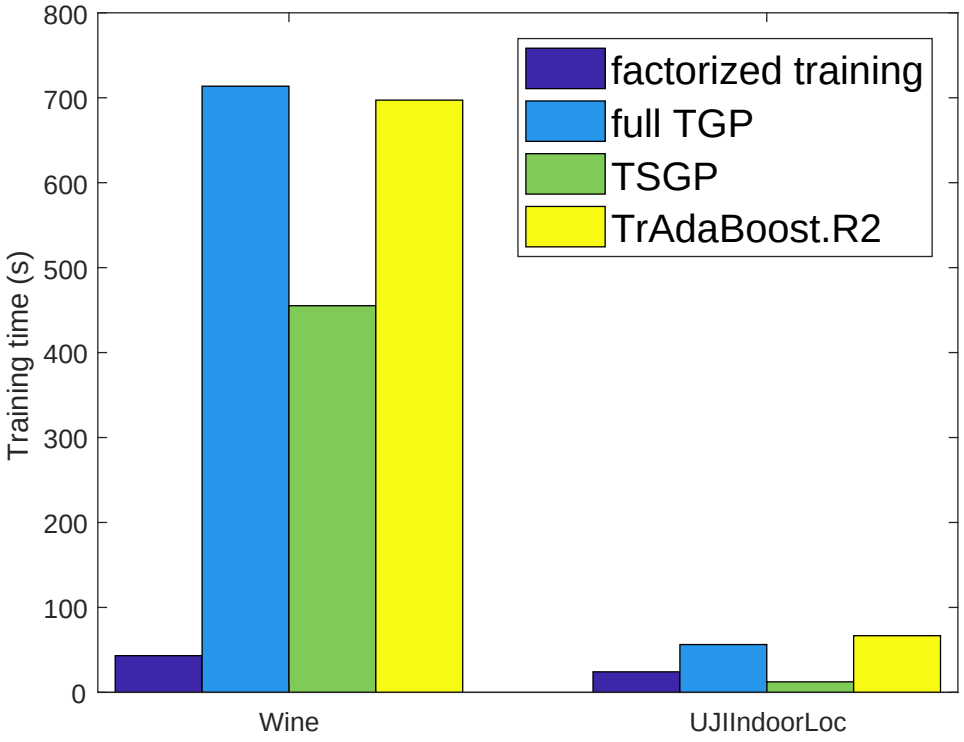


FIGURE 5.4: The averaged training time for each method.

All the experiments are conducted over 10 repetitions. The results are presented in Table 5.2. Note that the proposed Tr-BCM performs the best over the other compared methods. What is more, single-task GP outperforms full TGP. The reason is probably that optimizing the joint likelihood over source and target inputs may bias the TGP model towards the source task since $n_{\mathcal{T}} \ll n_{\mathcal{S}}$. On the other hand, in the proposed factorized training scheme, the training data for each expert is more balanced since within each local expert, the number of source inputs is limited to be twice that of target inputs. TrAdaBoost.R2 is always found to perform the worst over all the methods. This is consistent with the experimental results reported in [24].

Further, the training time of all the transfer learning methods is recorded, which are reported in Fig. 5.4. Note that there are more source inputs for Wine data than in UJIIndoorLoc. As a result, the proposed factorized training for transfer learning shows its advantages with larger source inputs. Comparing the number of source inputs for the two datasets, the runtime for the proposed factorized training does not increase

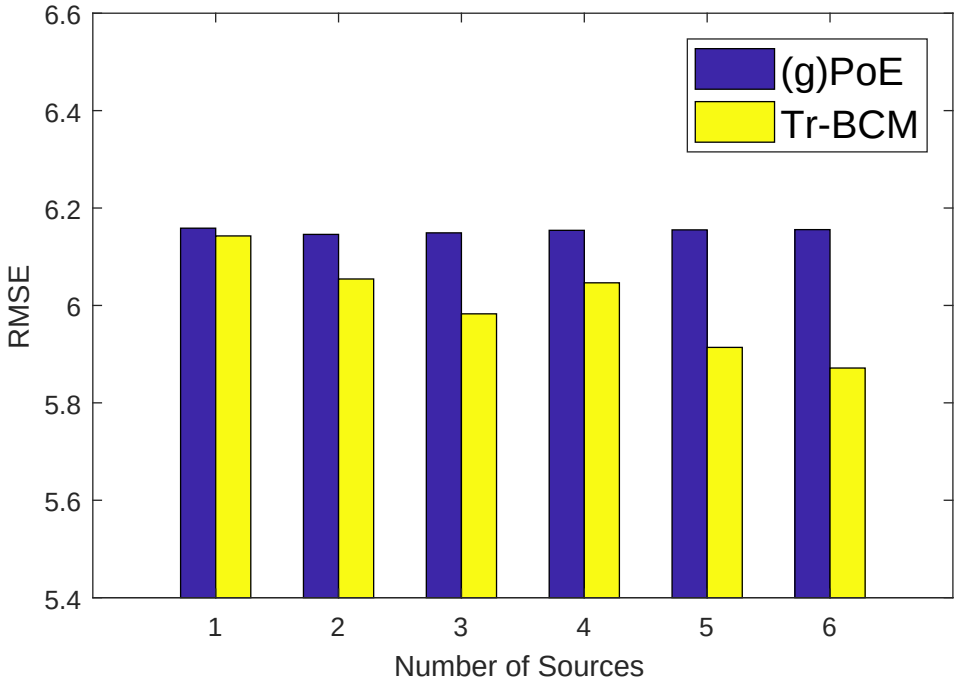


FIGURE 5.5: The averaged RMSE over 10 runs for (g)PoE, and Tr-BCM with increasing number of source tasks.

drastically with the increased number of source inputs (scales linearly), while training time for other methods increases drastically (scales cubically).

5.5.2 Large-scale Dataset

The SARCOS dataset [81] relates to an inverse dynamics problem for a seven degrees-of-freedom anthropomorphic robot arm. The task is to map from a 21-dimensional input space (7 joint position, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. Therefore, the input has 21 dimensions and there are 7 tasks for each input. The original problem is of multi-output regression. In this experiment, the first

TABLE 5.3: Results on the large-scale datasets. The RMSEs of different approaches are reported for SARCOS. Superior performance are highlighted using bold characters.

Methods	RMSE
Tr-BCM	**5.8715**$\pm$0.9077
(g)PoE	6.1557$\pm$1.1485
GP	14.8993$\pm$5.7632

joint torque is used as the target task, and the remaining six joint torques as six different source tasks. 5% of the available target data is used for training, and the remaining is used for evaluation. For the source tasks, 30,000 points are randomly choses in total. With huge amount of source inputs, most traditional GP-based methods become impractical. As we compare the aggregation models Tr-BCM and gPoE to a standard single-task GP, a huge performance enhancement is observed as shown in Table 5.3. Notably, Tr-BCM outperforms (g)PoE.

We have also analyzed the effect of increasing number of sources. Using factorized training, we first jointly train the six source tasks and target task. During prediction, different cases are considered in which the number of sources is gradually increased. The predictive performance of Tr-BCM and (g)PoE is shown in Fig. 5.5, averaged over 10 repetitions. With increasing number of source tasks, Tr-BCM is found to significantly improve its performance with the availability of more source inputs, while the performance enhancement for (g)PoE is only marginal.

5.6 Gaussian Process-Assisted Multi-Source Transfer Optimization

In Bayesian optimization, it is assumed that the function $f(\mathbf{x})$ is drawn from a Gaussian process prior. The queries to the function are denoted as $D = \{\mathbf{X}, \mathbf{y}\}$. The *acquisition function* (or *infill sampling criteria*) is denoted as $a : \mathcal{X} \rightarrow \mathbb{R}$ (typically the search space $\mathcal{X}$ is assumed to be a compact set of $\mathbb{R}^d$, where d is the dimension of the problem to be optimized). After evaluating the objective according to an initial space-filling experimental design, an iterative procedure, as shown in Algorithm 1, is applied. Deciding where to query f next is tradeoff between exploitation and exploration. There are several acquisition functions proposed in the literature to balance such tradeoff, including Probability of improvement, expected improvement, and GP Lower (Upper) Confidence Bound (GP-LCB).

To find the point which maximizes the acquisition function, we need to optimize $a(\mathbf{x})$ within search space A. DIRECT [141], NES [111], genetic algorithms [65, 66], are often deployed to optimize these cheap a_{LCB}.

It is straightforward to apply a Gaussian process regression model to the framework of transfer Bayesian optimization. Unlike multi-task Bayesian optimization [13], in which it is necessary to develop an acquisition function to dynamically allocate computational resources over multiple tasks, we only need to consider the target optimization problem. Thus, standard single-task acquisition function can be directly applied in the framework of surrogate-assisted transfer optimization. Specifically, GP-LCB proposed in [71] is adopted, which is proved to have exponentially vanishing regret. The acquisition function of GP-LCB is defined as follows:

$$a_{\text{LCB}}(\mathbf{x}; \mathcal{D}_{t-1}, \boldsymbol{\theta}) = \mu(\mathbf{x}|\mathcal{D}_{t-1}, \boldsymbol{\theta}) - \beta_t^{1/2}\sigma(\mathbf{x}|D_{t-1}, \boldsymbol{\theta}), \tag{5.13}$$

where $\beta_t = 2\log(dt^2\pi^2/6\delta)$, and $\delta \in (0, 1)$ (we set $\delta = 0.1$ throughout experimental study).

When extending the proposed Tr-BCM to Bayesian optimization, $\mu(\cdot)$ and $\sigma(\cdot)$ can be directly replaced with $\mu_{\text{Tr-BCM}}(\cdot)$ and $\sigma_{\text{Tr-BCM}}(\cdot)$, with the help from source optimization tasks.

In experimental study, three commonly used benchmark functions, Ackley, Griewank and Rastrigin, defined over a bounded set on $\mathbb{R}^2$, are optimized. When optimizing one of the three functions, the other functions serve as the source optimization tasks. The total number of function evaluations allowed is set to 30, and a large number (1,000) of observations on source optimization tasks are provided, therefore, full transfer GP assisted transfer optimization will be computationally expensive. The numerical results achieved at the end of 30 function evaluations averaged over 10 independent runs are shown in Table 5.4. Notably, multi-source transfer optimization with the proposed Tr-BCM as surrogate model outperforms Bayesian optimization with no knowledge transfer in all the three test problems.

TABLE 5.4: Averaged optimal value obtained by Bayesian optimization and transfer Bayesian optimization with the proposed surrogate model over 10 independent runs. Values in brackets indicate standard deviations.

Problem	BO	Transfer BO
Ackley	4.6180 (2.8098)	**4.1589** (2.8006)
Griewank	0.1603 (0.0703)	**0.1285** (0.0769)
Rastrigin	12.4621 (2.5201)	**9.1903** (3.4478)

5.7 Conclusion

In this chapter, we have introduced a theoretically principled aggregation model, namely transfer Bayesian Committee Machine (Tr-BCM), for transfer learning with large-scale source inputs. The salient features of Tr-BCM are three-fold: (1) it offers a distributed lightweight alternative that is capable of replicating the full (heavyweight) TGP model; (2) by relaxing the uniformity condition on inter-task similarity capture, Tr-BCM can even enhance model expressiveness compared to TGP; (3) the relaxed Tr-BCM formulation directly applies to the multi-source transfer learning scenario where different sources can have differing inter-task relationship with the target.

The proposed aggregation model has been applied to synthetic as well as real-world datasets, with the experimental results verifying its efficacy over existing state-of-art transfer learning methods. Compared to traditional transfer learning methods, the accuracy and scalability of Tr-BCM are both theoretically and empirically shown to be superior with increasing amounts of source data, i.e., Wine, UJIIndoorLoc, and SARCOS. The proposed Tr-BCM is applied in the framework of multi-source surrogate-assisted transfer optimization. The scalability of Tr-BCM as well as the tractable inference increase the practical applicability of transfer optimization with multiple sources.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, the motivation to conduct knowledge transfer in the area of optimization is highlighted in Chapter 1, and the notion of *transfer optimization* is formally defined. While transfer learning is already well established in the realm of machine learning, it is found that similar efforts of automatic knowledge transfer during optimization exercises have been found to be rare. In an attempt to incorporate human-like capabilities into optimization solvers, several methods in multi-source data-driven transfer optimization are proposed, in order to utilize past problem-solving experiences to enhance the optimization process for the new task of practical interest. In this chapter, the contributions of the present thesis, as well as several potential future directions, are summarized.

6.1.1 Summary of Contributions

- Inspired by supervised learning, a novel method to learn a more generalizable probabilistic model from multiple related source models is proposed in Chapter 3. In particular, in order to mitigate the potential negative transfer, a novel neural network structure is designed. By fine-tuning the model via standard reinforcement learning techniques, the final target model can predict higher-quality solutions over a wider range of optimization tasks. Case studies on traveling

salesman problems as well as knapsack problems have shown the generalization ability over combinatorial optimization problems with various sizes.

- In Chapter 4, an adaptive online transfer optimization framework for evolutionary computation is introduced in the literature of evolutionary computation. This novel model-based transfer evolutionary algorithm is capable of online learning and exploitation of similarities across distinct optimization problems, in a manner that minimizes the threat of negative transfer. While the knowledge transfer strategies in most existing approaches generally assume knowledge transfer is beneficial to the target optimization task, this framework automatically modulate the amount of knowledge transferred from various source tasks. As long as the experience on solving source optimization task can be represented or encoded as a probabilistic model, it can be incorporated into the knowledge base for future usage. Theoretical analysis of the proposed approach is also conducted, demonstrating that such knowledge-enhancement scheme guarantees to facilitate global convergence of the EA. Rigorous experimental verification of this framework on a diverse test suite empirically showcases the efficacy of the proposed unified framework.

- Multi-source surrogate-assisted transfer optimization is studied to deal with computationally expensive target optimization tasks in Chapter 5. With accumulated experiences on source optimization tasks, the computational complexity of the widely used multi-task/transfer Gaussian process surrogate may become an issue. Therefore, a novel scalable transfer Gaussian process is proposed, in order to speedup the optimization process on the optimization task of interest. The efficacy of the proposed model was verified not only in several machine learning tasks, but also in multi-source surrogate-assisted transfer optimization.

6.2 Future Work

There are some potential research directions to extend the present works introduced in this thesis.

- The proposed Tr-BCM is at times sensitive to the data partitioning, due to the disjoint steps of partitioning and hyperparameter optimization. Therefore, one promising research direction is to incorporate variational inference methods into the aggregation model, to dynamically allocate data points to each local model. In this way, joint training could be achieved. Similar to the method proposed in [142], the allocation of data points to each local experts could be based on the proximity to the experts probabilistically.

- Transfer learning in deep reinforcement learning can be a little trickier than transfer learning in neural networks and other supervised learning tasks. Specifically for model-free deep reinforcement learning techniques, negative transfer can be a real threat. While the proposed transfer learning strategy can do well for solving TSPs and knapsack problems, future work could be done to enhance the applicability of the proposed model for solving wider range of combinatorial optimization problems and across different problem types.

- In the double pole balancing problem, 13 source optimization tasks are utilized to help optimize the most difficult target optimization task. While in real-world settings, different types of optimization tasks, with heterogeneous search space, might be encountered. Therefore, one future direction could be to generalize AMT framework to encompass an even larger variety of practically relevant optimization problems. Furthermore, system-level implementations will be considered to assess the scalability of the algorithm for potential deployment in large-scale IoT/cloud-based applications.

- Another future direction could be to incorporate domain adaptation for efficient knowledge transfer. Domain adaptation can be a better strategy to handle heterogeneous search spaces between source and target optimization tasks, instead of using *universal search space*.

- The proposed Tr-BCM and its extension to Bayesian optimization are suitable for decentralized computation, thus, it would be interesting to integrate the proposed transfer Bayesian optimization into IoT/Fog computing platforms.

Appendix

Additional Experimental Study for Chapter 4

Knapsack problem (KP) is a classical NP-hard problem in discrete (combinatorial) optimization, popularly studied in the operations research literature. It offers several practical applications in many different areas, including logistics, auction winner determination, investment decision making, as well as portfolio optimization.

The objective of the problem, given a knapsack of capacity C, and a set of d items, each with a weight w_i and a value q_i, is to find a selection of items such that the total value is maximized without violating the capacity constraint. The mathematical formulation of the KP is defined as follows:

$$\begin{aligned} &\max \sum_{i=1}^{d} q_i x_i \\ &\text{s.t. } \sum_{i=1}^{d} w_i x_i \leq C \text{ and } x_i \in \{0,1\}, \end{aligned} \tag{6.1}$$

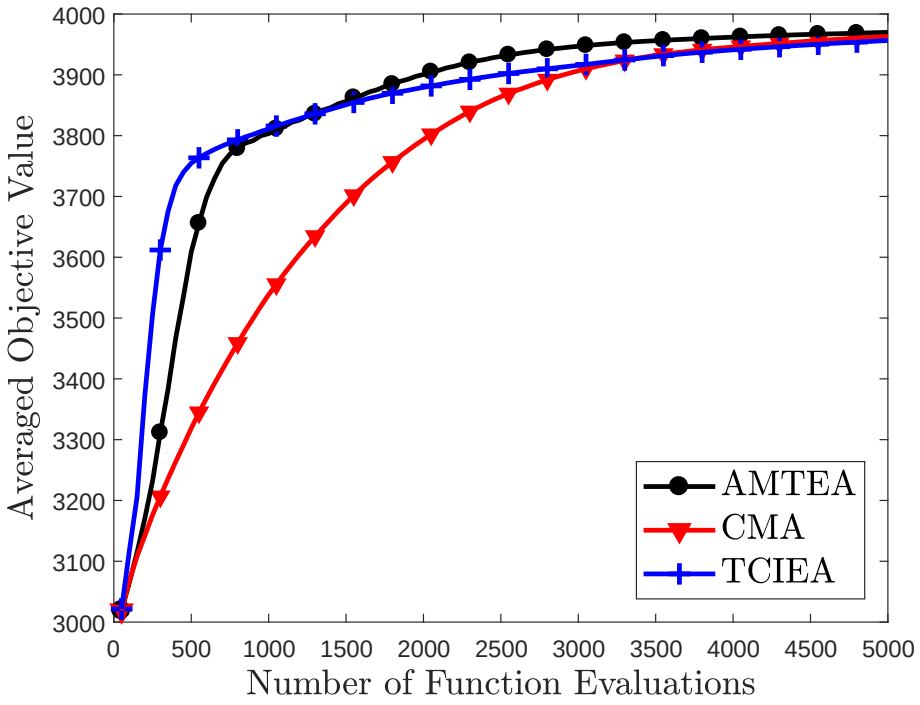
Here, $x_i = 1$ indicates that the i^{th} item is selected, while $x_i = 0$ indicates that the i^{th} item is not selected.

In this work, artificial KP instances are generated using the technique proposed in [143]. Accordingly, the KP instances are categorized into *uncorrelated* (uc), *weakly correlated* (wc), and *strongly correlated* (sc), depending on the relationship between the w's and q's. Further, there are considered to be two types of knapsacks: *restrictive capacity* (rc) and *average capacity* (ac). For a restrictive knapsack, only a small number of items are expected to be selected, while for an average knapsack, the number of

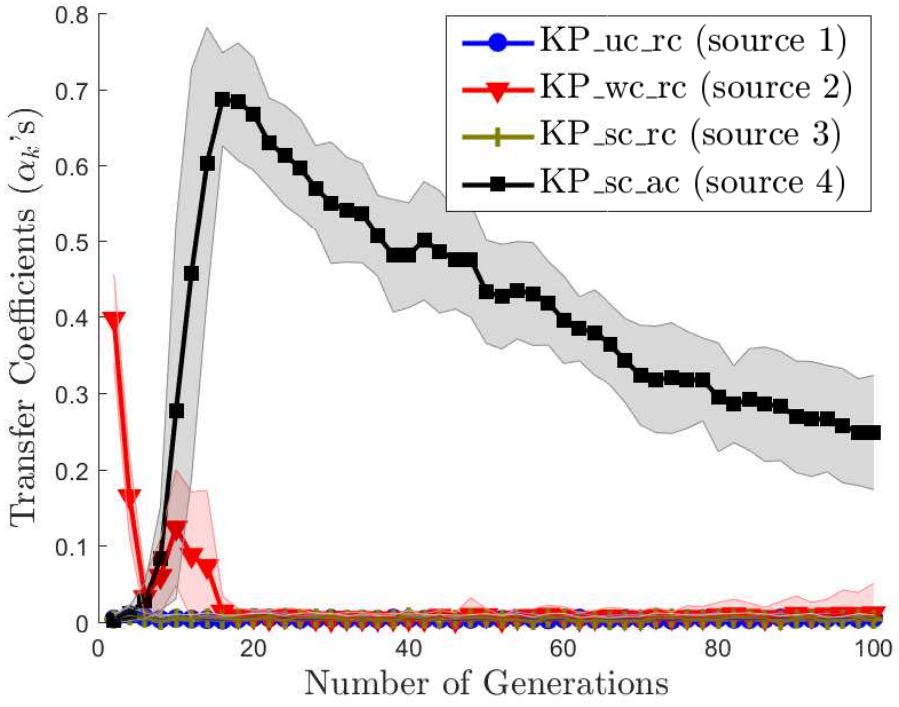
selected items will be larger. We concatenate the two subcategories to form six $d = 1000$ dimensional KP instances. For example, a KP problem denoted as 'KP_uc_rc' indicates that the w's and q's are weakly correlated, and the knapsack is of restrictive capacity.

In the experimental study, it is noted that the evolutionary search can often cause the capacity constraint of the knapsack to be violated. Therefore, a local solution refinement (repair) step is included in all solvers following Dantzig's greedy approximation algorithm [143]. Next, the optimization runs of four KP instances, namely, 'KP_uc_rc', 'KP_wc_rc', 'KP_sc_rc', and 'KP_sc_ac', are considered to act as sources from which source probabilistic models are drawn. Instances 'KP_wc_ac' and 'KP_uc_ac' act as the target optimization problems of interest. We compare the proposed AMTEA with CMA (since local refinement is applied) and TCIEA. The transfer interval is set as $\Delta = 2$. Fig. 6.1 and 6.2 show the averaged results obtained (over 30 independent runs). It is clear that algorithms AMTEA and TCIEA, with the scope of knowledge transfer, perform significantly better than CMA. While the TCIEA is found to rapidly increase the obtained objective function values in the initial stages of evolution, it is eventually surpassed by AMTEA on both the target tasks.

Most importantly, Figs. 6.1b and 6.2b show the source-target similarities captured by the AMTEA. While all the KP instances belong to distinct subcategories, it is remarkable to note that the algorithm successfully identifies the source task that is intuitively expected to be most relevant. To elaborate, since both target tasks are characterized by average knapsack capacity, a relatively large number of items must be selected. However, among the set of source probabilistic models available, only 'KP_sc_ac' belongs to the average knapsack capacity category. All other sources are characterized by restrictive knapsacks where only a small number of items can be selected. Therefore, simple intuition dictates that the optimum solutions of target tasks 'KP_wc_ac' and 'KP_uc_ac', respectively, should be most similar to that of source 'KP_sc_ac'. The expectation is precisely borne out by experiments, which highlights the key contribution of the work with regard to automatically unveiling the synergies between distinct optimization tasks online.

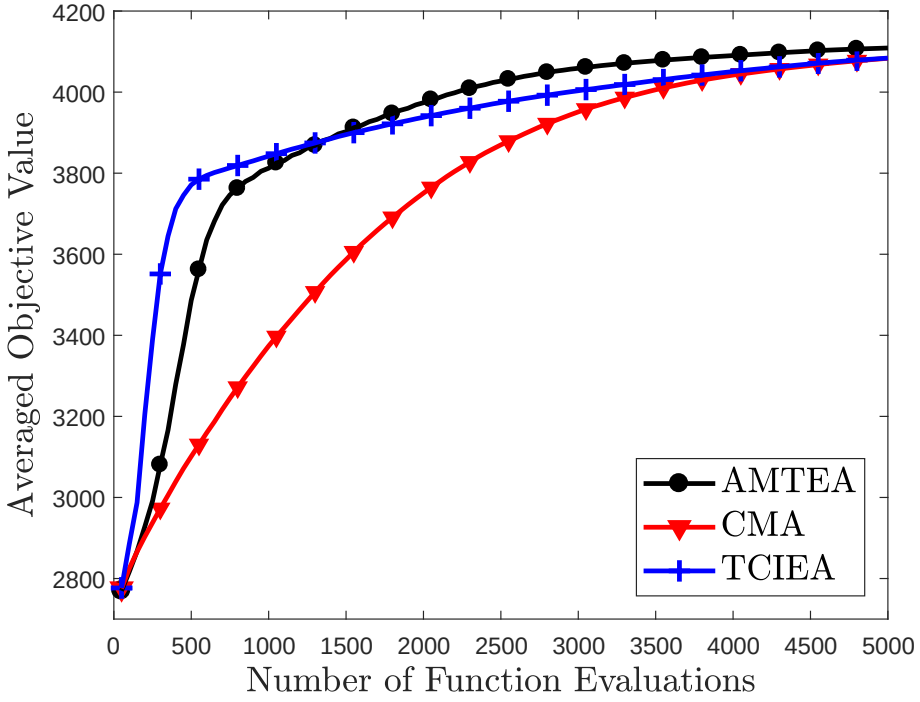


(a) KP_wc_ac

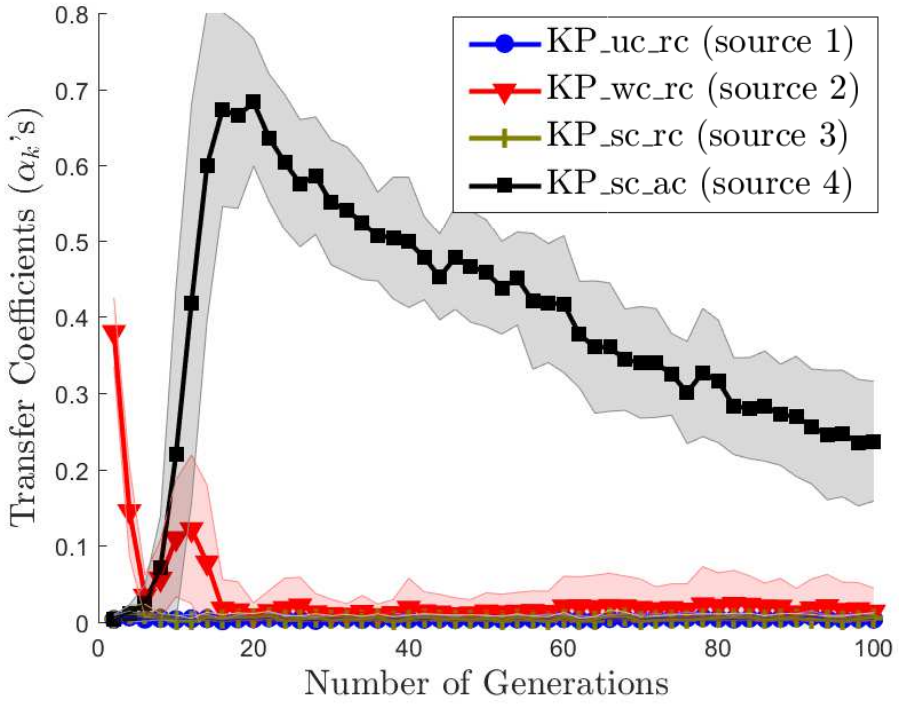


(b) Transfer coefficients learned for KP_wc_ac

FIGURE 6.1: Convergence trends and transfer coefficients for 'KP_wc_ac' (the shaded region spans one standard deviation either side of the mean). 'KP_uc_rc', 'KP_wc_rc', 'KP_sc_rc', and 'KP_sc_ac' serve as source optimization problems.



(a) KP_uc_ac



(b) Transfer coefficients learned for KP_uc_ac

FIGURE 6.2: Convergence trends and transfer coefficients for 'KP_uc_ac' (the shaded region spans one standard deviation either side of the mean). 'KP_uc_rc', 'KP_wc_rc', 'KP_sc_rc', and 'KP_sc_ac' serve as source optimization problems.

List of Publications

1. **Bingshui Da**, Yew-Soon Ong, Abhishek Gupta, Liang Feng, and Haitao Liu. Fast transfer gaussian process regression with large-scale sources. *Knowledge-Based Systems*, 2018. ISSN 0950-7051. doi: 10.1016/j.knosys.2018.11.029

2. **Bingshui Da**, Abhishek Gupta, and Yew-Soon Ong. Curbing negative influences online for seamless transfer evolutionary optimization. *IEEE Trans. Cybernetics*, pages 1–14, 2018. ISSN 2168-2267. doi: 10.1109/TCYB.2018.2864345

3. **Bingshui Da**, Yew-Soon Ong, Liang Feng, A. K. Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results. *CoRR*, abs/1706.03470, 2017

4. Abhishek Gupta, **Bingshui Da**, Yuan Yuan, and Yew-Soon Ong. On the emerging notion of evolutionary multitasking: A computational analog of cognitive multitasking. In *Recent Advances in Evolutionary Multi-objective Optimization*, volume 20 of *Adaptation, Learning, and Optimization*, pages 139–157. Springer, 2017

5. **Bingshui Da**, Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Evolutionary multitasking across single and multi-objective formulations for improved problem solving. In *CEC*, pages 1695–1701. IEEE, 2016

6. Abhishek Gupta, Yew-Soon Ong, **Bingshui Da**, Liang Feng, and Stephanus Daniel Handoko. Landscape synergy in evolutionary multitasking. In *CEC*, pages 3076–3083. IEEE, 2016

7. Chen Wang and Bingshui Da. Path determination under stochastic travel times using target-oriented robust optimization. In *SmartCity*, pages 159–164. IEEE Computer Society, 2015

8. **Bingshui Da**, Abhishek Gupta, Yew-Soon Ong, and Liang Feng. The boon of gene-culture interaction for effective evolutionary multitasking. In *ACALCI*, volume 9592 of *Lecture Notes in Computer Science*, pages 54–65. Springer, 2016

Abbreviations

AE	Autoencoding Evolutionary
AMT	Adaptive Model-based Transfer
AMTEA	Adaptive Model-based Transfer Evolutionary Algorithm
BCM	Bayesian Committee Machine
BERT	Bidirectional Encoder Representations from Transformers
BO	Bayesian Optimization
CEA	Canonical Evolutionary Algorithm
CMA	Canonical Memetic Algorithm
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EI	Expected Improvement
EM	Expectation-Maximization
FNN	Feedforward Neural Network
FRP	Fiber-Reinforced polymer
GA	Genetic Algorithm
GP	Gaussian Process

gPoE	generalized Product of Experts
GP-LCB	Gaussian Process Lower Confidence Bound
GP-UCB	Gaussian Process Upper Confidence Bound
IoT	Internet of Things
I/C-LCM	Injection/Compression Liquid Composite Molding
KD	Knowledge Distillation
KL	Kullback-Leibler
KP	Knapsack Problem
LCM	Liquid Composite Molding
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOO	Multi-Objective Optimization
NES	natural evolution strategies
NSGA-II	Non-dominated Sorting Genetic Algorithm II
PI	Probability of Improvement
PN	Pointer Network
PoE	Product of Experts
PSD	Positive Semidefinite
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RTM	Resin Transfer Molding
SAEA	Surrogate-Assisted Evolutionary Algorithm

TCIEA	Transfer Case-Injected Evolutionary Algorithm
TGP	Transfer Gaussian Process
tPN	Transfer Pointer Network
Tr-BCM	Transfer Bayesian Committee Machine
Tr-BCM-ls	Transfer Bayesian Committee Machine with localized similarity
TSP	Traveling Salesman Problem
TSGP	Transfer Stacking Gaussian Process

Notations of Chapter 3

$\boldsymbol{\pi}$	a tour for a TSP
$L(\boldsymbol{\pi}\|s)$	total length of the tour $\boldsymbol{\pi}$
s	a sequence of n cities in a two dimensional space
$\mathbf{x}_i$	the coordinates of the ith city (node)
$p_\theta(\boldsymbol{\pi}\|s)$	a probabilistic distribution parameterized by θ given a problem instance s
θ	parameters of the probabilistic distribution (i.e., parameters of the Pointer network)
$\mathbf{e}_i$	the embedded input of city $\mathbf{x}_i$
t	time step at decoding step
$\mathbf{a}_t$	a variable-length alignment vector
$\mathbf{h}_t$	the memory state of the RNN cell at step t
$\mathbf{u}_t$	an intermediate vector used to compute alignment vector $\mathbf{a}_t$
$\mathbf{c}_t$	the context vector at time step t
$\tilde{u}_t^i$	an intermediate vector used to compute final output
$\mathbf{v}_a$, $\mathbf{W}_a$, $\mathbf{v}_c$, $\mathbf{W}_c$	trainable parameters of a Pointer network

$J(\theta\|s)$	the expected tour length given a TSP described by a graph s
B	batch size
$b(s)$	the baseline function, which estimates the expected tour length.
θ_v	parameters of the critic network
$\mathcal{L}_v$	loss function for the critic network
K	number of source models
$\mathbf{z}_k, t$	logit vector at time t from kth source model
$\mathcal{L}_{\mathbf{KL}}$	Kullback-Leibler (KL) divergence, also referred as distillation loss
τ	temperature in the distillation loss
$\mathbf{v}_k, \mathbf{W}_k$	trainable parameters of a transfer Pointer network
C	capacity of a knapsack
d	number of items in a knapsack
w_i, q_i	weight and value of an item

Notations of Chapter 4

K	number of source and target optimization tasks
$\mathcal{T}_k$	kth optimization task
f_k	the objective function of kth task
φ_k	the probabilistic model for kth optimization task
ϵ_k	a small convergence tolerance threshold for kth optimization task
$\mathcal{X}$	a universal search space
$p(\mathbf{x})$	the latent distribution of the current population
α_k	mixture/transfer coefficient of kth optimization task
d_k	search space dimensionality of kth optimization task
$d_{\textbf{unified}}$	dimensionality of a unified space $\mathcal{X}$
t	tth generation of an EA
N	number of solutions within a generation of an EA
D_t	a dataset of solutions in a universal search space $\mathcal{X}$ at t^{th} generation
$p(\mathbf{x}\|t)$	the latent probability density function describing the target population dataset D_t

$q(\mathbf{x}\|t)$	approximation of the latent probability density function $p(\mathbf{x}\|t)$
D_{test}	out-of-sample test dataset
L	likelihood of the out-of-sample test dataset
v	number of folds in cross validation procedure
Δ	transfer interval
$P^s(t)$	parent individuals at tth generation
$P^c(t)$	child individuals at tth generation

Notations of Chapter 5

d	the dimensionality of the source and target data
$\mathbf{X}_{\mathcal{S}}$	source data
$\mathbf{y}_{\mathcal{S}}$	labels for source data
$n_{\mathcal{S}}$	number of source data
$\mathbf{X}_{\mathcal{T}}$	target data
$\mathbf{y}_{\mathcal{T}}$	labels for target data
$n_{\mathcal{T}}$	number of target data
$\mathcal{D}_{\mathcal{S}}$	overall source data
$\mathcal{D}_{\mathcal{T}}$	overall target data
$\epsilon_{\mathcal{S}}$	additive noise term for source task
$\epsilon_{\mathcal{T}}$	additive noise term for target task
$\sigma_{\mathcal{S}}^2$	variance of additive noise for source task
$\sigma_{\mathcal{T}}^2$	variance of additive noise for target task
$f_{\mathcal{S}}$	the latent function of source task
$f_{\mathcal{T}}$	the latent function of target task
$\mu(\mathbf{x})$	mean function of a Gaussian process
$k(\cdot, \cdot)$	covariance function, or kernel

$\tilde{k}(\cdot,\cdot)$	transfer covariance function, or transfer kernel
λ	the source-target similarity
$\widetilde{\mathbf{K}}$	transfer covariance matrix
$\boldsymbol{\theta}$	the hyperparameters of the transfer covariance function
$\mathcal{D}_{\mathcal{S}_i}$	ith source subset
$\mathcal{M}_i$	$i$the local transfer Gaussian process expert
$\mathscr{D}_{\mathcal{S}_i}$	the set of all source datasets with indices smaller or equal to i
$\mathcal{N}(\cdot,\cdot)$	normal distribution, or Gaussian distribution
K^f	a matrix capturing the inter-task (between source and target) and intra-task (between different source subsets) similarities
$a_{\mathbf{LCB}}$	the acquisition function of GP-LCB

Bibliography

[1] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646. MORGAN KAUFMANN PUBLISHERS, 1996.

[2] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. knowledge and data engineering*, 22(10):1345–1359, 2010.

[3] Alan Tan Wei Min, Ramon Almandoz, Abhishek Gupta, and Ong Yew Soon. Coping with data scarcity in aircraft engine design. *AIAA Multidisciplinary Analysis and Optimization Conference*, 2017.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[5] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 22(2):199–210, 2011.

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520. Omnipress, 2011.

[7] Peilin Zhao and Steven C. H. Hoi. OTL: A framework of online transfer learning. In *ICML*, pages 1231–1238. Omnipress, 2010.

[8] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[9] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*, 2016.

[10] Yee Whye Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *NIPS*, pages 4499–4509, 2017.

[11] Sushil J Louis and John McDonnell. with case-injected genetic algorithms. *IEEE Trans. Evol. Comput.*, 8(4):316–328, 2004.

[12] Alan Tan Wei Min, Ramon Almandoz, Abhishek Gupta, and Ong Yew Soon. Knowledge transfer through machine learning in aircraft design. *Computational Intelligence Magazine*, 2017, accepted.

[13] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Multi-task bayesian optimization. In *NIPS*, pages 2004–2012, 2013.

[14] Liang Feng, Yew-Soon Ong, Ah-Hwee Tan, and Ivor W Tsang. Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems. *Meme. Comput.*, 7(3):159–180, 2015.

[15] **Bingshui Da**, Abhishek Gupta, and Yew-Soon Ong. Curbing negative influences online for seamless transfer evolutionary optimization. *IEEE Trans. Cybernetics*, pages 1–14, 2018. ISSN 2168-2267. doi: 10.1109/TCYB.2018.2864345.

[16] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Multifactorial evolution: toward evolutionary multitasking. *IEEE Trans. Evol. Comput.*, 20(3):343–357, 2016.

[17] **Bingshui Da**, Yew-Soon Ong, Liang Feng, A. K. Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results. *CoRR*, abs/1706.03470, 2017.

[18] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NIPS*, pages 2692–2700, 2015.

[19] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! 2019.

[20] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *CoRR*, abs/1811.06128, 2018.

[21] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, Tom Sercu, Kartik Audhkhasi, Abhinav Sethy, Markus Nußbaum-Thom, and Andrew Rosenberg. Knowledge distillation across ensembles of multilingual models for low-resource languages. In *ICASSP*, pages 4825–4829. IEEE, 2017.

[22] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *CoRR*, abs/1904.09482, 2019.

[23] David Pardoe and Peter Stone. Boosting for regression transfer. In *ICML*, pages 863–870. Omnipress, 2010.

[24] Pengfei Wei, Ramón Sagarna, Yiping Ke, Yew-Soon Ong, and Chi-Keong Goh. Source-target similarity modelings for multi-source transfer gaussian process regression. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3722–3731. PMLR, 2017.

[25] **Bingshui Da**, Yew-Soon Ong, Abhishek Gupta, Liang Feng, and Haitao Liu. Fast transfer gaussian process regression with large-scale sources. *Knowledge-Based Systems*, 2018. ISSN 0950-7051. doi: 10.1016/j.knosys.2018.11.029.

[26] Abhishek Gupta, Yew Soon Ong, and Liang Feng. Insights on transfer optimization: Because experience is the best teacher. *IEEE Trans. Emerging Topics in Computational Intelligence*, PP(99):1–14, 2017. doi: 10.1109/TETCI.2017.2769104.

[27] Marjan Kaedi and Nasser Ghasem-Aghaee. Biasing bayesian optimization algorithm using case based reasoning. *Knowledge-Based Systems*, 24(8):1245–1253, 2011.

[28] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[29] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[30] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[31] Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, pages 6351–6361, 2017.

[32] Alan Wei Min Tan, Yew Soon Ong, Abhishek Gupta, and Chi Keong Goh. Multiproblem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems. *IEEE Trans. Evolutionary Computation*, PP (99):1–1, 2017. doi: 10.1109/TEVC.2017.2783441.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

[35] MohammadReza Nazari, Afshin Oroojlooy, Lawrence V. Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *NeurIPS*, pages 9861–9871, 2018.

[36] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[37] Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj K. Singh. Learning heuristics over large graphs via deep reinforcement learning. *CoRR*, abs/1903.03332, 2019.

[38] Pádraig Cunningham and Barry Smyth. Case-based reasoning in scheduling: reusing solution components. *Int. J. Prod. Res.*, 35(11):2947–2962, 1997.

[39] Shlomo Israel and Amiram Moshaiov. Bootstrapping aggregate fitness selection with evolutionary multi-objective optimization. *Parallel Problem Solving from Nature-PPSN XII*, pages 52–61, 2012.

[40] Barış Koçer and Ahmet Arslan. Genetic transfer learning. *Expert Systems with Applications*, 37(10):6997–7002, 2010.

[41] Liang Feng, Yew-Soon Ong, Meng-Hiot Lim, and Ivor W Tsang. Memetic search with interdomain learning: A realization between cvrp and carp. *IEEE Trans. Evol. Comput.*, 19(5):644–658, 2015.

[42] L. Feng, Y. S. Ong, S. Jiang, and A. Gupta. Autoencoding evolutionary search with learning across heterogeneous problems. *IEEE Trans. Evol. Comput.*, PP (99):1–1, 2017. ISSN 1089-778X. doi: 10.1109/TEVC.2017.2682274.

[43] Muhammad Iqbal, Will N Browne, and Mengjie Zhang. Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. *IEEE Trans. Evolutionary Computation*, 18(4):465–480, 2014.

[44] Muhammad Iqbal, Will N Browne, and Mengjie Zhang. Extracting and using building blocks of knowledge in learning classifier systems. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 863–870. ACM, 2012.

[45] Isidro M Alvarez, Will N Browne, and Mengjie Zhang. Human-inspired scaling in learning classifier systems: Case study on the n-bit multiplexer problem set. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 429–436. ACM, 2016.

[46] Maria Salamó and Maite López-Sánchez. Adaptive case-based reasoning using retention and forgetting strategies. *Knowl.-Based Syst.*, 24(2):230–247, 2011.

[47] Ryan Meuth, Meng-Hiot Lim, Yew-Soon Ong, and Donald C Wunsch. A proposition on memes and meta-memes in computing for higher-order learning. *Memetic Computing*, 1(2):85–100, 2009.

[48] Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. Multiobjective multifactorial optimization in evolutionary multitasking. *IEEE Trans. cybernetics*, 2017.

[49] Yew-Soon Ong and Abhishek Gupta. Evolutionary multitasking: a computer science view of cognitive multitasking. *Cognitive Computation*, 8(2):125–142, 2016.

[50] C. Yang, J. Ding, Y. Jin, C. Wang, and T. Chai. Multitasking multiobjective evolutionary operational indices optimization of beneficiation processes. *IEEE Trans. Automation Science and Engineering*, 2018. ISSN 1545-5955. doi: 10.1109/TASE.2018.2865593.

[51] H. Li, Y. Ong, M. Gong, and Z. Wang. Evolutionary multitasking sparse reconstruction: Framework and case study. *IEEE Trans. Evolutionary Computation*, 2018. ISSN 1089-778X. doi: 10.1109/TEVC.2018.2881955.

[52] Jing Tang, Yingke Chen, Zixuan Deng, Yanping Xiang, and Colin Paul Joy. A group-based approach to improve multifactorial evolutionary algorithm. In *IJCAI*, pages 3870–3876. ijcai.org, 2018.

[53] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. Ong, K. Tan, and A. K. Qin. Evolutionary multitasking via explicit autoencoding. *IEEE Trans. Cybernetics*, 2018. ISSN 2168-2267. doi: 10.1109/TCYB.2018.2845361.

[54] Kavitesh K. Bali, Yew-Soon Ong, Abhishek Gupta, and Puay Siew Tan. Multifactorial evolutionary algorithm with online transfer parameter estimation: Mfea-ii. *IEEE Trans. Evol. Comput.*, 2019. ISSN 1089-778X. doi: 10.1109/TEVC.2019.2906927.

[55] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.*, 9(1):3–12, 2005.

[56] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[57] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Trans. Evolutionary Computation*, 2018. ISSN 1089-778X. doi: 10.1109/TEVC.2018.2869001.

[58] Yew S Ong, Prasanth B Nair, and Andrew J Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41 (4):687–696, 2003.

[59] Yew-Soon Ong, Prasanth B. Nair, and Kai Yew Lum. Max-min surrogate-assisted evolutionary algorithm for robust design. *IEEE Trans. Evolutionary Computation*, 10(4):392–404, 2006.

[60] Zongzhao Zhou, Yew-Soon Ong, Prasanth B. Nair, Andy J. Keane, and Kai Yew Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 37(1):66–76, 2007.

[61] Yaochu Jin, Michael Hüsken, and Bernhard Sendhoff. Quality measures for approximate models in evolutionary computation. In *GECCO*, pages 170–173, 2003.

[62] Dudy Lim, Yaochu Jin, Yew-Soon Ong, and Bernhard Sendhoff. Generalizing surrogate-assisted evolutionary computation. *IEEE Trans. Evolutionary Computation*, 14(3):329–355, 2010.

[63] Tinkle Chugh, Yaochu Jin, Kaisa Miettinen, Jussi Hakanen, and Karthik Sindhya. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Trans. Evolutionary Computation*, 22(1):129–142, 2018.

[64] D. Guo, Y. Jin, J. Ding, and T. Chai. Heterogeneous ensemble-based infill criterion for evolutionary multiobjective optimization of expensive problems. *IEEE Trans. Cybernetics*, pages 1–14, 2018. ISSN 2168-2267. doi: 10.1109/TCYB.2018.2794503.

[65] J. Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evolutionary Computation*, 10(1):50–66, 2006.

[66] J. Tian, Y. Tan, J. Zeng, C. Sun, and Y. Jin. Multi-objective infill criterion driven gaussian process assisted particle swarm optimization of high-dimensional expensive problems. *IEEE Transactions on Evolutionary Computation*, 2018. ISSN 1089-778X. doi: 10.1109/TEVC.2018.2869247.

[67] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012.

[68] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2171–2180. JMLR.org, 2015.

[69] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018.

[70] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[71] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, pages 1015–1022. Omnipress, 2010.

[72] Alistair Shilton, Sunil Kumar Gupta, Santu Rana, and Svetha Venkatesh. Regret bounds for transfer learning in bayesian optimisation. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 307–315. PMLR, 2017.

[73] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[74] Edwin V Bonilla, Kian Ming Adam Chai, and Christopher KI Williams. Multi-task gaussian process prediction. In *NIPs*, volume 20, pages 153–160, 2007.

[75] Kian Ming Adam Chai. Generalization errors and learning curves for regression with multi-task gaussian processes. In *NIPS*, pages 279–287. Curran Associates, Inc., 2009.

[76] Edwin V. Bonilla, Felix V. Agakov, and Christopher K. I. Williams. Kernel multi-task learning using task-specific features. In *AISTATS*, volume 2 of *JMLR Proceedings*, pages 43–50. JMLR.org, 2007.

[77] Mauricio A. Álvarez and Neil D. Lawrence. Sparse convolved gaussian processes for multi-output regression. In *NIPS*, pages 57–64. Curran Associates, Inc., 2008.

[78] Gayle Leen, Jaakko Peltonen, and Samuel Kaski. Focused multi-task learning using gaussian processes. In *ECML/PKDD (2)*, volume 6912 of *Lecture Notes in Computer Science*, pages 310–325. Springer, 2011.

[79] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.

[80] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. Remarks on multi-output gaussian process regression. *Knowl.-Based Syst.*, 144:102–121, 2018.

[81] Bin Cao, Sinno Jialin Pan, Yu Zhang, Dit-Yan Yeung, and Qiang Yang. Adaptive transfer learning. In *AAAI*, volume 2, page 7, 2010.

[82] Xuezhi Wang, Tzu-Kuo Huang, and Jeff G. Schneider. Active transfer learning under model shift. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1305–1313. JMLR.org, 2014.

[83] Melih Kandemir. Asymmetric transfer learning with deep gaussian processes. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 730–738. JMLR.org, 2015.

[84] Andreas C. Damianou and Neil D. Lawrence. Deep gaussian processes. In *AISTATS*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 207–215. JMLR.org, 2013.

[85] Neeti Wagle and Eric W. Frew. Forward adaptive transfer of gaussian process regression. *J. Aerospace Inf. Sys.*, 14(4):214–231, 2017.

[86] Mauricio A. Álvarez and Neil D. Lawrence. Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12: 1459–1500, 2011.

[87] Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[88] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NIPS*, pages 1257–1264, 2005.

[89] James Hensman, Nicoló Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *UAI*. AUAI Press, 2013.

[90] Yarin Gal, Mark van der Wilk, and Carl E. Rasmussen. Distributed variational inference in sparse gaussian process regression and latent variable models. In *NIPS*, pages 3257–3265, 2014.

[91] Zhenwen Dai, Mauricio A. Álvarez, and Neil D. Lawrence. Efficient modeling of latent information in supervised learning using gaussian processes. In *NIPS*, pages 5137–5145, 2017.

[92] David A. Moore and Stuart J. Russell. Gaussian process random fields. In *NIPS*, pages 3357–3365, 2015.

[93] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940, 2016.

[94] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *NIPS*, pages 1008–1014. The MIT Press, 1999.

[95] Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *CoRR*, abs/1511.06295, 2015.

[96] Kaixiang Lin, Shu Wang, and Jiayu Zhou. Collaborative deep reinforcement learning. *CoRR*, abs/1702.05796, 2017.

[97] Yew-Soon Ong, Meng Hiot Lim, and Xianshun Chen. Memetic computation—past, present & future [research frontier]. *IEEE Comput. Intell. Mag.*, 5 (2):24–31, 2010.

[98] Xianshun Chen, Yew-Soon Ong, Meng-Hiot Lim, and Kay Chen Tan. A multi-facet survey on memetic computation. *IEEE Trans. on Evol. Comput.*, 15(5): 591–607, 2011.

[99] Kumara Sastry, David E. Goldberg, and Xavier Llorà. Towards billion-bit optimization via a parallel estimation of distribution algorithm. In *GECCO*, pages 577–584. ACM, 2007.

[100] Padhraic Smyth and David Wolpert. Stacked density estimation. In *NIPS*, pages 668–674, 1997.

[101] James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA J Comput.*, 6(2):154–160, 1994.

[102] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal Process. Mag.*, 13(6):47–60, 1996.

[103] Moritz Blume. Expectation maximization: A gentle introduction. *Technical University of Munich Institute for Computer Science*, 2002.

[104] Qingfu Zhang and Heinz Muhlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Trans. Evol. Comput.*, 8(2):127–136, 2004.

[105] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[106] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[107] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[108] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[109] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. evolutionary computation*, 6(2):182–197, 2002.

[110] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evolutionary Computation*, 11(6):712–731, 2007.

[111] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.

[112] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017.

[113] Heinz Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.

[114] Ram Bhushan Agrawal, K Deb, and RB Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[115] Kalyanmoy Deb and Debayan Deb. Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4(1):1–28, 2014.

[116] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[117] Kalyanmoy Deb and David E Goldberg. Analyzing deception in trap functions. *Foundations of genetic algorithms*, 2:93–108, 1993.

[118] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2): 173–195, 2000.

[119] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145, 2005.

[120] Simon Huband, Philip Hingston, Luigi Barone, and R. Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evolutionary Computation*, 10(5):477–506, 2006.

[121] Siwei Jiang, Yew-Soon Ong, Jie Zhang, and Liang Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Trans. Cybernetics*, 44(12):2391–2404, 2014.

[122] Faustino J Gomez and Risto Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *IJCAI*, volume 99, pages 1356–1361, 1999.

[123] Alexis P Wieland. Evolving neural network controllers for unstable systems. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 667–673. IEEE, 1991.

[124] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9(May):937–965, 2008.

[125] William Walbran. *Experimental validation of local and global force simulations for rigid tool liquid composite moulding processes*. PhD thesis, 2011.

[126] Abhishek Gupta. *Numerical modelling and optimization of non-isothermal, rigid tool liquid composite moulding processes*. PhD thesis, ResearchSpace@ Auckland, 2013.

[127] Sherry Hsu, Matthias Ehrgott, and Piaras Kelly. Optimisation of mould filling parameters of the compression resin transfer moulding process. In *Proceedings of the 45th Annual Conference of the ORSNZ*, 2010.

[128] PA Kelly and Simon Bickerton. A comprehensive filling and tooling force analysis for rigid mould lcm processes. *Composites Part A: Applied Science and Manufacturing*, 40(11):1685–1697, 2009.

[129] Abhishek Gupta and Piaras Kelly. Optimal galerkin finite element methods for non-isothermal liquid composite moulding process simulations. *International Journal of Heat and Mass Transfer*, 64:609–622, 2013.

[130] Volker Tresp. A bayesian committee machine. *Neural Computation*, 12(11): 2719–2741, 2000.

[131] Geoffrey E Hinton. Products of experts. 1999.

[132] Marc Peter Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1481–1490. JMLR.org, 2015.

[133] Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of gaussian process experts. In *NIPS*, pages 881–888. MIT Press, 2001.

[134] Chao Yuan and Claus Neubauer. Variational mixture of gaussian process experts. In *NIPS*, pages 1897–1904. Curran Associates, Inc., 2008.

[135] Andrew Gordon Wilson, David A. Knowles, and Zoubin Ghahramani. Gaussian process regression networks. In *ICML*. icml.cc / Omnipress, 2012.

[136] Trung V. Nguyen and Edwin V. Bonilla. Efficient variational inference for gaussian process regression networks. In *AISTATS*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 472–480. JMLR.org, 2013.

[137] Pedro García López, Alberto Montresor, Dick H. J. Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho P. Barcellos, Pascal Felber, and Etienne Rivière. Edge-centric computing: Vision and challenges. *Computer Communication Review*, 45(5):37–42, 2015.

[138] Fog Computing. the internet of things: Extend the cloud to where the things are. *Cisco White Paper*, 2015.

[139] Leslie Hogben. *Handbook of linear algebra*. CRC Press, 2006.

[140] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[141] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993.

[142] Trung V. Nguyen and Edwin V. Bonilla. Fast allocation of gaussian process experts. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 145–153. JMLR.org, 2014.

[143] Zbigniew Michalewicz and Jarosław Arabas. Genetic algorithms for the 0/1 knapsack problem. *Methodologies for Intelligent Systems*, pages 134–143, 1994.

[144] Abhishek Gupta, **Bingshui Da**, Yuan Yuan, and Yew-Soon Ong. On the emerging notion of evolutionary multitasking: A computational analog of cognitive multitasking. In *Recent Advances in Evolutionary Multi-objective Optimization*, volume 20 of *Adaptation, Learning, and Optimization*, pages 139–157. Springer, 2017.

[145] **Bingshui Da**, Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Evolutionary multitasking across single and multi-objective formulations for improved problem solving. In *CEC*, pages 1695–1701. IEEE, 2016.

[146] Abhishek Gupta, Yew-Soon Ong, **Bingshui Da**, Liang Feng, and Stephanus Daniel Handoko. Landscape synergy in evolutionary multitasking. In *CEC*, pages 3076–3083. IEEE, 2016.

[147] Chen Wang and Bingshui Da. Path determination under stochastic travel times using target-oriented robust optimization. In *SmartCity*, pages 159–164. IEEE Computer Society, 2015.

[148] **Bingshui Da**, Abhishek Gupta, Yew-Soon Ong, and Liang Feng. The boon of gene-culture interaction for effective evolutionary multitasking. In *ACALCI*, volume 9592 of *Lecture Notes in Computer Science*, pages 54–65. Springer, 2016.